Dipartimento di Informatica
Università degli Studi di Torino
C.so Svizzera 185
I-10149 Torino (Italy)
http://www.di.unito.it

Dottorato di Ricerca in Informatica
(IX ciclo)

PhD Thesis
(February, 1998)

# Non-standard type inference for functional programs

Ferruccio Damiani

Promotor: Prof. Mario Coppo
Co-promotor: Dott.$^{ssa}$ Paola Giannini

*A Carmen*

## Abstract

We propose a general framework for "non-standard type inference based" static analyses of typed functional programs and study inference systems (with and without conjunction) for strictness, totality, and dead-code analyses. A key feature of our framework is that it provides a foundation of the program analyses which is based directly on the operational semantics of the programming language considered. This results in a rather direct presentation which involves relatively little mathematical overhead. We develop a semantical investigation of the systems and discuss in some detail possible implementations of the presented analyses. In particular we give sound and complete inference algorithms for strictness and dead-code analyses without conjunction.

## Acknowledgements

First of all I wish to thank my promotors, Mario Coppo and Paola Giannini, for their invaluable help and guidance, and Betti Venneri, for her encouragement and support when I decided to apply for a PhD position in Torino.

I also thank Mariangiola Dezani-Ciancaglini, which has deeply influenced my way of doing research, and Stefano Berardi, that has been always present for hearing my conjectures and solving by doubts.

A special thank to Simona Ronchi della Rocca, for reading a draft of this thesis and giving me many useful comments.

Moreover I am in various ways indebted with the people of the Computer Science Department of the University of Torino, in particular with the members of the "Semantics and Logics of Computation" group and with my colleagues PhD students (especially Matteo Baldoni and Viviana Bono).

Outside of the Department, I would like to thank Frédéric Prost (Ecole Normale Supérieure de Lyon) which found examples of dead-code in programs extracted from proofs (using Coq) for the conjunctive dead-code analysis presented in Chapter 8 of this thesis, Grzegorz Gruziński (Warsaw University), for interesting discussions about inference algorithms for strictness, and Kirsten Lackner Solberg Gasser (Aarhus University), for answering some questions about her work.

I would like also to thank Martin Bunder (Wollongong University) for reading[1] a draft of Part I of the thesis and giving me some suggestions to improve the presentation.

Finally I thank the referees Furio Honsell (Udine University) and Rinus Plasmeijer (Nijmegen University) for many useful comments and suggestions.

[1] While visiting the Computer Science Department of the University of Torino in November/December 1997.

# Contents

# Part I

# A framework for non-standard type inference

# Chapter 1

# Introduction

## 1.1 Functional programming languages

*Functional programming languages* are so called because they rely on the notions of *function definition* and *function application*. A functional program is written as a function that receives the input as its argument and returns the output as result. Typically a functional program is defined in terms of other functions, which in turn are defined in term of other functions, and so on, until the functions are primitives of the language. All these functions are much like ordinary mathematical functions, and this has made functional programming languages very popular in research and educational environments.

The foundation of functional programming languages lies in the $\lambda$-*calculus* (see [Barendregt, 1984; Hindley, 1997; Barendregt, 1992]), a theory of functions introduced by the logician Alonzo Church in the 1930's. In particular the $\lambda$-calculus term formation rules:

- "$\lambda$-abstraction of a variable $x$ from a term $M$" ($\lambda x.M$), and
- "application of a term $M_1$ to a term $M_2$" ($M_1 M_2$),

represent function definition an function application, respectively.

Many functional languages have been proposed after the early work in the 1960's (see [Hughes, 1989] for a survey). Depending on the semantics of function application, functional languages can be distinguished in two classes.

- *Eager* (or *strict*) languages, in which function application has a *call-by-value* semantics. I.e., when the function is applied to an actual parameter, the function's formal parameter is bound to the value obtained by evaluating the actual parameter.
- *Lazy* (or *non-strict*) languages, in which function application has a *call-by-name* semantics. I.e., when the function is applied to an actual parameter, the function's formal parameter is bound to the unevaluated actual parameter and, each time the

value of the formal parameter is required, the actual parameter is evaluated. For efficiency, application in lazy languages is usually implemented using *call-by-need*, which is the same as call-by-name, except that after the first evaluation of the actual parameter its value is retained and used whenever the formal parameter is subsequently required.

According to the above definition, Scheme [Abelson and Susman, 1985], ML [Milner *et al.*, 1990; Milner *et al.*, 1991; Paulson, 1996; Ullman, 1993] and Caml [Leroy, 1997] are eager, whereas Miranda [Turner, 1985], Haskell [Peterson *et al.*, 1997] and Clean [Plasmeijer and van Eekelen, 1997; de Mast *et al.*, 1997] are lazy.

Much has been written about the benefits of functional programming (see for instance [Hughes, 1989; Hudak, 1989]). One of the main drawbacks of the functional paradigm is that the efficiency is often rather poor.

## 1.2 Static analysis of functional programs

Modern compilers for functional programming languages use various kinds of transformations to improve the efficiency of the code produced. Such transformations are based on static analysis of programs and must be proved to preserve their semantics. It is of practical interest, then, to find reliable methods for detecting properties useful to perform such optimizations.

Various kinds of static analysis for functional languages have been proposed in literature. Some of them are listed below.

- *Strictness analysis.* A function is *strict* in an argument (formal parameter) if diverges whenever the argument (actual parameter) diverges. Strictness analysis aims to determine whether a function is strict. This information can be used, for instance, to transform a call-by-need into a call-by-value or to exploit parallelism in the evaluation of lazy functional programs.

- *Totality analysis* aims to determine whether the evaluation of an expression terminates. Totality information can be used in combination with strictness information (as explained in the introduction to Part II of this thesis, page 35).

- *Binding-time analysis* aims to determine the parts of a program that can be evaluated at compile time. This information can be used to perform partial evaluation.

- *Dead-code analysis* (also called *liveness analysis*) aims to determine the parts of a program that are useless for the computation of the value of a program. This information allows the simplification of programs by removing these useless parts (which are called dead-code).

- *Usage analysis* aims to determine how many times an expression is used. An expression that is used zero times is dead-code, while an expression that is used once can be garbage collected as soon as it is used.

- *Flow analysis* aims to determine the functions that are possibly called during the evaluation of an expression.

Two major approaches have been proposed to specify static analyses of functional languages: *abstract interpretation* and *non-standard type inference*.

Abstract interpretation was first introduced for imperative languages in [Cousot and Cousot, 1977; Cousot and Cousot, 1979] and then applied to functional languages by a number of authors, see [Mycroft, 1980; Mycroft, 1981; Burn *et al.*, 1986; Nielson, 1988; Abramsky, 1990a; Hunt and Sands, 1991; Cousot and Cousot, 1994]. In this approach programs are interpreted in abstract (usually finite) domains, instead of the standard denotational domains. Such abstract domains highlight the properties that are considered and are related to the standard denotational domains. The detection of properties then is obtained by evaluating the program over the abstract domain.

In the non-standard type inference approach instead, properties are represented by particular types (called *non-standard* types or *annotated* types), and program analyses are presented via a system of logical inference rules. These systems are usually built by two sets of rules. One set of rules (the "subtyping" rules) determines an entailment relation between the properties under investigation, and another set of rules (the "type assignment" rules) defines the way in which properties are assigned to the terms of the language. Non-standard type systems tailored to a specific analysis, such as strictness, totality, binding-time etc. have been introduced, see [Kuo and Mishra, 1989; Jensen, 1992; Benton, 1992a; Hankin and Le Métayer, 1994b; Solberg, 1995a; Wright, 1992; Dussart *et al.*, 1995; Barendsen and Smetsers, 1995a; Barendsen and Smetsers, 1995b; Mossin, 1997].

In the case of strictness properties Jensen's thesis, [Jensen, 1992] (see also Benton's thesis, [Benton, 1992a]), reconciles the abstract interpretation and non-standard type inference approaches by introducing a non-standard type system which is a logical presentation of the abstract domain for strictness properties of [Burn *et al.*, 1986]. In [Hankin and Le Métayer, 1994b] a checking algorithm implementing a variation of Jensen's system is presented.

Dead-code analysis has been mainly studied in the context of *logical frameworks*, like Coq [Barras *et al.*, 1996] (see [Pfenning, 1996] for a short survey on logical frameworks), to remove redundant code from functional programs extracted from formal proofs (see [Paulin-Mohring, 1989b; Paulin-Mohring, 1989a] for an introduction to the subject). In fact, programs extracted from proofs usually contain large parts that are useless for the computation of the final result (i.e. dead-code) and some sort of simplification is mandatory. Various simplification techniques have been developed in the last few years, see for instance [Takayama, 1991; Berardi, 1996; Boerio, 1995; Prost, 1995]. Dead-code can be present also in programs in which general-purpose functions, taken from standard libraries, are used in particular contexts.

## 1.3 About this thesis

The subject of this thesis is the study of non-standard type inference systems for the static analysis of typed functional programs. In particular we propose a general framework for "non-standard type inference based" analyses of typed functional programs and study inference systems (with and without conjunction) for strictness, totality, and dead-code analyses. We develop a semantical investigation of the systems, based on operational semantics, and discuss in some detail possible implementations of the analyses presented. In particular we give sound and complete inference algorithms for strictness and dead-code analyses without conjunction.

The language under analysis is a variant of PCF [Plotkin, 1977], however, the inference approach seems to be suitable for extensions to polymorphic languages and type free languages (see [Coppo and Ferrari, 1993]).

### 1.3.1 A framework for non-standard type inference

Non-standard type inference shares both theoretical and practical tools with standard type theory. The distinction between "types" and "non-standard types" is particular useful when the programming language considered is itself typed, as for the language considered in this thesis (a variant of PCF, basically the simply typed $\lambda$-calculus with cartesian product, if-then-else, fixpoint, and arithmetic constants) and most modern functional programming languages (like ML, Caml, Miranda, Haskell, Clean,... which are equipped with extensions of the Hindley/Milner type system [Hindley, 1969; Milner, 1978; Damas and Milner, 1982]).

The non-standard type inference framework proposed in this thesis (Chapter 3) exploits features common to different static analysis. In particular it provides

1. a core for non-standard type syntax, entailment, and assignment rules, and

2. a foundation for the program analyses which is based directly on the operational semantics of the programming language considered.

### 1.3.2 Outline of the thesis

The thesis is structured in four parts.

Chapter 2 introduces the programming language we are dealing with, its operational semantics, and some term models based on this semantics.

Chapter 3 describes a framework for "non-standard type inference based" analyses of typed functional programs and introduces the strictness, totality, and dead-code analyses considered in this thesis.

In Part II of the thesis we present non-standard type inference systems for simple (in Chapter 4) and conjunctive (in Chapter 5) strictness and totality analyses.

We also consider the problem, first introduced in [Solberg *et al.*, 1994] (see also [Solberg *et al.*, ; Solberg, 1995a; Solberg, 1995b]), of studying the interactions between strictness and totality properties. In particular we show that, even if we spend some effort in defining assignment systems for languages of combined strictness and totality properties (obtaining new systems extending those proposed in *op. cit.*), the weakness of type assignment rules does not allow to use in an essential way the (greater) expressive power of the combined languages of properties.

In Chapter 6 we present an inference algorithm for a strictness analysis presented in Chapter 4.

In Part III of this thesis we study formal systems for detecting and removing dead-code in functional programs. The main application is the optimization of programs extracted from proofs in logical frameworks, but the technique proposed could be used as well in the elimination of dead-code determined by program specialization.

Inference algorithms for the simple dead-code analysis (in Chapter 7) and for a restriction to rank 2 of the conjunctive analysis (in Chapter 9) are presented.

Part IV of the thesis consists of a short chapter discussing related topics and further work.

### 1.3.3 Contributions

A main contribution of this thesis is to propose a framework in which various analyses can be founded by relying directly on operational semantics (see Chapter 3).

As far as strictness and totality analyses, we prove completeness results for various axiomatizations of entailment relations for strictness and totality types.

The results for the analyses without conjunction (in Chapter 4) and for the conjunctive totality analysis are new, while the ones for conjunctive strictness, even though very similar results are already presented in [Jensen, 1992] and [Benton, 1992a], are proved using a different technique.

The results on the "separability" of strictness and totality analyses (see Sections 4.4.5 and 5.4.5) are new.

The simple dead-code analysis (in Chapter 7) is essentially a reformulation in our framework of techniques developed in [Berardi, 1993b] (journal version in [Berardi, 1996]) and [Berardi and Boerio, 1995] (see also [Boerio, 1995] Chapter 4), however the completeness result for the dead-code type entailment relation, and the inference algorithms are new. The conjunctive dead-code analysis (in Chapter 8) is new.

### 1.3.4 Prerequisites

This thesis is mostly self-contained. A basic knowledge of set-theory and logic it is assumed. We also assume some knowledge about functional programming (Chapter 2 is indeed too terse to be considered an introduction to functional programming languages and their operational semantics).

The inference algorithms are written in a functional flavor (the notation is very close to the functional programming language ML).

# Chapter 2

# The language PCFP

In this chapter we introduce a simple functional programming language (basically the simply typed $\lambda$-calculus with cartesian product, if-then-else, fixpoint, and arithmetic constants) and its operational semantics. We use the acronym PCFP for this language, standing for "Programming Computable Function with Pairs", since it is the dialect of the language PCF [Plotkin, 1977] obtained by adding type constructors for pairs (and by using natural numbers instead of integers).

The language PCFP (as described in Section 2.1) is the language considered in Part II of the thesis, while in Part III of the thesis we will consider an extension of PCFP (the language $\mathrm{PCFP}^{\mathcal{T}}$ described in Section 2.6) which includes a program constructor for primitive recursion over natural numbers.

PCFP is a lazy language, however the framework described in Chapter 3 applies also to eager languages. For this reason in Section 2.7 we breafly describe an eager version of PCFP.

For a more comprehensive presentation of the material in Sections 2.1, 2.2 and 2.3 see [Pitts, 1997], where the language considered includes also lazy lists.

## 2.1 PCFP syntax and evaluation rules

The set of PCFP types is defined assuming as *ground* types $\mathsf{nat}$ and $\mathsf{bool}$: the set of natural numbers and the set of booleans. Types are ranged over by $\rho$, $\sigma$, $\tau$.

**Definition 2.1 (PCFP types)** *The language of* types *(***T***) is defined by the following grammar:* $\rho ::= \iota \mid \rho \to \rho \mid \rho \times \rho$*, where* $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$.

PCFP terms are defined from a set of typed *term constants*

$$\begin{aligned}\mathcal{K} \;=\; \{\;& \mathsf{0}^{\mathsf{nat}},\ \mathsf{1}^{\mathsf{nat}},\ \ldots\ \mathsf{succ}^{\mathsf{nat}\to\mathsf{nat}},\ \mathsf{pred}^{\mathsf{nat}\to\mathsf{nat}},\ \ldots \\ & +^{\mathsf{nat}\times\mathsf{nat}\to\mathsf{nat}},\ *^{\mathsf{nat}\times\mathsf{nat}\to\mathsf{nat}},\ \ldots \\ & \mathsf{true}^{\mathsf{bool}},\ \mathsf{false}^{\mathsf{bool}},\ \mathsf{not}^{\mathsf{bool}\to\mathsf{bool}},\ \mathsf{and}^{\mathsf{bool}\times\mathsf{bool}\to\mathsf{bool}},\ \ldots \\ & =^{\mathsf{nat}\times\mathsf{nat}\to\mathsf{bool}},\ <^{\mathsf{nat}\times\mathsf{nat}\to\mathsf{bool}},\ \ldots \qquad\qquad \}\ ,\end{aligned}$$

$$\text{(Var)}\ \vdash x^\rho : \rho \qquad \text{(Con)}\ \vdash \mathsf{c}^\rho : \rho$$

$$(\to \text{I})\ \frac{\vdash M : \sigma}{\vdash \lambda x^\rho.M : \rho \to \sigma} \qquad (\to \text{E})\ \frac{\vdash M : \rho \to \sigma \quad \vdash N : \rho}{\vdash MN : \sigma}$$

$$(\times \text{I})\ \frac{\vdash M_1 : \rho_1 \quad \vdash M_2 : \rho_2}{\vdash \langle M_1, M_2\rangle : \rho_1 \times \rho_2} \qquad (\times \text{E}_i)\ \frac{\vdash M : \rho_1 \times \rho_2}{\vdash \mathsf{proj}_i M : \rho_i}\ i \in \{1,2\}$$

$$(\text{Fix})\ \frac{\vdash M : \rho}{\vdash \mathsf{fix}\, x^\rho.M : \rho} \qquad (\text{If})\ \frac{\vdash N : \mathsf{bool} \quad \vdash M_1 : \rho \quad \vdash M_2 : \rho}{\vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \rho}$$

Figure 2.1: Rules for PCFP term formation (system $\vdash_{\mathbf{T}}$)

(ranged over by $\mathsf{c}$), and a set $\mathcal{V}$ of typed *term variables* (ranged over by $x^\rho, y^\sigma, \ldots$). The type of a constant $\mathsf{c}$ is denoted by $\mathbf{T}(\mathsf{c})$. PCFP terms, ranged over by $M$, $N$, ..., are defined as follows.

**Definition 2.2 (PCFP terms)** *We write $\vdash_{\mathbf{T}} M : \rho$, and say that $M$ is a term of type $\rho$, if $\vdash M : \rho$ is derivable by the rules in Fig. 2.1.*

**Notation 2.3** *1. According to Definition 2.2, in a PCFP term $M$ the types of variables and constants are explicitly mentioned. In the following we often omit to write types which are clear from the context.*

*2. The finite set of the* free variables *of a term $M$, denoted by $\mathrm{FV}(M)$, is defined in the standard way. In the following we take PCFP terms to be $\alpha$-equivalence classes of syntax tree, i.e. we will identify terms modulo renaming of the bound variables.*

*3. We will often use, as syntactic sugar, the infix notation for the binary operators of the language, e.g. we will write "$3 + 5$" instead of "$+\langle 3, 5\rangle$".*

*4. As usual a substitution is a finite function mapping term variables to terms, denoted by $[x_1 := N_1, \ldots, x_n := N_n]$ ($[\vec{x} := \vec{N}]$ for short), which respects the types, i.e. each $x_i^{\rho_i}$ is substituted by a term $N_i$ of the same type. Substitution acts on free variables, the renaming of the bound variables is implicitly supposed. We say that a substitution $[x_1 := N_1, \ldots, x_n := N_n]$ is* closed *if $\mathrm{FV}(N_i) = \emptyset$ $(1 \le i \le n)$.*

Let $\Lambda_{\mathbf{T}}$ be the set of PCFP terms, i.e. $\Lambda_{\mathbf{T}} = \{M \mid \vdash_{\mathbf{T}} M : \rho \text{ for some type } \rho\}$, and $\Lambda^{\mathbf{c}}_{\mathbf{T}}$ be the set of the *closed* terms, i.e. $\Lambda^{\mathbf{c}}_{\mathbf{T}} = \{M \mid M \in \Lambda_{\mathbf{T}} \text{ and } \mathrm{FV}(M) = \emptyset\}$. The values of the terms in $\Lambda^{\mathbf{c}}_{\mathbf{T}}$ are defined via a standard operational semantics (see [Plotkin, 1981a; Kahn, 1988]) described by judgments of the form $M \Downarrow K$, where $M$ is a closed term and $K$ is a closed term in *weak head normal form* (*w.h.n.f.*), i.e. $K \in \mathbf{V}_{\mathbf{T}}$, where

$$\mathbf{V}_{\mathbf{T}} = \mathcal{K} \cup \{\lambda x^\rho.N \mid \lambda x^\rho.N \in \Lambda^{\mathbf{c}}_{\mathbf{T}}\} \cup \{\langle M_1, M_2\rangle \mid \langle M_1, M_2\rangle \in \Lambda^{\mathbf{c}}_{\mathbf{T}}\}.$$

$$(\text{CAN})\ \frac{K \in \mathbf{V_T}}{K \Downarrow K}$$

$$(\text{APP})\ \frac{M \Downarrow \lambda x.P \quad P[x := N] \Downarrow K}{MN \Downarrow K}$$

$$(\text{APP}_1)\ \frac{M \Downarrow \mathsf{c} \quad N \Downarrow \mathsf{c}_1}{MN \Downarrow \mathsf{c}_2}\ (\mathsf{c}_1, \mathsf{c}_2) \in \mathbf{mean}(\mathsf{c})$$

$$(\text{APP}_2)\ \frac{M \Downarrow \mathsf{c} \quad N \Downarrow \langle N_1, N_2\rangle \quad N_1 \Downarrow \mathsf{c}_1 \quad N_2 \Downarrow \mathsf{c}_2}{MN \Downarrow \mathsf{c}_3}\ (\langle \mathsf{c}_1, \mathsf{c}_2\rangle, \mathsf{c}_3) \in \mathbf{mean}(\mathsf{c})$$

$$(\text{PROJ}_i)\ \frac{P \Downarrow \langle M_1, M_2\rangle \quad M_i \Downarrow K}{\mathsf{proj}_i P \Downarrow K}\ i \in \{1, 2\}$$

$$(\text{FIX})\ \frac{M[x := \mathsf{fix}\, x.M] \Downarrow K}{\mathsf{fix}\, x.M \Downarrow K}$$

$$(\text{IF}_1)\ \frac{N \Downarrow \mathsf{true} \quad M_1 \Downarrow K}{\mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 \Downarrow K} \qquad (\text{IF}_2)\ \frac{N \Downarrow \mathsf{false} \quad M_2 \Downarrow K}{\mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 \Downarrow K}$$

Figure 2.2: "Natural semantics" evaluation rules

Assume that any functional constant has a type of the shape $\iota_1 \to \iota_2$ or $\iota_1 \times \iota_2 \to \iota_3$, for some $\iota_1, \iota_2, \iota_3 \in \{\mathsf{nat}, \mathsf{bool}\}$. The meaning of a functional constant $\mathsf{c}$ is given by a set $\mathbf{mean}(C)$ of pairs, i.e. if $(P_1, P_2) \in \mathbf{mean}(C)$ then $CP_1$ evaluates to $P_2$. For example $(5, 6) \in \mathbf{mean}(\mathsf{succ})$ and $(\langle 1, 3\rangle, 4) \in \mathbf{mean}(+)$.

**Definition 2.4 (Value of a term)** *Let $M \in \Lambda^{\mathbf{c}}_{\mathbf{T}}$. We write $M \Downarrow K$, and say that $M$ evaluates to $K$, if this statement is derivable by using the rules in Fig. 2.2.*

Let $M \Downarrow$, to be read "$M$ is convergent", mean that, for some $K$, $M \Downarrow K$, and let $M \Uparrow$, to be read "$M$ is divergent", mean that, for no $K$, $M \Downarrow K$.

## 2.2 Contextual equivalences

Following [Pitts, 1997] we introduce two congruences on PCFP terms. The first is the congruence on terms induced by the contextual preorder that compares the behaviour of terms just at the ground type $\mathsf{nat}$ (*ground contextual equivalence*), while the second is the one induced by the contextual preorder that compares the behavior of terms at every type (*lazy contextual equivalence*).

Let $(\mathcal{C}[\ ]^{\rho})^{\sigma}$ denote a typed context of type $\sigma$ with a hole of type $\rho$ in it.

**Definition 2.5 (Ground contextual equivalence)** *Let $M$ and $N$ be terms of type $\rho$. Define $M \preceq^{\mathrm{gnd}}_{\mathrm{obs}} N$ whenever, for all closed contexts $(\mathcal{C}[\ ]^{\rho})^{\mathsf{nat}}$, if $\mathcal{C}[M]$ and $\mathcal{C}[N]$ are closed terms, then $\mathcal{C}[M] \Downarrow$ implies $\mathcal{C}[N] \Downarrow$. The relation $\preceq^{\mathrm{gnd}}_{\mathrm{obs}}$ is the* ground contextual preorder *and the equivalence induced by $\preceq^{\mathrm{gnd}}_{\mathrm{obs}}$, denoted by $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$, is the* ground observational equivalence.

**Definition 2.6 (Lazy contextual equivalence)** *Let $M$ and $N$ be terms of type $\rho$. Define $M \preceq^{\mathrm{lazy}}_{\mathrm{obs}} N$ whenever, for all type $\sigma$ and for all closed contexts $(\mathcal{C}[\ ]^{\rho})^{\sigma}$, if $\mathcal{C}[M]$ and $\mathcal{C}[N]$ are closed terms, then $\mathcal{C}[M] \Downarrow$ implies $\mathcal{C}[N] \Downarrow$. The relation $\preceq^{\mathrm{lazy}}_{\mathrm{obs}}$ is the* lazy contextual preorder *and the equivalence induced by $\preceq^{\mathrm{lazy}}_{\mathrm{obs}}$, denoted by $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$, is the* lazy observational equivalence.

It is immediate to see that, $M \simeq^{\mathrm{lazy}}_{\mathrm{obs}} N$ implies $M \simeq^{\mathrm{gnd}}_{\mathrm{obs}} N$. The reverse is not true.

## 2.3 Bisimilarity

In [Pitts, 1997] a co-inductive characterization of ground contextual equivalence in terms of bisimulation is given. Some pioneering works importing these techniques (developed in the context of concurrency theory [de Roever, 1978; Park, 1981; Milner, 1989]) into the world of the $\lambda$-calculus and functional programming are [Dybjer and Sander, 1989; Abramsky, 1990b; Howe, 1989; Howe, 1996; Egidi *et al.*, 1992; Gordon, 1994].

We give here the definition of bisimulation for PCFP, and state the main result presented in [Pitts, 1997].

**Definition 2.7** *A* PCFP bisimulation $\mathcal{B}$ *is a type indexed family of relations on closed terms,*

$$\mathcal{B}_{\rho} \subseteq \{\langle M, N\rangle \mid \vdash_{\mathbf{T}} M : \rho, \vdash_{\mathbf{T}} N : \rho \text{ and } \mathrm{FV}(M) \cup \mathrm{FV}(N) = \emptyset\}$$

*($\rho \in \mathbf{T}$), satisfying the conditions in Fig. 2.3.* PCFP bisimilarity *is the largest bisimulation and will be denoted by $\simeq$.*

A PCFP bisimulation $\mathcal{B}$ can be extended to a relation on (possibly) open terms $\mathcal{B}^{\mathbf{o}}$ (called the *open extension* of $\mathcal{B}$). For any PCFP terms $P$ and $Q$ such that $\mathrm{FV}(P) \cup \mathrm{FV}(Q) \subseteq \{x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\}$, define: $P\ \mathcal{B}^{\mathbf{o}}_{\rho}\ Q$ if and only if, for every closed substitution $[x_1^{\sigma_1} := N_1, \ldots, x_n^{\sigma_n} := N_n]$,

$$P[\vec{x} := \vec{N}]\ \mathcal{B}_{\rho}\ Q[\vec{x} := \vec{N}].$$

Let $\simeq^{\mathbf{o}}$ denote the *PCFP open bisimilarity*, i.e. the relation obtained by applying the previous construction to $\simeq$. The following result holds.

**Theorem 2.8 (Operational extensionality for PCFP)** *Ground contextual equivalence coincides with bisimilarity:*

$$M \simeq^{\mathrm{gnd}}_{\mathrm{obs}} N \ \text{ if and only if } \ M \simeq^{\mathbf{o}} N.$$

□

| | |
|---|---|
| (bis 1a) | $(M\ \mathcal{B}_{\mathsf{bool}}\ N$ and $M \Downarrow \mathsf{b})$ implies $N \Downarrow \mathsf{b}$ |
| (bis 1b) | $(M\ \mathcal{B}_{\mathsf{bool}}\ N$ and $N \Downarrow \mathsf{b})$ implies $M \Downarrow \mathsf{b}$ |
| (bis 2a) | $(M\ \mathcal{B}_{\mathsf{nat}}\ N$ and $M \Downarrow \mathsf{n})$ implies $N \Downarrow \mathsf{n}$ |
| (bis 2b) | $(M\ \mathcal{B}_{\mathsf{nat}}\ N$ and $N \Downarrow \mathsf{n})$ implies $M \Downarrow \mathsf{n}$ |
| (bis 3) | $M\ \mathcal{B}_{\rho\to\sigma}\ N$ implies, for all $P$ such that $\vdash_{\mathbf{T}} P : \rho$, $MP\ \mathcal{B}_\sigma\ NP$ |
| (bis 4) | $M\ \mathcal{B}_{\rho\times\sigma}\ N$ implies $(\mathsf{proj}_1(M)\ \mathcal{B}_\rho\ \mathsf{proj}_1(N)$ and $\mathsf{proj}_2(M)\ \mathcal{B}_\sigma\ \mathsf{proj}_2(N))$ |

Figure 2.3: Bisimulation conditions for PCFP

## 2.4 Closed term models of PCFP

In this section we introduce the term models induced by the ground and lazy equivalences. We also introduce richer term models (containing a "convergence to w.h.n.f." test, $\mathsf{isdef}$, and a "parallel or" operator, $\mathsf{por}$). Some of the entailment relations between non-standard types presented in Part II of the thesis are complete w.r.t. these extended models.

### 2.4.1 The models $\mathcal{M}^{\mathrm{gnd}}$ and $\mathcal{M}^{\mathrm{lazy}}$

The *closed term model* $\mathcal{M}^{\mathrm{gnd}}$ of PCFP is defined by interpreting each type $\rho$ as the set of the equivalence classes of the relation $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$ on the closed terms of type $\rho$ in $\Lambda^{\mathbf{c}}_{\mathbf{T}}$. Let $\mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{gnd}}}$ denote the interpretation of type $\rho$ in this model, and let $[M]^{\mathcal{M}^{\mathrm{gnd}}}$ denote the equivalence class of the closed term $M$. For every type $\rho$, $[\mathsf{fix}\, x^\rho.x]^{\mathcal{M}^{\mathrm{gnd}}}$ (the equivalence class of the canonical divergent element, which represents the notion of endless computation), is the least element, w.r.t. $\preceq^{\mathrm{gnd}}_{\mathrm{obs}}$, of $\mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{gnd}}}$. An *environment* is a mapping $e : \mathcal{V} \to \bigcup_{\rho\in\mathbf{T}} \mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{gnd}}}$ which respects types, i.e. a mapping such that, for all $x^\rho$, $e(x^\rho) \in \mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{gnd}}}$. The interpretation of a term $M$ in an environment $e$ is defined in a standard way by:

$$[\![M]\!]^{\mathcal{M}^{\mathrm{gnd}}}_e = [M[x_1 := N_1, \ldots, x_n := N_n]]^{\mathcal{M}^{\mathrm{gnd}}},$$

where $\{x_1, \ldots, x_n\} = \mathrm{FV}(M)$ and $[N_l]^{\mathcal{M}^{\mathrm{gnd}}} = e(x_l)$ $(1 \le l \le n)$.

The *closed term model* $\mathcal{M}^{\mathrm{lazy}}$ of PCFP is defined similarly by interpreting each type $\rho$ as the set of the equivalence classes of the relation $\simeq^{\mathrm{lazy}}_{\mathrm{obs}}$ on the closed terms of type $\rho$ in $\Lambda^{\mathbf{c}}_{\mathbf{T}}$. Note that $\mathcal{M}^{\mathrm{gnd}}$ is an extensional model, while $\mathcal{M}^{\mathrm{lazy}}$ is not.

The $\mathcal{M}^{\mathrm{gnd}}$ semantics identifies terms such as $\mathsf{fix}\, x^{\mathsf{nat}\to\mathsf{nat}}.x$ and $\lambda\, y^{\mathsf{nat}}.\ \mathsf{fix}\, z^{\mathsf{nat}}.z$ (which are extensionally equivalent). This amounts to taking a view in which, as in [Plotkin, 1977] (see also [Benton, 1992a] page. 19), *programs* are closed terms of ground type ($\mathsf{nat}$ or $\mathsf{bool}$) and the only behaviours that a program can exhibit are to diverge or to converge to an integer or boolean value. So the only way in which terms of higher types can be observed is to plug them into a complete program. The $\mathcal{M}^{\mathrm{lazy}}$ semantics, instead, does not identify $\mathsf{fix}\, x^{\mathsf{nat}\to\mathsf{nat}}.x$ and $\lambda\, y^{\mathsf{nat}}.\ \mathsf{fix}\, z^{\mathsf{nat}}.z$. In Section 4.3.3 we will discuss why the lazy model is more suitable for the study of the totality properties.

### 2.4.2 The models $\mathcal{M}^{\mathrm{gnd}}_{\mathsf{por}}$, $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$, and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$.

To define these models of PCFP, we will refer to a set of terms extended by two new program constructors: one for the "parallel or" (por) and one for testing whether a term has a w.h.n.f. (isdef). The resulting set of terms is denoted by $\Lambda_{\mathbf{T}}^{\mathsf{isdef},\mathsf{por}}$. We will also need to consider the set of terms obtained by adding just one of the constructors isdef and por. The corresponding set of terms are called $\Lambda_{\mathbf{T}}^{\mathsf{isdef}}$ and $\Lambda_{\mathbf{T}}^{\mathsf{por}}$.

#### Extended terms

The term formation rules for the new constructors are the following:

$$\text{(Por)}\ \frac{\vdash M_1 : \mathsf{bool} \quad \cdots \quad \vdash M_n : \mathsf{bool}}{\vdash \mathsf{por}(M_1,\ldots,M_n) : \mathsf{bool}}\ n \geq 0 \qquad \text{(Isdef)}\ \frac{\vdash M : \rho}{\vdash \mathsf{isdef}(M) : \mathsf{bool}}\ ,$$

and the evaluation rules are as follows:

$$\text{(POR)}\ \frac{\exists i \in \{1,\ldots,n\}.M_i \Downarrow \mathsf{true}}{\mathsf{por}(M_1,\ldots,M_n) \Downarrow \mathsf{true}} \qquad \text{(ISDEF)}\ \frac{N \Downarrow}{\mathsf{isdef}(N) \Downarrow \mathsf{true}}\ .$$

Note that, according to the previous rules, the parallel or of zero arguments, por(), is divergent, and the parallel or of $n \geq 1$ arguments convergent to false is divergent. It is easy to see that this $n$-ary parallel or operator can be defined by using the more standard binary parallel or operator, $\mathsf{por}_2$, with the following formation and evaluation rules

$$\text{(Por}_2\text{)}\ \frac{\vdash M_1 : \mathsf{bool} \quad \vdash M_2 : \mathsf{bool}}{\vdash \mathsf{por}_2(M_1,M_2) : \mathsf{bool}}$$

$$\text{(POR}_2^{\text{TRUE}}\text{)}\ \frac{\exists i \in \{1,2\}.M_i \Downarrow \mathsf{true}}{\mathsf{por}_2(M_1,M_2) \Downarrow \mathsf{true}} \qquad \text{(POR}_2^{\text{FALSE}}\text{)}\ \frac{M_1 \Downarrow \mathsf{false} \quad M_2 \Downarrow \mathsf{false}}{\mathsf{por}_2(M_1,M_2) \Downarrow \mathsf{false}}\ ,$$

such that the parallel or of 2 arguments convergent to false converges to false.

The constructor isdef is just the extension of the isfn constructor of [Pitts, 1997] page 280 to consider ground types and pairs. Both por and (for non-ground types) isdef are not internally definable in PCFP but, while isdef could be easily added, por requires a completely different evaluation strategy.

#### Extended term models

The *closed term model* $\mathcal{M}^{\mathrm{gnd}}_{\mathsf{por}}$ ($\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$) of PCFP is defined in the obvious way by interpreting each type $\rho$ as the set of the equivalence classes of the relation $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$ (resp. $\simeq^{\mathrm{lazy}}_{\mathrm{obs}}$) on the closed terms of type $\rho$ in $\Lambda_{\mathbf{T}}^{\mathsf{por}}$ (resp. $\Lambda_{\mathbf{T}}^{\mathsf{isdef},\mathsf{por}}$ and $\Lambda_{\mathbf{T}}^{\mathsf{isdef}}$).

## 2.5 The language PCF′

In this section we introduce a restriction of PCFP, called PCF′, which will be used (in Sections 4.4 and 5.4) to study the interaction between strictness and totality properties.

$$\text{(Case)}\ \frac{\vdash_{\mathbf{T}} N : \mathsf{nat} \quad \vdash_{\mathbf{T}} M : \rho \quad \vdash_{\mathbf{T}} F : \mathsf{nat} \to \rho}{\vdash_{\mathbf{T}} \mathsf{case}(N, M, F) : \rho}$$

$$\text{(It)}\ \frac{\vdash_{\mathbf{T}} N : \mathsf{nat} \quad \vdash_{\mathbf{T}} M : \rho \quad \vdash_{\mathbf{T}} F : \rho \to \rho}{\vdash_{\mathbf{T}} \mathsf{it}(N, M, F) : \rho}$$

$$\text{(Rec)}\ \frac{\vdash_{\mathbf{T}} N : \mathsf{nat} \quad \vdash_{\mathbf{T}} M : \rho \quad \vdash_{\mathbf{T}} F : \mathsf{nat} \to \rho \to \rho}{\vdash_{\mathbf{T}} \mathsf{rec}(N, M, F) : \rho}$$

Figure 2.4: Term formation rules for case, it and rec

PCF′ is the language obtained from PCFP by removing pairs and by replacing each binary operator (a constant of type $\iota_1 \times \iota_2 \to \iota_3$, for some $\iota_1, \iota_2, \iota_3 \in \{\mathsf{nat}, \mathsf{bool}\}$) by its curryfied version (a constant of type $\iota_1 \to \iota_2 \to \iota_3$).

As for PCFP, the meaning of a functional constant $\mathsf{c}$ is given by a set $\mathbf{mean}(\mathsf{c})$ of pairs, i.e. if $(\mathsf{c}_1, \mathsf{c}_2) \in \mathbf{mean}(\mathsf{c})$ then $\mathsf{cc}_1$ evaluates to $\mathsf{c}_2$. For example, if $+'^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}$ is the curryfied version of $+^{\mathsf{nat}\times\mathsf{nat}\to\mathsf{nat}}$, then $(5, +_5) \in \mathbf{mean}(+')$ and $(3, 8) \in \mathbf{mean}(+_5)$.

The evaluation rules for PCF′are a subset of the ones for PCFP (in Figure 2.2), since the rules $\mathrm{APP}_2$ and $\mathrm{PROJ}_i$ ($i \in \{1, 2\}$) are not needed.

All the results in the previous sections apply to PCF′ as well.

## 2.6 PCFP with primitive recursion

In this section we introduce an extension of PCFP, called $\mathrm{PCFP}^{\mathcal{T}}$, which is more suitable for expressing programs extracted from formal proofs. The language $\mathrm{PCFP}^{\mathcal{T}}$ is obtained from PCFP by adding a program constructor for primitive recursion over natural numbers (rec). There are also two constructors for specifying simplified uses of primitive recursion: the constructor it (for iteration) and the constructor case. $\mathrm{PCFP}^{\mathcal{T}}$, which is a variant of Gödel's system $\mathcal{T}$, is the language considered in Part III of this thesis.

The term formation rules for the new constructors are in Fig. 2.4, and the evaluation rules are in Fig. 2.5.

Since for every $\mathrm{PCFP}^{\mathcal{T}}$ term there is an equivalent (w.r.t. the interpretation in all the closed term model described in Section 2.4) PCFP term (see Fact 2.10 below), we have that all the results for PCFP presented in the previous sections apply to $\mathrm{PCFP}^{\mathcal{T}}$ as well.

**Definition 2.9** *For every $\mathrm{PCFP}^{\mathcal{T}}$ term $P$, let $(P)^\star$ be the PCFP term inductively defined*

$$(\mathrm{CASE}_1)\ \frac{N \Downarrow \mathsf{0} \quad M \Downarrow K}{\mathsf{case}(N,M,F) \Downarrow K} \qquad (\mathrm{CASE}_2)\ \frac{N \Downarrow \mathsf{n} \quad F\,\mathsf{n} \Downarrow K}{\mathsf{case}(N,M,F) \Downarrow K}\ \mathsf{n} \neq \mathsf{0}$$

$$(\mathrm{IT}_1)\ \frac{N \Downarrow \mathsf{0} \quad M \Downarrow K}{\mathsf{it}(N,M,F) \Downarrow K} \qquad (\mathrm{IT}_2)\ \frac{N \Downarrow \mathsf{n} \quad F(\mathsf{it}(\mathsf{pred}\,\mathsf{n},M,F)) \Downarrow K}{\mathsf{it}(N,M,F) \Downarrow K}\ \mathsf{n} \neq \mathsf{0}$$

$$(\mathrm{REC}_1)\ \frac{N \Downarrow \mathsf{0} \quad M \Downarrow K}{\mathsf{rec}(N,M,F) \Downarrow K} \qquad (\mathrm{REC}_2)\ \frac{N \Downarrow \mathsf{n} \quad F\,\mathsf{n}\,(\mathsf{rec}(\mathsf{pred}\,\mathsf{n},M,F)) \Downarrow K}{\mathsf{rec}(N,M,F) \Downarrow K}\ \mathsf{n} \neq \mathsf{0}$$

Figure 2.5: Evaluation rules for case, it and rec

*by:*

$$\begin{aligned}
(P)^\star &= P, \text{ if } P \in \mathcal{K} \cup \mathcal{V} \\
(\lambda x^\rho.M)^\star &= \lambda x^\rho.(M)^\star \\
(MN)^\star &= (M)^\star(N)^\star \\
(\mathsf{case}(N,M,F))^\star &= (\lambda n.\mathsf{if}\ n = 0\ \mathsf{then}\ (M)^\star\ \mathsf{else}\ (F)^\star n)\ (N)^\star \\
(\mathsf{it}(N,M,F))^\star &= (\mathsf{fix}\, g.\lambda n.\mathsf{if}\ n = 0\ \mathsf{then}\ (M)^\star\ \mathsf{else}\ (F)^\star(g(\mathsf{pred}n)(M)^\star(F)^\star))\ (N)^\star \\
(\mathsf{rec}(N,M,F))^\star &= (\mathsf{fix}\, g.\lambda n.\mathsf{if}\ n = 0\ \mathsf{then}\ (M)^\star\ \mathsf{else}\ (F)^\star n(g(\mathsf{pred}n)(M)^\star(F)^\star))\ (N)^\star.
\end{aligned}$$

**Fact 2.10** *Let* $\mathcal{M} \in \{\mathcal{M}^{\mathrm{gnd}}, \mathcal{M}^{\mathrm{lazy}}, \mathcal{M}^{\mathrm{gnd}}_{\mathsf{por}}, \mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}, \mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}\}$. *For every PCFP*$^{\mathcal{T}}$ *term* $P$, $[P]^{\mathcal{M}} = [(P)^\star]^{\mathcal{M}}$. □

## 2.7 Eager PCFP

PCFP is a lazy language, however the framework in Section 3.1 applies also to eager languages. In particular, as we will explain in Sections 7.8 and 7.9, the program improvements introduced by the dead-code analyses presented in Part III of the thesis are greater when an eager language is considered.

An eager version of PCFP (and of PCFP$^{\mathcal{T}}$) can be obtained by replacing the call-by-name application rule APP in Fig. 2.2 by the following call-by-value application rule

$$(\mathrm{APP}^{\mathrm{E}})\ \frac{M \Downarrow \lambda x.P \quad N \Downarrow H \quad P[x := H] \Downarrow K}{MN \Downarrow K}\ .$$

When considering eager application it is natural to consider eager pairs. This can be done by

1. considering a set of values, $\mathbf{V}^{\mathrm{E}}_{\mathbf{T}}$, inductively defined by
   - $K \in \mathbf{V}^{\mathrm{E}}_{\mathbf{T}}$, if $K \in \mathcal{K}$,
   - $\lambda x^\rho.N \in \mathbf{V}^{\mathrm{E}}_{\mathbf{T}}$ if $\lambda x^\rho.N \in \Lambda^{\mathbf{c}}_{\mathbf{T}}$, and
   - $\langle K_1, K_2\rangle \in \mathbf{V}^{\mathrm{E}}_{\mathbf{T}}$ if $K_1, K_2 \in \mathbf{V}^{\mathrm{E}}_{\mathbf{T}}$.

2. Replacing rules CAN, $\mathrm{APP}_2$ and $\mathrm{PROJ}_i$ in Fig. 2.2 by the following rules

$$(\mathrm{CAN}^{\mathrm{E}})\ \frac{K \in \mathbf{V}_{\mathbf{T}}^{\mathrm{E}}}{K \Downarrow K}$$

$$(\mathrm{APP}_2{}^{\mathrm{E}})\ \frac{M \Downarrow \mathsf{c} \quad N \Downarrow \langle \mathsf{c}_1, \mathsf{c}_2 \rangle}{MN \Downarrow \mathsf{c}_3}\ (\langle \mathsf{c}_1, \mathsf{c}_2 \rangle, \mathsf{c}_3) \in \mathbf{mean}(\mathsf{c})$$

$$(\mathrm{PROJ}_i{}^{\mathrm{E}})\ \frac{P \Downarrow \langle K_1, K_2 \rangle}{\mathsf{proj}_i P \Downarrow K_i}\ i \in \{1, 2\}.$$

3. Adding the following rule for the eager evaluation of pairs

$$(\mathrm{PAIR}^{\mathrm{E}})\ \frac{M_1 \Downarrow K_1 \quad M_2 \Downarrow K_2}{\langle M_1, M_2 \rangle \Downarrow \langle K_1, K_2 \rangle}\ i \in \{1, 2\}.$$

Of course the properties of both $\simeq_{\mathrm{obs}}^{\mathrm{gnd}}$ and $\simeq_{\mathrm{obs}}^{\mathrm{lazy}}$ (and of the induced models) change (see [Egidi *et al.*, 1992]).

# Chapter 3

# A framework for non-standard type inference

In this chapter we sketch a framework for "non-standard type inference based" analyses of typed functional programs.

The presentation is tailored to PCFP and its variants (see Chapter 2), and aims to introduce the analyses that will be carried on in the rest of the thesis.

## 3.1 The framework

This framework exploits common features of the various static analyses considered in the thesis. In particular it provides

1. a core for non-standard type syntax, entailment, and assignment rules, and
2. a foundation for the program analyses which is based directly on the operational semantics of the programming language considered.

As we will remark in Section 3.5, instantiating the framework to a particular analysis is not in general a straightforward task.

### 3.1.1 Non-standard types

For every PCFP type $\rho \in \mathbf{T}$ we consider a finite non-empty set $\mathbf{L}(\rho)$ of *non-standard types* expressing properties of terms of type $\rho$.

#### Syntax

For every type $\rho \in \mathbf{T}$ let $\mathbf{B}(\rho)$ be a finite set of *basic non-standard types* (or *basic properties*) for terms of type $\rho$. The sets $\mathbf{B}(\rho)$ depend on the particular analysis considered, however, in any case we require that for every PCFP ground type $\iota$ (i.e. $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$) $\mathbf{B}(\iota) \neq \emptyset$.

$$(\mathbf{B})\ \frac{\phi \in \mathbf{B}(\rho)}{\phi \in \mathbf{L}(\rho)}$$

$$(\rightarrow)\ \frac{\phi \in \mathbf{L}(\rho) \quad \psi \in \mathbf{L}(\sigma)}{\phi \rightarrow \psi \in \mathbf{L}(\rho \rightarrow \sigma)} \qquad (\times)\ \frac{\phi_1 \in \mathbf{L}(\rho_1) \quad \phi_2 \in \mathbf{L}(\rho_2)}{\phi_1 \times \phi_2 \in \mathbf{L}(\rho_1 \times \rho_2)}$$

Figure 3.1: "Core" non-standard types formation rules

The set of non-standard types, $\mathbf{L} = \cup_{\rho \in \mathbf{T}} \mathbf{L}(\rho)$, is defined from the set of basic properties, $\mathbf{B} = \cup_{\rho \in \mathbf{T}} \mathbf{B}(\rho)$, according to formation rules including

- rules for the basic properties and for the standard type constructors (in Fig. 3.1), and
- rules for some additional (possibly none) constructors (see Section 3.3).

**Notation 3.1** *For every non-standard type $\phi \in \mathbf{L}(\rho)$, $\rho$ is called the* underlying type *of $\phi$ and it will be denoted by $\epsilon(\phi)$.*

### Semantics

For all the analysis considered in this thesis the semantics of non-standard types satisfies the following 3 conditions.

1. Non-standard types are interpreted as *partial equivalence relations*[1] (*p.e.r.s*) over a model $\mathcal{M}$ of the functional language considered. In particular, non-standard types in $\mathbf{L}(\rho)$, are interpreted as p.e.r.s over the interpretation of the underlying type, $\mathbf{I}(\rho)^{\mathcal{M}}$.
2. $\mathcal{M}$ is a term model of some operationally defined program equivalence (like $\mathcal{M}^{\mathrm{gnd}}$, $\mathcal{M}^{\mathrm{lazy}}$, ... in Section 2.4).
3. These p.e.r.s are inductively defined according to a set of clauses including the ones in Fig. 3.2 (which specify the semantics of the basic properties and of the standard type constructors "arrow" and "times") starting from the interpretation of basic properties, $\mathbf{P}(\phi)^{\mathcal{M}}$ (which, for every $\phi \in \mathbf{B}$, is a p.e.r. over $\mathbf{I}(\epsilon(\phi))^{\mathcal{M}}$).

This means that the analyses are founded relying directly on the operational semantics of the functional language considered.

It is easy to check that, whenever all the basic properties are interpreted as p.e.r.s consisting of exactly one equivalence class, we have that every non-standard type is interpreted as a p.e.r. consisting of exactly one equivalence class. In this case we can replace each p.e.r. by the corresponding equivalence class and switch to the (simpler) set semantics defined according to the clauses in Fig. 3.3, where, for every basic property $\phi \in \mathbf{B}(\rho)$,

$$\mathbf{S}(\phi)^{\mathcal{M}} = \{[M] \mid \langle [M], [M] \rangle \in \mathbf{P}(\phi)^{\mathcal{M}}\} \subseteq \mathbf{I}(\rho)^{\mathcal{M}}$$

is the semantics of $\phi$.

[1] A partial equivalence relation over a set $D$ is a binary relation which is symmetric and transitive.

$$
\begin{array}{lcl}
[\![\phi]\!]^{\mathcal{M}} & = & \mathbf{P}(\phi)^{\mathcal{M}}, \text{ if } \phi \in \mathbf{B} \\
[\![\phi \to \psi]\!]^{\mathcal{M}} & = & \{\langle [M], [N] \rangle \mid \forall \langle [P], [Q] \rangle \in [\![\phi]\!]^{\mathcal{M}}. \langle [MP], [NQ] \rangle \in [\![\psi]\!]^{\mathcal{M}}\} \\
[\![\phi_1 \times \phi_2]\!]^{\mathcal{M}} & = & \{\langle [M], [N] \rangle \mid \forall i \in \{1,2\}.\ \langle [\mathsf{proj}_i M], [\mathsf{proj}_i N] \rangle \in [\![\phi_i]\!]^{\mathcal{M}}\}.
\end{array}
$$

Figure 3.2: P.e.r. semantics of "core" non-standard types

$$
\begin{array}{lcl}
[\![\phi]\!]^{\mathcal{M}} & = & \mathbf{S}(\phi)^{\mathcal{M}}, \text{ if } \phi \in \mathbf{B} \\
[\![\phi \to \psi]\!]^{\mathcal{M}} & = & \{[M] \mid \forall [P] \in [\![\phi]\!]^{\mathcal{M}}. [MP] \in [\![\psi]\!]^{\mathcal{M}}\} \\
[\![\phi_1 \times \phi_2]\!]^{\mathcal{M}} & = & \{[M] \mid \forall i \in \{1,2\}.\ [\mathsf{proj}_i M] \in [\![\phi_i]\!]^{\mathcal{M}}\}.
\end{array}
$$

Figure 3.3: Set semantics of "core" non-standard types

### 3.1.2 Entailment rules

The entailment relation between non-standard types (which represents a logical implication between properties) models the set-theoretic inclusion between their interpretations.

Let $\phi, \psi \in \mathbf{L}$. We write $\phi \leq \psi$ to mean that this judgement is derivable by a set of entailment rules including

- reflexivity, transitivity, the rules for the standard type constructors (in Fig. 3.4), and
- rules reflecting the meaning of basic properties and of the additional (if any) non-standard type constructors (see Section 3.3).

By $\cong$ we denote the equivalence relation induced by $\leq$.

As usual we say that the non-standard type entailment relation $\leq$ is

- *sound* w.r.t. the interpretation $[\![\cdot]\!]^{\mathcal{M}}$ if, for every $\phi, \psi \in \mathbf{L}$,

$$\phi \leq \psi \text{ implies } [\![\phi]\!]^{\mathcal{M}} \subseteq [\![\psi]\!]^{\mathcal{M}},$$

$$
\begin{array}{ll}
\text{(Ref) } \phi \leq \phi & \text{(Trans) } \dfrac{\phi_1 \leq \phi_2 \quad \phi_2 \leq \phi_3}{\phi_1 \leq \phi_3} \\[2ex]
(\to)\ \dfrac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_2 \to \psi_1 \leq \phi_1 \to \psi_2} & (\times)\ \dfrac{\phi_1 \leq \psi_1 \quad \phi_2 \leq \psi_2}{\phi_1 \times \phi_2 \leq \psi_1 \times \psi_2}
\end{array}
$$

Figure 3.4: "Core" entailment rules for non-standard types

- *complete* w.r.t. the interpretation $[\![\cdot]\!]^{\mathcal{M}}$ if, for every $\phi, \psi \in \mathbf{L}$,

$$[\![\phi]\!]^{\mathcal{M}} \subseteq [\![\psi]\!]^{\mathcal{M}} \text{ implies } \phi \leq \psi.$$

Note that soundness implies the following syntactic condition,

$$\text{for all } \phi_1, \phi_2 \in \mathbf{L},\ \phi_1 \leq \phi_2 \text{ implies } \epsilon(\phi_1) = \epsilon(\phi_2).$$

### 3.1.3 Assignment rules

Non-standard types are assigned to terms by a set of inference rules.

If $x^\rho$ is a term variable of type $\rho$ an assumption for $x^\rho$ is an expression of the shape $x^\rho : \phi$ , or $x : \phi$ for short, where $\epsilon(\phi) = \rho$. A basis is a set $\Sigma$ of non-standard type assumptions for term variables. $\Sigma, x : \psi$ denotes the basis $\Sigma \cup \{x : \psi\}$ where it is assumed that $x$ does not appear in $\Sigma$. The notation $\epsilon(\cdot)$ (see Notation 3.1) is extended to basis. More precisely: $\epsilon(\Sigma)$ is the set of term variables which occur in $\Sigma$.

For each constant $\mathsf{c}$ a finite non-empty set $\mathbf{L}(\mathsf{c})$ of minimal non-standard types for $\mathsf{c}$ is specified. Of course, for each $\phi \in \mathbf{L}(\mathsf{c})$, $\epsilon(\phi) = \mathbf{T}(\mathsf{c})$. The idea is that every non-standard type which is entailed ($\leq$) by an element of $\mathbf{L}(\mathsf{c})$ can be assigned to $\mathsf{c}$.

A non-standard typing statement is an expression $\Sigma \vdash M : \phi$ where $M$ is a term of type $\epsilon(\phi)$ and $\Sigma$ is a basis containing an assumption for each free variable of $M$. We write $\Sigma \vdash M : \phi$ to mean that this judgement can be derived by a set of assignment rules including

- a rule allowing the use of the non-standard type entailment relation, rules for the basic program constructs variable, constant, $\lambda$-abstraction, application, pair and projection (in Fig. 3.5), and
- other rules including at least rules for the remaining program constructors ($\mathsf{if}$ and $\mathsf{fix}$).

We say that

- two environments $e_1$, $e_2$ are $\Sigma$-related if and only if, for all $x : \psi \in \Sigma$, $\langle e_1(x), e_2(x)\rangle \in [\![\psi]\!]^{\mathcal{M}}$, and
- the non-standard type assignment relation $\vdash$ is *sound* w.r.t. the interpretation $[\![\cdot]\!]^{\mathcal{M}}$ if, for all $e_1$, $e_2$, if $e_1$ and $e_2$ are $\Sigma$-related, then

$$\Sigma \vdash M : \phi \text{ implies } \langle [\![M]\!]^{\mathcal{M}}_{e_1}, [\![M]\!]^{\mathcal{M}}_{e_2}\rangle \in [\![\phi]\!]^{\mathcal{M}}.$$

For the simpler set semantics described at the end of Section 3.1.1, we have that the assignment system $\vdash$ is sound if, for every environment $e$ that respects $\Sigma$ (i.e. one such that, for every assumption $x : \psi \in \Sigma$, $e(x) \in [\![\psi]\!]^{\mathcal{M}}$),

$$\Sigma \vdash M : \phi \text{ implies } [\![M]\!]^{\mathcal{M}}_e \in [\![\phi]\!]^{\mathcal{M}}.$$

$$(\leq)\ \frac{\Sigma \vdash M : \phi \quad \phi \leq \psi}{\Sigma \vdash M : \psi}$$

(Var) $\Sigma, x : \phi \vdash x : \phi$ (Con) $\frac{\phi \in \mathbf{L}(\mathsf{c})}{\Sigma \vdash \mathsf{c} : \phi}$

$(\to \mathrm{I})\ \frac{\Sigma, x : \phi \vdash M : \psi}{\Sigma \vdash \lambda x.M : \phi \to \psi}$ $(\to \mathrm{E})\ \frac{\Sigma \vdash M : \phi \to \psi \quad \Sigma \vdash N : \phi}{\Sigma \vdash MN : \psi}$

$(\times \mathrm{I})\ \frac{\Sigma \vdash M_1 : \phi_1 \quad \Sigma \vdash M_2 : \phi_2}{\Sigma \vdash \langle M_1, M_2 \rangle : \phi_1 \times \phi_2}$ $(\times \mathrm{E}_i)\ \frac{\Sigma \vdash M : \phi_1 \times \phi_2}{\Sigma \vdash \mathsf{proj}_i M : \phi_i}\ i \in \{1, 2\}$

Figure 3.5: "Core" non-standard type assignment rules

## 3.2 Instantiating the framework

To instantiate the framework described in Section 3.1 to a particular analysis we have to provide the following.

1. A term model, $\mathcal{M}$, of some operationally defined program equivalence (like $\mathcal{M}^{\mathrm{gnd}}$, $\mathcal{M}^{\mathrm{lazy}}$, ... in Section 2.4).

2. For every PCFP type $\rho \in \mathbf{T}$, a set of basic properties, $\mathbf{B}(\rho)$, and some (possibly none) additional non-standard type constructors (see Section 3.3) with the corresponding non-standard type formation rules (to be added to the rules in Fig. 3.1).

3. The semantics of the basic properties (i.e. a function $\mathbf{P}(\cdot)^{\mathcal{M}}$ mapping the basic properties in $\mathbf{B}(\rho)$ to p.e.r.s over $\mathbf{I}(\rho)^{\mathcal{M}}$) and of the additional (if any) non-standard type constructors.

4. The entailment rules for the basic properties and for the additional (if any) non-standard type constructors (to be added to the rules in Fig. 3.4), together with a proof that the resulting entailment relation is sound w.r.t. the given semantics.

5. For each PCFP constant $\mathsf{c}$, a finite non-empty set, $\mathbf{L}(\mathsf{c})$, of minimal non-standard types for $\mathsf{c}$. Of course, for each $\phi \in \mathbf{L}(\mathsf{c})$, $\epsilon(\phi) = \mathbf{T}(\mathsf{c})$. The idea is that every non-standard type which is entailed ($\leq$) by an element of $\mathbf{L}(\mathsf{c})$ can be assigned to $\mathsf{c}$.

6. Some (possibly none) additional assignment rules (to be added to the rules in Fig. 3.5) and a proof that the resulting non-standard type assignment is sound w.r.t. the given semantics.

We now show how the framework can be instantiated to the simple strictness, totality, and dead-code analyses of Chapters 4 and 7.

### 3.2.1 Simple strictness analysis for PCFP

We want to consider a strictness analysis that fits lazy contextual equivalence, i.e. such that whenever a PCFP term $F$ is proved to be strict (according to the analysis) then, for every possible argument $M$, the (value obtained from the) evaluation of $FM$ by making the application of $F$ to $M$ with a call-by-need is lazy observational equivalent to the (value obtained from the) evaluation of $FM$ by making the application of $F$ to $M$ with a call-by-value.

To meet this requirement we take $\mathcal{M} = \mathcal{M}^{\mathrm{lazy}}$ (the closed PCFP term model induced by lazy observation equivalence, see Section 2.4.1).

#### Strictness types (syntax)

For every $\rho \in \mathbf{T}$, the set of the basic properties is $\mathbf{B}^{\mathrm{s}}(\rho) = \{\bot^\rho, \top^\rho\}$. The corresponding set of non-standard types, called *strictness types* (*s-types* for short), is be denoted by $\mathbf{L}^{\mathrm{s}}$.

#### Strictness types (semantics)

For every $\rho \in \mathbf{T}$, let

- $\mathbf{P}(\bot^\rho)^{\mathcal{M}^{\mathrm{lazy}}} = \{\langle [\mathsf{fix}\, x^\rho.x]^{\mathcal{M}^{\mathrm{lazy}}}, [\mathsf{fix}\, x^\rho.x]^{\mathcal{M}^{\mathrm{lazy}}}\rangle\}$, and
- $\mathbf{P}(\top^\rho)^{\mathcal{M}^{\mathrm{lazy}}} = \mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}} \times \mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}}$.

This semantics says that, for every $\rho \in \mathbf{T}$,

- $\bot^\rho$ (which is interpreted as the p.e.r. consisting of one class whose unique element is the $\simeq^{\mathrm{lazy}}_{\mathrm{obs}}$-class of the PCFP terms of type $\rho$ without w.h.n.f.) is the property of being a divergent term of type $\rho$, characterizing the PCFP terms of type $\rho$ without w.h.n.f. (lazy observational equivalent to $\mathsf{fix}\, x^\rho.x$), while
- $\top^\rho$ (which is interpreted as the trivial p.e.r. over $\mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}}$, consisting of one class equating all the elements of $\mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}}$) is the trivial property, characterizing all the PCFP terms of type $\rho$.

Since every basic property is interpreted as a p.e.r. consisting of exactly one class, we can switch to the set semantics described at the end of Section 3.1.1 and define, for every $\rho \in \mathbf{T}$,

- $\mathbf{S}(\bot^\rho)^{\mathcal{M}^{\mathrm{lazy}}} = \{[\mathsf{fix}\, x^\rho.x]^{\mathcal{M}^{\mathrm{lazy}}}\}$, and
- $\mathbf{S}(\top^\rho)^{\mathcal{M}^{\mathrm{lazy}}} = \mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}}$.

**Entailment rules**

Let $\mathbf{L}^{\mathrm{s}}_{\top}$ is the subset of the s-types corresponding to the trivial properties, i.e. such that $[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}} = \mathbf{I}(\epsilon(\phi))^{\mathcal{M}^{\mathrm{lazy}}}$, which (as we will see in Chapter 4) is inductively defined by the following clauses:

- $\top^{\rho} \in \mathbf{L}^{\mathrm{s}}_{\top}$, for $\rho \in \mathbf{T}$
- $\phi \to \psi \in \mathbf{L}^{\mathrm{s}}_{\top}$, for $\phi \in \mathbf{L}^{\mathrm{s}}$ and $\psi \in \mathbf{L}^{\mathrm{s}}_{\top}$
- $\phi_1 \times \phi_2 \in \mathbf{L}^{\mathrm{s}}_{\top}$, for $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{s}}_{\top}$.

We introduce the following rules capturing the semantics of the basic properties:

$$(\bot)\ \bot^{\epsilon(\phi)} \leq \phi \qquad\qquad (\top)\ \frac{\psi \in \mathbf{L}^{\mathrm{s}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}\ .$$

**Assignment rules**

In Section 4.2.2 we will introduce rules for the program constructors $\mathsf{if}$ and $\mathsf{fix}$. Here we just look at a simple example.

Take the term

$$\mathrm{app} = \lambda f^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.fx,$$

of type $\rho = (\mathsf{nat} \to \mathsf{nat}) \to \mathsf{nat} \to \mathsf{nat}$. By using the assignment rules in Fig. 3.5 it is possible to assign to app both the s-types

- $\phi_1 = (\bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$ (strict function to strict function), and
- $\phi_2 = (\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$ (function divergent on every argument to function divergent on every argument),

as the following derivations show.

1. Let $\Sigma_1 = \{f : \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}, x : \bot^{\mathsf{nat}}\}$, and $\Sigma_1' = \{f : \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}\}$.

$$(\to \mathrm{E})\ \frac{(\mathrm{Var})\ \Sigma_1 \vdash f : \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \qquad (\mathrm{Var})\ \Sigma_1 \vdash x : \bot^{\mathsf{nat}}}{(\to \mathrm{I})\ \dfrac{\Sigma_1 \vdash fx : \bot^{\mathsf{nat}}}{(\to \mathrm{I})\ \dfrac{\Sigma_1' \vdash \lambda x^{\mathsf{nat}}.fx : \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}}{\emptyset \vdash \mathrm{app} : \phi_1}}}$$

2. Let $\Sigma_2 = \{f : \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}, x : \top^{\mathsf{nat}}\}$, and $\Sigma_2' = \{f : \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}\}$.

$$(\to \mathrm{E})\ \frac{(\mathrm{Var})\ \Sigma_2 \vdash f : \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \qquad (\mathrm{Var})\ \Sigma_2 \vdash x : \top^{\mathsf{nat}}}{(\to \mathrm{I})\ \dfrac{\Sigma_2 \vdash fx : \bot^{\mathsf{nat}}}{(\to \mathrm{I})\ \dfrac{\Sigma_2' \vdash \lambda x^{\mathsf{nat}}.fx : \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}}{\emptyset \vdash \mathrm{app} : \phi_2}}}$$

### 3.2.2 Simple totality analysis for PCFP

As for the simple strictness analysis of Section 3.2.1, take $\mathcal{M} = \mathcal{M}^{\mathrm{lazy}}$.

#### Totality types (syntax)

For every $\rho \in \mathbf{T}$, the set of the basic properties is $\mathbf{B}^{\mathrm{t}}(\rho) = \{\not\perp^{\rho}, \top^{\rho}\}$. The corresponding set of non-standard types, called *totality types* (*t-types* for short), is denoted by $\mathbf{L}^{\mathrm{t}}$.

#### Totality types (semantics)

For every $\rho \in \mathbf{T}$, let

- $\mathbf{P}(\not\perp^{\rho})^{\mathcal{M}^{\mathrm{lazy}}} = (\mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}} - [\mathsf{fix}\, x^{\rho}.x]^{\mathcal{M}^{\mathrm{lazy}}}) \times (\mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}} - [\mathsf{fix}\, x^{\rho}.x]^{\mathcal{M}^{\mathrm{lazy}}})$, and
- $\mathbf{P}(\top^{\rho})^{\mathcal{M}^{\mathrm{lazy}}} = \mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}} \times \mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}}$.

This semantics says that, for every $\rho \in \mathbf{T}$,

- $\not\perp^{\rho}$ (which is interpreted as the p.e.r. consisting of one class equating all the $\simeq^{\mathrm{lazy}}_{\mathrm{obs}}$-classes of the PCFP terms of type $\rho$ in w.h.n.f.) is the property of being a convergent term of type $\rho$, characterizing the PCFP terms of type $\rho$ with w.h.n.f. (not lazy observational equivalent to $\mathsf{fix}\, x^{\rho}.x$),
- while, as for strictness, $\top^{\rho}$ (which is interpreted as the trivial p.e.r. over $\mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}}$, consisting of one class equating all the elements of $\mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}}$) is the trivial property, characterizing all the PCFP terms of type $\rho$.

Since, as for strictness, every basic property is interpreted as a p.e.r. consisting of exactly one class, we can switch to the set semantics described at the and of Section 3.1.1 and define, for every $\rho \in \mathbf{T}$,

- $\mathbf{S}(\not\perp^{\rho})^{\mathcal{M}^{\mathrm{lazy}}} = \mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}} - \{[\mathsf{fix}\, x^{\rho}.x]^{\mathcal{M}^{\mathrm{lazy}}}\}$, and
- $\mathbf{S}(\top^{\rho})^{\mathcal{M}} = \mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}}$.

#### Entailment rules

Let $\mathbf{L}^{\mathrm{t}}_{\top}$ is the subset of the t-types corresponding to the trivial properties, i.e. such that $[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}} = \mathbf{I}(\epsilon(\phi))^{\mathcal{M}^{\mathrm{lazy}}}$, which (as we will see in Chapter 4) is inductively defined by the following clauses:

- $\top^{\rho} \in \mathbf{L}^{\mathrm{t}}_{\top}$, for $\rho \in \mathbf{T}$
- $\phi \to \psi \in \mathbf{L}^{\mathrm{t}}_{\top}$, for $\phi \in \mathbf{L}^{\mathrm{t}}$ and $\psi \in \mathbf{L}^{\mathrm{t}}_{\top}$
- $\phi_1 \times \phi_2 \in \mathbf{L}^{\mathrm{t}}_{\top}$, for $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{t}}_{\top}$.

We introduce the following rules capturing the semantics of the basic properties:

$$(\not\perp)\ \frac{\phi \in \mathbf{L}^{\mathrm{t}} - \mathbf{L}^{\mathrm{t}}_{\top}}{\phi \leq \not\perp^{\epsilon(\phi)}} \qquad (\top)\ \frac{\psi \in \mathbf{L}^{\mathrm{t}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}\ .$$

**Assignment rules**

In Section 4.3.2 we will introduce an additional rule for $\lambda$-abstraction and rules for the program constructors $\mathsf{if}$ and $\mathsf{fix}$. Here we just look at a simple example.

Take the term

$$\mathrm{app} = \lambda f^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.fx,$$

of type $\rho = (\mathsf{nat} \to \mathsf{nat}) \to \mathsf{nat} \to \mathsf{nat}$. By using the assignment rules in Fig. 3.5 it is possible to assign to app both the t-types

- $\phi_1 = (\not\downarrow^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}) \to \not\downarrow^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}$ (total function to total function), and
- $\phi_2 = (\top^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}$ (function convergent on every argument to function convergent on every argument),

as the following derivations show.

1. Let $\Sigma_1 = \{f : \not\downarrow^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}, x : \not\downarrow^{\mathsf{nat}}\}$, and $\Sigma_1' = \{f : \not\downarrow^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}\}$.

$$(\to \mathrm{E})\ \frac{(\mathrm{Var})\ \Sigma_1 \vdash f : \not\downarrow^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}} \qquad (\mathrm{Var})\ \Sigma_1 \vdash x : \not\downarrow^{\mathsf{nat}}}{(\to \mathrm{I})\ \dfrac{\Sigma_1 \vdash fx : \not\downarrow^{\mathsf{nat}}}{(\to \mathrm{I})\ \dfrac{\Sigma_1' \vdash \lambda x^{\mathsf{nat}}.fx : \not\downarrow^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}}{\emptyset \vdash \mathrm{app} : \phi_1}}}$$

2. Let $\Sigma_2 = \{f : \top^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}, x : \top^{\mathsf{nat}}\}$, and $\Sigma_2' = \{f : \top^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}\}$.

$$(\to \mathrm{E})\ \frac{(\mathrm{Var})\ \Sigma_2 \vdash f : \top^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}} \qquad (\mathrm{Var})\ \Sigma_2 \vdash x : \not\downarrow^{\mathsf{nat}}}{(\to \mathrm{I})\ \dfrac{\Sigma_2 \vdash fx : \not\downarrow^{\mathsf{nat}}}{(\to \mathrm{I})\ \dfrac{\Sigma_2' \vdash \lambda x^{\mathsf{nat}}.fx : \top^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}}{\emptyset \vdash \mathrm{app} : \phi_2}}}$$

### 3.2.3 Simple dead-code analysis for PCFP$^{\mathcal{T}}$

We want to consider a dead-code analysis that fits ground contextual equivalence, i.e. such that, if a subterm $N$ of a PCFP$^{\mathcal{T}}$ term $M$ is proved to be useless (according to the analysis), then $M$ is ground observational equivalent to the term obtained from $M$ by replacing the useless subterm $N$ with any term $N'$ of the same type.

To meet this requirement we take $\mathcal{M} = \mathcal{M}^{\mathrm{gnd}}$ (the closed PCFP$^{\mathcal{T}}$ term model induced by ground observational equivalence, see Section 2.4.1).

**Dead-code types (syntax)**

For every ground type $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$, the set of the basic properties is $\mathbf{B}^{\mathrm{d}}(\iota) = \{\delta^\iota, \top^\iota\}$, while, for every non-ground type $\rho \in \mathbf{T} - \{\mathsf{nat}, \mathsf{bool}\}$, we have $\mathbf{B}^{\mathrm{d}}(\rho) = \emptyset$. The corresponding set of non-standard types, called *dead-code types* (*d-types* for short), is denoted by $\mathbf{L}^{\mathrm{d}}$.

**Dead-code types (semantics)**

For every $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$, let

- $\mathbf{P}(\delta^\iota)^{\mathcal{M}^{\mathrm{gnd}}} = \{\langle [M]^{\mathcal{M}^{\mathrm{gnd}}}, [M]^{\mathcal{M}^{\mathrm{gnd}}}\rangle \mid [M]^{\mathcal{M}^{\mathrm{gnd}}} \in \mathbf{I}(\iota)^{\mathcal{M}^{\mathrm{gnd}}}\}$, and
- $\mathbf{P}(\top^\rho)^{\mathcal{M}^{\mathrm{gnd}}} = \mathbf{I}(\iota)^{\mathcal{M}^{\mathrm{gnd}}} \times \mathbf{I}(\iota)^{\mathcal{M}^{\mathrm{gnd}}}$.

This semantics says that, for every $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$,

- $\delta^\iota$ (which is interpreted as the diagonal p.e.r. over $\mathbf{I}(\iota)^{\mathcal{M}^{\mathrm{gnd}}}$, consisting of the classes equating each element of $\mathbf{I}(\iota)^{\mathcal{M}^{\mathrm{gnd}}}$ only with itself) characterizes terms whose values have a precise identity ("useful" terms), since its interpretation codifies the fact that an occurrence of a term having this non-standard type is allowed to be replaced only with an observational equivalent term,
- while $\top^\iota$ (which is interpreted as the trivial p.e.r. over $\mathbf{I}(\iota)^{\mathcal{M}^{\mathrm{gnd}}}$, consisting of one class equating all the elements of $\mathbf{I}(\iota)^{\mathcal{M}^{\mathrm{gnd}}}$) characterizes terms whose values do not have a precise identity ("useless" terms), since its interpretation codifies the fact that an occurrence of a term having this non-standard type is allowed to be replaced with any term of the same type.

Let

- $\mathbf{L}^{\mathrm{d}}_\delta$ be the subset of the d-types $\phi$ identifying useful terms, i.e. such that $[\![\phi]\!]^{\mathcal{M}^{\mathrm{gnd}}} = \{\langle [M]^{\mathcal{M}^{\mathrm{gnd}}}, [M]^{\mathcal{M}^{\mathrm{gnd}}}\rangle \mid [M]^{\mathcal{M}^{\mathrm{gnd}}} \in \mathbf{I}(\epsilon(\phi))^{\mathcal{M}^{\mathrm{gnd}}}\}$, which (as we will see in Chapter 7) is exactly the set of d-types containing only $\delta$'s, and
- let $\mathbf{L}^{\mathrm{d}}_\top$ be the subset of the d-types $\phi$ identifying useless terms, i.e. such that $[\![\phi]\!]^{\mathcal{M}^{\mathrm{gnd}}} = \mathbf{I}(\epsilon(\phi))^{\mathcal{M}^{\mathrm{gnd}}} \times \mathbf{I}(\epsilon(\phi))^{\mathcal{M}^{\mathrm{gnd}}}$, which (as we will see in Chapter 7, Proposition 7.7) is inductively defined by the following clauses:
  - $\top^\iota \in \mathbf{L}^{\mathrm{d}}_\top$, for $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$
  - $\phi \to \psi \in \mathbf{L}^{\mathrm{d}}_\top$, for $\phi \in \mathbf{L}^{\mathrm{d}}$ and $\psi \in \mathbf{L}^{\mathrm{d}}_\top$
  - $\phi_1 \times \phi_2 \in \mathbf{L}^{\mathrm{d}}_\top$, for $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{d}}_\top$.

To detect the dead code in a given term $M$ we will assume that $M$ is a useful term (i.e. we force it to have a d-type in $\mathbf{L}^{\mathrm{d}}_\delta$) and try to assign the maximum amount of $\top$'s to its subterms (i.e. to detect as many useless subterms as possible).

**Entailment rules**

We introduce the following rule capturing the semantics of the basic properties:

$$(\top)\ \frac{\psi \in \mathbf{L}^{\mathrm{d}}_\top \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}\ .$$

**Assignment rules**

In Section 4.2.2 we will introduce rules for the program constructors $\mathsf{if}$, $\mathsf{fix}$, $\mathsf{rec}$, $\mathsf{it}$ and $\mathsf{case}$. Here we just look at a simple example.

Take the term

$$P = \lambda x^{\mathsf{nat}}.(\lambda y^{\mathsf{nat}}.x)x,$$

of type $\rho = \mathsf{nat} \to \mathsf{nat}$. By using the assignment rules in Fig. 3.5 it is possible to assign to $P$ d-type

$$\delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}} \text{ (useful term of type } \mathsf{nat} \to \mathsf{nat}),$$

by a derivation which assigns d-type

$$\top^{\mathsf{nat}} \text{ (useless term of type } \mathsf{nat})$$

to the second occurrence of $x$ in the body of the $\lambda$-abstraction $P$. In fact, for $\Sigma = \{x : \delta^{\mathsf{nat}}, y : \top^{\mathsf{nat}}\}$ and $\Sigma' = \{x : \delta^{\mathsf{nat}}\}$, we have the following derivation.

$$(\to \mathrm{E})\ \frac{(\to \mathrm{I})\ \dfrac{(\mathrm{Var})\ \Sigma \vdash x : \delta^{\mathsf{nat}}}{\Sigma' \vdash \lambda y^{\mathsf{nat}}.x : \top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}} \qquad (\leq)\ \dfrac{(\mathrm{Var})\ \Sigma' \vdash x : \delta^{\mathsf{nat}} \quad \delta^{\mathsf{nat}} \leq \top^{\mathsf{nat}}}{\Sigma' \vdash x : \top^{\mathsf{nat}}}}{(\to \mathrm{I})\ \dfrac{\Sigma' \vdash (\lambda y^{\mathsf{nat}}.x)x : \delta^{\mathsf{nat}}}{\emptyset \vdash \lambda x^{\mathsf{nat}}.(\lambda y^{\mathsf{nat}}.x)x : \delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}}}$$

Since we are interested in finding the subterms to which $\top$-d-types are assigned, in Chapter 7 we will consider judgements of the form

$$\Sigma \vdash M^{\phi},$$

where $M^{\phi}$ is a decorated term, i.e. a term in which each subterm is decorated by a d-type. Using the new judgement (and writing, for short, $\delta$ instead of $\delta^{\mathsf{nat}}$ and $\top$ instead of $\top^{\mathsf{nat}}$) the above derivation takes the form

$$(\to \mathrm{E})\ \frac{(\to \mathrm{I})\ \dfrac{(\mathrm{Var})\ \Sigma \vdash x^{\delta}}{\Sigma' \vdash (\lambda y^{\top}.x^{\delta})^{\top \to \delta}} \qquad (\leq)\ \dfrac{(\mathrm{Var})\ \Sigma' \vdash x^{\delta} \quad \delta \leq \top}{\Sigma' \vdash x^{\top}}}{(\to \mathrm{I})\ \dfrac{\Sigma' \vdash ((\lambda y^{\top}.x^{\delta})^{\top \to \delta} x^{\top})^{\delta}}{\emptyset \vdash (\lambda x^{\delta}.((\lambda y^{\top}.x^{\delta})^{\top \to \delta} x^{\top})^{\delta})^{\delta \to \delta}}}$$

Note that the use of decorated terms allows us to see that the second occurrence of $x$ in the body of the $\lambda$-abstraction $P$ is dead-code just by inspecting the final judgement

$$\emptyset \vdash (\lambda x^{\delta}.((\lambda y^{\top}.x^{\delta})^{\top \to \delta} x^{\top})^{\delta})^{\delta \to \delta}, \tag{3.1}$$

while with the original formulation of the assignment it is necessary to inspect the derivation tree.

Given the decorated version (3.1) of $P$ we can produce a simplified version of $P$ by removing the dead-code showed by the decoration (i.e. the second occurrence of $x$ in the

body of the $\lambda$-abstraction). In any case we can replace this dead-code by some placeholder $\mathsf{d}$ (as we will see in Section 7.2.3) and obtain

$$P' = \lambda x^{\mathsf{nat}}.(\lambda y^{\mathsf{nat}}.x)\mathsf{d}^{\mathsf{nat}},$$

which is ground observationally equivalent to $P$. But in some cases, like this, we can remove the dead-code in a clever way by performing a deeper transformation on the terms (as we will explain in Section 7.3.3). In fact we can remove also the formal parameter $y^{\mathsf{nat}}$ and obtain

$$P'' = \lambda x^{\mathsf{nat}}.x,$$

which is still ground observationally equivalent to $P$.

**Remark 3.2 (Dead-code analysis for Eager PCFP$^{\mathcal{T}}$)** *As we will explain in Sections 7.8 and 7.9, the dead-code detection and elimination techniques for PCFP$^{\mathcal{T}}$ programs developed in Part III of the thesis can be straightforwardly applied also to Eager PCFP$^{\mathcal{T}}$ programs. Indeed the program improvement introduced by dead-code elimination is even more relevant in the case of eager languages (see Section 7.9). Moreover the system Coq [Barras* et al.*, 1996] (which is the most significant example of proof assistant fully implementing a program extraction) supports, as target language for the program extraction, the eager language Caml [Leroy, 1997].*

## 3.3 About non-standard type constructors

When instantiating our framework to a particular program analysis we can gain expressiveness by introducing non-standard type constructors. In particular in Chapters 5 and 8 we consider, as a non-standard type constructor, the *conjunction* (or *intersection*) operator $\wedge$. We also consider, in Section 5.4, operators denoting downward and upward closures of the corresponding interpretations.

### 3.3.1 *Conjunctive* (or *intersection*) types

*Conjunctive types* have been introduced (under the name of *intersection types*) in [Coppo and Dezani-Ciancaglini, 1980; Coppo *et al.*, 1981] (see also [Barendregt *et al.*, 1983]) to overcome some weakness of the Hindley/Milner type system, and then extensively studied in literature (see, for instance, [van Bakel, 1993; Pierce, 1991]).

Conjunctive types can naturally be introduced in type inference systems in which the same term can have many different types: the basic idea is that a type of the form $\phi \wedge \psi$ is seen as a type of terms which have both type $\phi$ and type $\psi$. As we will see in Chapters 5 and 8 the introduction of conjunction greatly increases the power of the inference systems for strictness, totality and dead-code analyses (but the inference/checking problem, see Section 3.4, becomes more difficult).

## 3.4 Inference algorithms and checking algorithms

In this section we breafly discuss the problem of *implementing* a static analysis specified in the framework sketched in Section 3.1. To this aim we consider two different problems.

1. Given a term $M$ find (possibly all the) pairs $\langle \Sigma, \phi \rangle$ such that $\Sigma \vdash M : \phi$.
2. Given a term $M$ and a pair $\langle \Sigma, \phi \rangle$, does $\Sigma \vdash M : \phi$ hold?

The first problem is addressed by an *inference algorithm*, i.e. an algorithm that given $M$ computes (a representation of) pairs $\langle \Sigma, \phi \rangle$ for $M$, while the second one is addressed by a *checking algorithm*, i.e. an algorithm that answers to the question in point 2 above.

An inference algorithm is *complete* if it finds a representation of all the pairs (non-standard type, basis) for $M$, while a checking algorithm is *complete* if it answers to the question for all the possible pairs (non-standard type, basis).

If the set of non-standard types that can be assigned to a program is finite (as in the framework of Section 3.1) a complete checking algorithm can be used to build a complete inference algorithm, and vice versa. However, in many applications, the choice between inference and checking is crucial, since both the design and the use of the corresponding algorithms have different advantages and drawbacks.

In general the first step in designing an (inference or checking) algorithm for a given non-standard type assignment system is to provide a syntax-directed version of the assignment system. In some cases this step can reduce the power of the system (i.e. the set of properties that can be assigned to a given term).

## 3.5 A remark about instantiating the framework

As we will see in the rest of thesis, instantiating our framework to a particular analysis is not in general a straightforward task. It may involve some departures from the core formulation (like restricting the occurrences of the conjunction operator, to make the non-standard type manipulation easier, for the conjunctive analyses in Chapters 5, 8 and 9). Moreover, in Chapter 7, to simplify the presentation, we will introduce directly a syntax directed formulation of the dead-code type assignment system (which is immediately suggestive of an inference algorithm).

## 3.6 Operational semantics vs denotational semantics

Program analyses are often related to a *denotational semantics* of the programming language under analysis, using the mathematical theory of domains [Scott, 1982; Plotkin, 1981b]. This is mainly due to the fact that domain theory provides a very powerful tool for reasoning about program properties.

In this thesis we study program analyses relying exclusively on *operational semantics* [Plotkin, 1981a; Kahn, 1988]. This leads to a rather direct presentation which involves relatively little mathematical overhead.

## 3.7 Relations with the technique of Hankin and Le Métayer

In [Hankin and Le Métayer, 1994a] (see also [Hankin and Le Métayer, 1994b; Hankin and Le Métayer, 1995]), a framework for "non-standard type inference based" analyses of functional programs is presented. This framework mainly addresses algorithmic aspects, by providing a checking algorithm for a schematic conjunctive logic which is parameterized by the set of types and constants of the functional languages. The semantics of the non-standard types is captured by a general notion of "type structure" which can be "naturally" interpreted in most of the various representation of program properties which have been used in the literature.

In this thesis we take a different perspective, since we are mainly concerned to provide a foundation for the program analyses relying directly on operational semantics. Moreover we do not address algorithmic aspects in general, but we consider them only when considering particular analyses.

It is worth mentioning that some of the conjunctive non-standard type assignment systems presented in this thesis (see Remarks 5.32 and 5.51) can be implemented using the technique proposed in [Hankin and Le Métayer, 1994a][2]. However in this thesis we focus on inference algorithms (which involve quite different techniques). The use of inference algorithms is particular useful in the case of dead-code analysis, since (as we will see in Chapter 7) they allow to detect (in a straightforward way) the maximum amount of dead-code (which can can be proved by the corresponding non-standard type assignment) in the term in input.

## 3.8 How to use the results of the analyses

The framework proposed concerns the specification of program analysis. How to use the result of the analyses for performing optimizing program transformation depends from the particular analysis considered. We will breafly discuss this issue in Section 10.5. By the way

- in Part II of the thesis we do not consider at all how the results of the various strictness and totality analyses presented can be used,
- while, in Part III of the thesis, we will introduce, for each dead-code analysis presented, a program transformation based on the results of the analysis (following the ideas sketched at the end of Section 3.2.3).

[2] In particular, as explained in Remark 5.32, one of them (the conjunctive strictness analysis of Section 5.2.3) is actually implemented (see [Hankin and Le Métayer, 1994b]).

# Part II

# Strictness and totality analyses

# An introduction to strictness and totality analyses

In this part of the thesis we present various non-standard type inference systems for assigning strictness and/or totality properties to PCFP terms.

Both strictness and totality analyses have been proved useful in the implementation of lazy functional languages like Miranda, Haskell and Clean. For instance, if a function is strict it is safe to do the application of the function "by value", i.e. to evaluate the argument before performing the function call. Dually, when the argument of a function is known to terminate, it is safe to do the application of the function to that argument "by value".

Several approaches to the specification of strictness analysis have been investigated: abstract interpretation (see [Mycroft, 1980; Mycroft, 1981; Burn *et al.*, 1986; Abramsky, 1990a]), projections (see [Wadler and Hughes, 1987]), and non-standard type inference (see [Kuo and Mishra, 1989; Jensen, 1992; Benton, 1992a; Wright, 1991; Wright, 1992]). Totality analysis has been specified mainly by abstract interpretation (see [Mycroft, 1980; Mycroft, 1981; Abramsky, 1990a]). Moreover in [Solberg *et al.*, 1994] (see also [Solberg *et al.*, ; Solberg, 1995a; Solberg, 1995b]) it is introduced a non-standard type assignment system for performing combined strictness and totality analysis.

Following [Solberg *et al.*, 1994] we consider a language expressing both strictness and totality properties, i.e. we define, for each type $\rho \in \mathbf{T}$, a set $\mathbf{L}^{\mathrm{st}}(\rho)$ of strictness and totality types expressing properties of terms of type $\rho$. To this aim, for each type $\rho$, we choose as set of basic properties $\mathbf{B}^{\mathrm{st}}(\rho) = \{\bot^\rho, \not\bot^\rho, \top^\rho\}$, where

- $\bot^\rho$, denotes non terminating computations of type $\rho$,
- $\not\bot^\rho$, denotes terminating computations of type $\rho$, and
- $\top^\rho$, denotes every computation of type $\rho$.

Here divergence means not having a w.h.n.f., and convergence means having a w.h.n.f.. The symbols $\bot$, $\not\bot$, and $\top$ are seen as unary operators, called *basic property constructors*. Strictness and totality types are constructed from these basic properties using the standard type constructors (see Fig. 3.1 in Section 3.1.1). For instance, $\bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$ is the property of strict functions from integers to integers, while $\not\bot^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}}$ is the property of functions that, when applied to a terminating argument, results in a terminating computation.

If a function is strict it is safe to apply it "by value" and, dually, when the argument of a function is known to terminate, it is safe to do the application of the function "by value". I.e. given an application

$$FM_1 \cdots M_n,$$

where $n \geq 1$, it is possible to replace safely the lazy application of $FM_1 \cdots M_{k-1}$ to $M_k$ $(1 \leq k \leq n)$ with an eager "by value" application, if

- either $F$ has the property

$$\psi_1 \to \cdots \to \psi_{k-1} \to \bot^{\rho_k} \to \psi_{k+1} \to \cdots \to \psi_n \to \bot^{\sigma}$$

  for some $\psi_i$, and each $M_i$ ($i \in \{1, \ldots, k-1, k+1, \ldots, n\}$) has the property $\psi_i$,

- or $M_k$ has the property

$$\not\bot^{\rho_k} \ .$$

Let us look at the following example[3] to motivate the combined use of strictness and totality information.

**Example II.1** *Consider the program:*

$$\mathsf{let} \quad f = \lambda g.\lambda x.g(x), \quad h = \ldots, \quad y = \ldots \quad \mathsf{in} \quad f\,(\lambda x.x)\,y \,+\, f\,h\,1 \quad \mathsf{end} \ .$$

*Among the strictness types of the function $f$ there are:*

$$\bot^{\mathsf{nat}\to\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \qquad \textit{and} \qquad (\bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \ ,$$

*while the identity function on integers $\lambda x.x$ has strictness type $\bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$, and the integer number 1 has totality type $\not\bot^{\mathsf{nat}}$.*

*The strictness type $\bot^{\mathsf{nat}\to\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$ says that the function $f$ is strict in its first argument, therefore the application to the first argument can safely be done "by value". For the second argument no general consideration can be done, so we have to look at the properties of the two applications of $f$ to an argument (in the body of the* let *above): $f\,(\lambda x.x)$, and $f\,h$.*
*Since $f$ and $(\lambda\, x.x)$ have, respectively, the properties $(\bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$ and $\bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$, then $f\,(\lambda x.x)$ has the property $\bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$, i.e. it is strict in its argument. The same cannot be said for $f\,h$. However, knowing that, in this case, $f\,h$ is applied to an always terminating term, 1 (having totality type $\not\bot^{\mathsf{nat}}$), makes possible to say that for $f$ also the application to the second argument can be done "by value".*

[3] Taken from [Solberg, 1995a] page 28.

# Chapter 4

# Simple strictness and totality analyses

The first section of this chapter introduces a language of strictness and totality types and its semantics. Sections 4.2 and 4.3 introduce respectively a strictness type assignment system and a totality type assignment system for PCFP terms.

In Section 4.4 we present an assignment system for combined strictness and totality types (which extends the one proposed in [Solberg *et al.*, 1994; Solberg *et al.*, ] and [Solberg, 1995a] Chapter 2) and prove a separability result, i.e. we show that (w.r.t. the possibility of replacing safely a lazy application with a strict one) the strictness and totality types inferred by the combined system give the same information given by the strictness types inferred by the system of Section 4.2 and the totality types inferred by the system of Section 4.3.

A preliminary version of the material presented in this chapter appeared in [Coppo *et al.*, 1997], where the programming language considered does not include pairs.

## 4.1 Strictness and totality types

In this section we introduce a language of non-standard types, the language of strictness and totality types, which includes both the language of strictness types of Section 3.2.1 and the language of totality types of Section 3.2.2. I.e., for every type $\rho \in \mathbf{T}$, we choose as set of basic properties $\mathbf{B}^{\mathrm{st}}(\rho) = \{\bot^\rho, \not\bot^\rho, \top^\rho\}$.

**Definition 4.1 (Strictness and totality types)** *The language* $\mathbf{L}^{\mathrm{st}}$ *of* strictness and totality types *(*st-types *for short) is defined by the following grammar:* $\phi ::= a^\rho \mid \phi \to \phi \mid \phi \times \phi$, *where* $a \in \{\bot, \not\bot, \top\}$ *and* $\rho \in \mathbf{T}$.

Since st-types are properties of terms, in the following we use the words st-type and property interchangeably.

The meaning of st-types is formalized by using a set-semantics. I.e. each st-type $\phi$ is interpreted as a subset of the interpretation of the type $\epsilon(\phi)$. In particular $\top^\rho$ (which denotes absence of information) is interpreted as the whole interpretation of the type $\rho$ (meaning that we do not have any additional information on terms except their being of type $\rho$); $\bot^\rho$ (which denotes divergence) is interpreted as the singleton set containing the $\simeq^{\text{lazy}}_{\text{obs}}$-class of the closed terms of type $\rho$ not having a w.h.n.f., and $\not\bot^\rho$ (which denotes convergence) is interpreted as the set of the $\simeq^{\text{lazy}}_{\text{obs}}$-classes of the closed terms of type $\rho$ having a w.h.n.f..

**Definition 4.2 (Semantics of st-types)** *The interpretation $[\![\phi]\!]^{\mathcal{M}}$ of a st-type $\phi$ in the model $\mathcal{M} \in \{\mathcal{M}^{\text{lazy}}, \mathcal{M}^{\text{lazy}}_{\mathsf{isdef}}\}$ is defined by:*

$$
\begin{array}{lcl}
[\![\top^\rho]\!]^{\mathcal{M}} & = & \mathbf{I}(\rho)^{\mathcal{M}} \\
[\![\bot^\rho]\!]^{\mathcal{M}} & = & \{[\mathsf{fix}\, x^\rho.x]^{\mathcal{M}}\} \\
[\![\not\bot^\rho]\!]^{\mathcal{M}} & = & \mathbf{I}(\rho)^{\mathcal{M}} - \{[\mathsf{fix}\, x^\rho.x]^{\mathcal{M}}\} \\
[\![\phi \to \psi]\!]^{\mathcal{M}} & = & \{[M]^{\mathcal{M}} \mid \forall [N]^{\mathcal{M}} \in [\![\phi]\!]^{\mathcal{M}^{\text{lazy}}}.\ [MN]^{\mathcal{M}} \in [\![\psi]\!]^{\mathcal{M}}\} \\
[\![\phi_1 \times \phi_2]\!]^{\mathcal{M}} & = & \{[M]^{\mathcal{M}} \mid \forall i \in \{1,2\}.\ [\mathsf{proj}_i M]^{\mathcal{M}} \in [\![\phi_i]\!]^{\mathcal{M}}\}\ .
\end{array}
$$

In the rest of the chapter, whenever a statement holds for both the interpretations $\mathcal{M}^{\text{lazy}}$ and $\mathcal{M}^{\text{lazy}}_{\mathsf{isdef}}$, we will not use the superscript, i.e. we will simply write $\mathbf{I}(\rho)$, $[M]$, $[\![\phi]\!]$.

The st-types can be classified according to their rightmost (w.r.t. the arrow type constructor) basic property constructor. In particular st-types whose rightmost constructors are non informative ($\top$) are themselves non informative, they are called *$\top$-st-types*. The properties of being a term that, when applied to some arguments and/or projected (i.e. passed as argument to $\mathsf{proj}_1$ or $\mathsf{proj}_2$) a certain number (possibly 0) of times, result in a divergent (convergent) computation are represented by *$\bot$-st-types* (*$\not\bot$-st-types*).

**Definition 4.3 ($\top$-st-types, $\bot$-st-types and $\not\bot$-st-types)** *The set $\mathbf{L}^{\text{st}}_\top$ of $\top$-st-types is defined by:*

- $\top^\rho \in \mathbf{L}^{\text{st}}_\top$, *for* $\rho \in \mathbf{T}$
- $\phi \to \psi \in \mathbf{L}^{\text{st}}_\top$, *for* $\phi \in \mathbf{L}^{\text{st}}$ *and* $\psi \in \mathbf{L}^{\text{st}}_\top$
- $\phi_1 \times \phi_2 \in \mathbf{L}^{\text{st}}_\top$, *for* $\phi_1, \phi_2 \in \mathbf{L}^{\text{st}}_\top$.

*The sets $\mathbf{L}^{\text{st}}_\bot$ of $\bot$-st-types $\mathbf{L}^{\text{st}}_{\not\bot}$ of $\not\bot$-st-types are defined in a similar way with respectively $\bot$ and $\not\bot$ instead of $\top$.*

**Proposition 4.4** *$\phi \in \mathbf{L}^{\text{st}}_\top$ if and only if $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$.*

**Proof.** By structural induction on $\phi \in \mathbf{L}^{\mathrm{st}}$. The case $\phi \in \{\top^\rho, \bot^\rho, \not\bot^\rho\}$, for some $\rho \in \mathbf{T}$, is immediate by Definition 4.2.
For $\phi = \phi_1 \to \phi_2$, we have $\phi \in \mathbf{L}^{\mathrm{st}}_\top$ if and only if (by Definition 4.3) $\phi_2 \in \mathbf{L}^{\mathrm{st}}_\top$ if and only if (by induction hypothesis) $[\![\phi_2]\!] = \mathbf{I}(\epsilon(\phi_2))$ if and only if (by Definition 4.2) $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$. As for this last step, the *(only if)* part is immediate. For the *(if)* one observe that, for all $[P]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\phi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, $[\lambda z^{\epsilon(\phi_1)}.P]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\phi_1 \to \phi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$.
For $\phi = \phi_1 \times \phi_2$, we have $\phi \in \mathbf{L}^{\mathrm{st}}_\top$ if and only if (by Definition 4.3) $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{st}}_\top$ if and only if (by induction hypothesis) $[\![\phi_1]\!] = \mathbf{I}(\epsilon(\phi_1))$ and $[\![\phi_2]\!] = \mathbf{I}(\epsilon(\phi_2))$ if and only if (by Definition 4.2) $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$. □

The set of $\bot\top$-st-types (in Definition 4.5) collects all the properties that are satisfied by terms without w.h.n.f. (see Proposition 4.6.1), and its complement collects all the properties that are satisfied by terms with w.h.n.f. (see Proposition 4.6.2).

**Definition 4.5 ($\bot\top$-st-types)** *The set $\mathbf{L}^{\mathrm{st}}_{\bot\top}$ of $\bot\top$-st-types is defined by:*

- $\phi \in \mathbf{L}^{\mathrm{st}}_{\bot\top}$, *for* $\phi \in \mathbf{L}^{\mathrm{st}}_\bot \cup \mathbf{L}^{\mathrm{st}}_\top$
- $\phi \to \psi \in \mathbf{L}^{\mathrm{st}}_{\bot\top}$, *for* $\phi \in \mathbf{L}^{\mathrm{st}}$ *and* $\psi \in \mathbf{L}^{\mathrm{st}}_{\bot\top}$
- $\phi_1 \times \phi_2 \in \mathbf{L}^{\mathrm{st}}_{\bot\top}$, *for* $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{st}}_{\bot\top}$.

Note that the sets $\mathbf{L}^{\mathrm{st}}_\top$, $\mathbf{L}^{\mathrm{st}}_\bot$ and $\mathbf{L}^{\mathrm{st}}_{\not\bot}$ are pairwise disjoint. Moreover $\mathbf{L}^{\mathrm{st}}_\bot \cup \mathbf{L}^{\mathrm{st}}_\top \subset \mathbf{L}^{\mathrm{st}}_{\bot\top}$ and $\mathbf{L}^{\mathrm{st}}_{\not\bot} \cap \mathbf{L}^{\mathrm{st}}_{\bot\top} = \emptyset$.

**Proposition 4.6** *1. $\phi \in \mathbf{L}^{\mathrm{st}}_{\bot\top}$ if and only if $[\![\bot^{\epsilon(\phi)}]\!] \subseteq [\![\phi]\!]$, and*

*2. $\phi \in \mathbf{L}^{\mathrm{st}} - \mathbf{L}^{\mathrm{st}}_{\bot\top}$ if and only if $[\![\phi]\!] \subseteq [\![\not\bot^{\epsilon(\phi)}]\!]$.*

**Proof.**

1. Similar to the proof of Proposition 4.4.
2. Immediate from point 1., by observing that $[\![\phi]\!] \subseteq [\![\not\bot^{\epsilon(\phi)}]\!]$ if and only if (by Definition 4.2) $[\![\bot^{\epsilon(\phi)}]\!] \not\subseteq [\![\phi]\!]$. □

As we will see in Section 4.4, this language of non-standard types is indeed quite difficult to study, owing to the rather complex interactions of the strictness and totality properties expressed by the non-standard types. Moreover, in Section 4.4.5, we will show that this interactions are rather weak when considering the st-types that can be assigned to terms. For this reason in the next two sections we study strictness and totality properties, and their assignment to PCFP terms, separately.

## 4.2 Strictness analysis

In this section we carry on the strictness analysis introduced in Section 3.2.1. Note that the set of *strictness types* (*s-types* for short) $\mathbf{L}^{\mathrm{s}}$ (introduced in Section 3.2.1) is exactly the subset of the st-types (Definition 4.1) that do not contain the constructor $\not\bot$.

$$(\bot)\ \bot^{\epsilon(\phi)} \leq \phi \quad (\top)\ \frac{\psi \in \mathbf{L}^{\mathrm{s}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}$$

$$(\mathrm{Ref})\ \phi \leq \phi \quad (\rightarrow)\ \frac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_2 \rightarrow \psi_1 \leq \phi_1 \rightarrow \psi_2} \quad (\times)\ \frac{\phi_1 \leq \psi_1 \quad \phi_2 \leq \psi_2}{\phi_1 \times \phi_2 \leq \psi_1 \times \psi_2}$$

Figure 4.1: Entailment rules for strictness types (system $\leq_{\mathrm{s}}$)

### 4.2.1 An entailment relation for strictness types

**Definition 4.7 (Entailment relation for strictness types)** *Let $\phi, \psi \in \mathbf{L}^{\mathrm{s}}$. We write $\phi \leq_{\mathrm{s}} \psi$ to mean that $\phi \leq \psi$ is derivable by the rules in Fig. 4.1. By $\cong_{\mathrm{s}}$ we denote the equivalence relation induced by $\leq_{\mathrm{s}}$.*

Note that the relation $\leq_{\mathrm{s}}$ is transitive, since the rule (Trans) of Fig. 3.4 is admissible. For every $\rho \in \mathbf{T}$, rule $(\bot)$ states that $\bot^{\rho}$ is the smallest s-type in

$$\mathbf{L}^{\mathrm{s}}(\rho) = \{\phi \mid \phi \in \mathbf{L}^{\mathrm{s}} \text{ and } \epsilon(\phi) = \rho\},$$

and therefore the most informative. Indeed the property $\bot^{\rho}$ is characterizes the divergent terms of type $\rho$. Rule $(\top)$ says that the $\top$-s-types in $\mathbf{L}^{\mathrm{s}}_{\top}(\rho) = \mathbf{L}^{\mathrm{s}}(\rho) \cap \mathbf{L}^{\mathrm{st}}_{\top}(\rho)$ are the maximal elements of $\mathbf{L}^{\mathrm{s}}(\rho)$ and are all equivalent ($\top$-s-types do not give any information about terms). Rules (Ref), $(\rightarrow)$ and $(\times)$ are standard.

Soundness of entailment $\leq_{\mathrm{s}}$ (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$) is stated by the following theorem.

**Theorem 4.8 (Soundness of $\leq_{\mathrm{s}}$)** *Let $\phi, \psi \in \mathbf{L}^{\mathrm{s}}$. $\phi \leq_{\mathrm{s}} \psi$ implies $[\![\phi]\!] \subseteq [\![\psi]\!]$.*

**Proof.** By induction on the derivation of the entailment judgment. □

**Proposition 4.9** *Let $\phi \in \mathbf{L}^{\mathrm{s}}$. $[\![\phi]\!] = [\![\top^{\epsilon(\phi)}]\!]$ if and only if $\phi \in \mathbf{L}^{\mathrm{s}}_{\top}$ if and only if $\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}$.*

**Proof.** $[\![\phi]\!] = [\![\top^{\epsilon(\phi)}]\!]$ implies (by Proposition 4.4, since $\mathbf{L}^{\mathrm{s}}_{\top} = \mathbf{L}^{\mathrm{s}} \cap \mathbf{L}^{\mathrm{st}}_{\top}$) $\phi \in \mathbf{L}^{\mathrm{s}}_{\top}$, which implies (by rule $(\top)$ of $\leq_{\mathrm{s}}$) $\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}$, which implies (by Theorem 4.8) $[\![\phi]\!] = [\![\top^{\epsilon(\phi)}]\!]$. □

The entailment rules of Definition 4.7 are not complete w.r.t. the semantics $\mathcal{M}^{\mathrm{lazy}}$, as the following example shows.

**Example 4.10 (Incompleteness of $\leq_{\mathrm{s}}$ w.r.t. $\mathcal{M}^{\mathrm{lazy}}$)** *In $\Lambda_{\mathbf{T}}$ the functional terms with w.h.n.f. cannot be discriminated. For instance even though*

$$[\![\bot^{\mathsf{nat}\rightarrow\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} \neq [\![\top^{\mathsf{nat}} \rightarrow \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}}$$

*(in particular $[\![\top^{\mathsf{nat}} \rightarrow \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} = [\![\bot^{\mathsf{nat}\rightarrow\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} \cup \{[\lambda\, y^{\mathsf{nat}}.\mathsf{fix}\, x^{\mathsf{nat}}.x]^{\mathcal{M}^{\mathrm{lazy}}}\}$), we have that*

$$[\![\bot^{\mathsf{nat}\rightarrow\mathsf{nat}} \rightarrow \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} = [\![(\top^{\mathsf{nat}} \rightarrow \bot^{\mathsf{nat}}) \rightarrow \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} \quad .$$

*This is because it is not possible to write a context of* ground type *with a "hole" of functional type that converges if and only if its argument has a w.h.n.f., and, in particular, that converges when filled by* $\lambda\, y^{\mathsf{nat}}.\mathsf{fix}\, x^{\mathsf{nat}}.x$ *and diverges when filled by* $\mathsf{fix}\, z^{\mathsf{nat}\to\mathsf{nat}}.z$*. However*

$$\bot^{\mathsf{nat}\to\mathsf{nat}} \to \bot^{\mathsf{nat}} \not\leq_{\mathrm{s}} (\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \ .$$

The entailment $\leq_{\mathrm{s}}$ is complete w.r.t. the term model $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$ of Section 2.4.2. To our knowledge this completeness result is new.

**Theorem 4.11 (Completeness of $\leq_{\mathrm{s}}$ w.r.t. $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$)** *Let* $\phi, \psi \in \mathbf{L}^{\mathrm{s}}$*.* $[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ *imply* $\phi \leq_{\mathrm{s}} \psi$*.* □

**Notation 4.12** *Let* $W, W' \subseteq \Lambda^{\mathsf{isdef}}_{\mathbf{T}}$ *and* $M \in \Lambda^{\mathsf{isdef}}_{\mathbf{T}}$*. We define:*

$$\begin{array}{lcl} [W]^{\mathcal{M}} & = & \{[N]^{\mathcal{M}} \mid N \in W\}, \\ WM & = & \{NM \mid N \in W\}, \\ MW & = & \{MN \mid N \in W\}, \\ WW' & = & \{NN' \mid N \in W \text{ and } N' \in W'\}. \end{array}$$

To prove completeness we define for each s-type $\phi$ a set of terms, $U^{\phi}$, which characterizes the property $\phi$, i.e. such that for every s-type $\psi$, $[U^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ if and only if $\phi \leq_{\mathrm{s}} \psi$. We also need to define a set of test terms, $V^{\psi}$, such that, for every s-type $\phi$, if every test term in $V^{\psi}$ diverges when applied to a characteristic term in $U^{\phi}$, then $\phi \leq_{\mathrm{s}} \psi$. The definition is by mutual induction on s-types.

**Notation 4.13** *Let* $\mathrm{C}^{\rho}_{\top}$ *be defined by:* $\mathrm{C}^{\mathsf{nat}}_{\top} = 0$*,* $\mathrm{C}^{\mathsf{bool}}_{\top} = \mathsf{true}$*,* $\mathrm{C}^{\rho_1\to\rho_2}_{\top} = \lambda z^{\rho_1}.\mathrm{C}^{\rho_2}_{\top}$ *and* $\mathrm{C}^{\rho_1\times\rho_2}_{\top} = \langle \mathrm{C}^{\rho_1}_{\top}, \mathrm{C}^{\rho_2}_{\top} \rangle$*.*

**Definition 4.14** *Let* $\phi$ *be an s-type. The sets of closed terms* $\mathrm{U}^{\phi} \subseteq \Lambda^{\mathsf{isdef}}_{\mathbf{T}}$ *and* $\mathrm{V}^{\phi} \subseteq \Lambda^{\mathsf{isdef}}_{\mathbf{T}}$ *are defined according to the clauses in Fig. 4.2, where (for sake of readability) the* else *branches associated to conditions that never evaluates to* false *(i.e. either divergent or evaluating to* true*) are filled with the symbol* $\sharp$*.*

The following proposition can be easily proved by induction on s-types.

**Proposition 4.15** *Let* $\phi$ *be an s-type.*

1. $[\mathrm{U}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$*, and*
2. $[\mathrm{V}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\phi \to \bot^{\epsilon(\phi)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ *and, for every* $[M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in \mathbf{I}(\epsilon(\phi))^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$*,* $[\mathrm{V}^{\phi}M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = \{[\mathsf{fix}\, x^{\epsilon(\phi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}, [\mathsf{true}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}\}$*.* □

The proof of completeness rely on the following lemmas.

**Lemma 4.16** *Let* $\phi \in \mathbf{L}^{\mathrm{s}}$*.*

- If $\phi \cong_{\mathrm{s}} \top^{\rho}$, then
  - $\mathrm{U}^{\phi} = \{\mathrm{C}^{\rho}_{\top}\}$
  - $\mathrm{V}^{\phi} = \emptyset$
- If $\phi = \bot^{\rho}$, then
  - $\mathrm{U}^{\phi} = \{\mathsf{fix}\, x^{\rho}.x\}$
  - $\mathrm{V}^{\phi} = \{\lambda z^{\rho}.\mathsf{isdef}(z)\}$
- If $\phi = \phi_1 \to \phi_2 \ncong_{\mathrm{s}} \top^{\epsilon(\phi)}$, then
  - $\mathrm{U}^{\phi} = \{\lambda z^{\epsilon(\phi_1)}.P \mid P \in \mathrm{U}^{\phi_2}\} \cup \{\lambda z^{\epsilon(\phi_1)}.\mathsf{if}\ Qz\ \mathsf{then}\ \mathrm{C}^{\rho}_{\top}\ \mathsf{else}\ \sharp \mid Q \in \mathrm{V}^{\phi_1}\}$
  - $\mathrm{V}^{\phi} = \{\lambda z^{\epsilon(\phi)}.Q(zP) \mid Q \in \mathrm{V}^{\phi_2}\ \text{and}\ P \in \mathrm{U}^{\phi_1}\}$
- If $\phi = \phi_1 \times \phi_2 \ncong_{\mathrm{s}} \top^{\epsilon(\phi)}$, then
  - $\mathrm{U}^{\phi} = \{\langle P_1, P_2\rangle \mid P_1 \in \mathrm{V}^{\phi_1}\ \text{and}\ P_2 \in \mathrm{U}^{\phi_2}\}$
  - $\mathrm{V}^{\phi} = \{\lambda z^{\epsilon(\phi)}.Q(\mathsf{proj}_1 z) \mid Q \in \mathrm{V}^{\phi_1}\} \cup \{\lambda z^{\epsilon(\phi)}.Q(\mathsf{proj}_2 z) \mid Q \in \mathrm{V}^{\phi_2}\}$

Figure 4.2: Sets $\mathrm{U}^{\phi}$ and $\mathrm{V}^{\phi}$

1. *$\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ if and only if* $[\mathrm{C}^{\epsilon(\phi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$.

2. *$\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ if and only if, for all* $[\mathrm{V}^{\phi}\mathrm{C}^{\epsilon(\phi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = \{[\mathsf{fix}\, x^{\epsilon(\phi)}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}\}$.

**Proof.**

1. *(only if)* Immediate, since $\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ implies $[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\![\top^{\epsilon(\phi)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ by Theorem 4.8 (soundness of $\leq_{\mathrm{s}}$).
   *(if)* The case $\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ is immediate. The other cases are by structural induction on $\phi \in \mathbf{L}^{\mathrm{s}}$.
   The case $\phi = \bot^{\epsilon(\phi)}$ is immediate.
   Let $\phi = \phi_1 \to \phi_2 \ncong_{\mathrm{s}} \top^{\epsilon(\phi)}$, then $\phi_2 \ncong_{\mathrm{s}} \top^{\epsilon(\phi_2)}$ (by Proposition 4.9). By induction hypothesis $[\mathrm{C}^{\epsilon(\phi_2)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \notin [\![\phi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$. Now note that $\mathrm{C}^{\epsilon(\phi_1\to\phi_2)}_{\top} = \lambda z^{\epsilon(\phi_1)}.\mathrm{C}^{\epsilon(\phi_2)}_{\top}$, so, for every $[M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\phi_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, $[\mathrm{C}^{\epsilon(\phi_1\to\phi_2)}_{\top}M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathrm{C}^{\epsilon(\phi_2)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \notin [\![\phi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, which implies $[\mathrm{C}^{\epsilon(\phi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \notin [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$.
   The case $\phi = \phi_1 \times \phi_2 \ncong_{\mathrm{s}} \top^{\epsilon(\phi)}$ is similar.

2. *(only if)* Immediate, since $\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ implies $\mathrm{V}^{\phi} = \emptyset$.
   *(if)* The case $\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ is immediate. The other cases are by structural induction on $\phi \in \mathbf{L}^{\mathrm{s}}$.

The case $\phi = \bot^{\epsilon(\phi)}$ is immediate.
Let $\phi = \phi_1 \to \phi_2 \not\cong_{\mathrm{s}} \top^{\epsilon(\phi)}$, then $\phi_2 \not\cong_{\mathrm{s}} \top^{\epsilon(\phi_2)}$ (by Proposition 4.9). Since

$$[\mathrm{V}^{\phi_1\to\phi_2}\mathrm{C}_\top^{\epsilon(\phi_1\to\phi_2)}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathrm{V}^{\phi_2}\mathrm{C}_\top^{\epsilon(\phi_2)}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}},$$

the result follows by induction hypothesis.
The case $\phi = \phi_1 \times \phi_2 \not\cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ is similar. □

**Lemma 4.17** *Let $\phi, \psi \in \mathbf{L}^{\mathrm{s}}(\rho)$.*

1. *If $[\mathrm{U}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ then $\phi \leq_{\mathrm{s}} \psi$ .*
2. *If $[\mathrm{V}^{\psi}\mathrm{U}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = \{[\mathsf{fix}\, x^{\rho}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}\}$ then $\phi \leq_{\mathrm{s}} \psi$.*

**Proof.** We first consider the case in which either $\psi \cong_{\mathrm{s}} \top^{\epsilon(\psi)}$ or $\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}$.
Let $\underline{\psi \cong_{\mathrm{s}} \top^{\epsilon(\psi)}}$. Both point 1. and point 2. are immediate.
Let $\underline{\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}}$. The case $\psi \cong_{\mathrm{s}} \top^{\epsilon(\psi)}$ has been already considered, so let $\psi \not\cong_{\mathrm{s}} \top^{\epsilon(\psi)}$. Both point 1. and point 2. follows by Lemma 4.16.
If $\psi \not\cong_{\mathrm{s}} \top^{\epsilon(\psi)}$ and $\phi \not\cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ the proof is by structural induction on $\psi$ and $\phi$.

Let $\underline{\psi = \bot^{\epsilon(\psi)}}$. The case $\phi \cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ has been already considered, and the case $\phi = \bot^{\epsilon(\phi)}$ is trivial.
If $\phi = \phi_1 \to \phi_2 \not\cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ then $\phi \not\leq_{\mathrm{s}} \psi$. Both point 1. and point 2. are obvious since every $P \in \mathrm{U}^{\phi_1\to\phi_2}$ is a $\lambda$-abstraction, i.e. $P = \lambda x^{\epsilon(\phi_1)}.P'$ for some $P' \in \Lambda^{\mathsf{isdef}}_{\mathbf{T}}$, and $\mathrm{V}^{\psi} = \{\lambda z^{\epsilon(\psi)}.\mathsf{isdef}(z)\}$. So, for all $P \in [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$,

1. $[P]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \notin [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, and
2. $(\lambda z^{\epsilon(\psi)}.\mathsf{isdef}(z))P \Downarrow \mathsf{true}$.

The case $\phi = \phi_1 \times \phi_2 \not\cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ is analogous.

Let $\underline{\phi = \bot^{\epsilon(\phi)}}$. The case $\psi \cong_{\mathrm{s}} \top^{\epsilon(\psi)}$ and the case $\psi = \bot^{\epsilon(\psi)}$ have been already considered.
If $\psi = \psi_1 \to \psi_2 \not\cong_{\mathrm{s}} \top^{\epsilon(\psi)}$ then $\phi \leq_{\mathrm{s}} \psi$. Point 1. is obvious since $\mathrm{U}^{\phi} = \{\mathsf{fix}\, x^{\epsilon(\phi)}.x\}$. For point 2. note that $[\mathrm{V}^{\psi}\mathrm{U}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathrm{V}^{\psi_2}(\mathsf{fix}\, x^{\epsilon(\psi_2)}.x)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, and, by induction hypothesis and monotonicity, $[\mathrm{V}^{\psi_2}(\mathsf{fix}\, x^{\epsilon(\psi_2)}.x)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathsf{fix}\, y^{\mathsf{bool}}.y]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$.
The case $\psi = \psi_1 \times \psi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\psi)}$ is analogous.

Let $\underline{\psi = \psi_1 \to \psi_2 \not\cong_{\mathrm{s}} \top^{\epsilon(\psi)}}$. All the cases but $\phi = \phi_1 \to \phi_2 \not\cong_{\mathrm{s}} \top^{\epsilon(\phi)}$ have been already considered. Then 1. and 2. can be proved as follows.

1. $[\mathrm{U}^{\phi_1\to\phi_2}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\psi_1 \to \psi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ implies $[\mathrm{U}^{\phi_1\to\phi_2}\mathrm{U}^{\psi_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\psi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ (by Definition 4.14). But

$$[\mathrm{U}^{\phi_1\to\phi_2}\mathrm{U}^{\psi_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathrm{U}^{\phi_2}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \cup \{[\mathsf{if}\, R\, \mathsf{then}\, \mathrm{C}_\top^{\epsilon(\phi_2)}\, \mathsf{else}\, \sharp]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \mid R \in \mathrm{V}^{\phi_1}\mathrm{U}^{\psi_1}\},$$

and so $[\mathrm{U}^{\phi_1\to\phi_2}\mathrm{U}^{\psi_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\psi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ implies

- $[\mathrm{U}^{\phi_2}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\psi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ and
- $\{[\mathsf{if}\ R\ \mathsf{then}\ \mathrm{C}^{\epsilon(\phi_2)}_{\top}\ \mathsf{else}\ \sharp]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \mid R \in \mathrm{V}^{\phi_1}\mathrm{U}^{\psi_1}\} \subseteq [\![\psi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, which implies (by Lemma 4.16.1, since $\psi_2 \not\cong_{\mathrm{s}} \top^{\epsilon(\psi_2)}$) $[\mathrm{V}^{\phi_1}\mathrm{U}^{\psi_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathsf{fix}\, x^{\mathsf{bool}}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$.

Using the induction hypotheses we get $\psi_1 \leq_{\mathrm{s}} \phi_1$ and $\phi_2 \leq_{\mathrm{s}} \psi_2$, which imply $\phi_1 \to \phi_2 \leq_{\mathrm{s}} \psi_1 \to \psi_2$.

2. $[\mathrm{V}^{\psi_1\to\psi_2}\mathrm{U}^{\phi_1\to\phi_2}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = \{[\mathsf{fix}\, x^{\mathsf{bool}}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}\}$ implies

   - $[\mathrm{V}^{\psi_2}\mathrm{U}^{\phi_2}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq \{[\mathsf{fix}\, x^{\mathsf{bool}}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}\}$, and
   - $[\mathrm{V}^{\psi_2}\{\mathsf{if}\ R\ \mathsf{then}\ \mathrm{C}^{\epsilon(\phi_2)}_{\top}\ \mathsf{else}\ \sharp \mid R \in \mathrm{V}^{\phi_1}\mathrm{U}^{\psi_1}\}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq \{[\mathsf{fix}\, x^{\mathsf{bool}}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}\}$, which implies (by Lemma 4.16.2, since $\psi_2 \not\cong_{\mathrm{s}} \top^{\epsilon(\psi_2)}$) $[\mathrm{V}^{\phi_1}\mathrm{U}^{\psi_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathsf{fix}\, x^{\mathsf{bool}}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$.

   Using the induction hypotheses we get $\psi_1 \leq_{\mathrm{s}} \phi_1$ and $\phi_2 \leq_{\mathrm{s}} \psi_2$, which imply $\phi_1 \to \phi_2 \leq_{\mathrm{s}} \psi_1 \to \psi_2$.

The proof of the only remaining case, $\underline{\psi = \psi_1 \times \psi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\psi)}}$ and $\phi = \phi_1 \times \phi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\phi)}$, is similar to the proof of the previous one. □

From the previous proposition and lemmas we derive the completeness theorem.

**Proof of Theorem 4.11.** By contraposition. Assume that $\phi \not\leq_{\mathrm{s}} \psi$. Then $\psi \not\cong_{\mathrm{s}} \top^{\epsilon(\psi)}$. By Lemma 4.17.1, $[\mathrm{U}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \not\subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$. Since from Proposition 4.15.1 $[\mathrm{U}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, this implies that $[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \not\subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$. □

**Remark 4.18** *As showed in Example 4.10 the entailment relation $\leq_{\mathrm{s}}$ is not complete w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$. At the moment we do not know whether a sound (and decidable) extension of the $\leq_{\mathrm{s}}$ relation which is complete w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$ exists.*

### 4.2.2 An assignment system for strictness types

For each constant $\mathsf{c}$ a set $\mathbf{L}^{\mathrm{s}}(\mathsf{c})$ of minimal s-types is specified. The idea is that every s-type which is entailed ($\leq_{\mathrm{s}}$) by an element of $\mathbf{L}^{\mathrm{s}}(\mathsf{c})$ can be assigned to $\mathsf{c}$. For instance, for all integers $\mathsf{n}$, $\mathbf{L}^{\mathrm{s}}(\mathsf{n}) = \{\top^{\mathsf{nat}}\}$ and $\mathbf{L}^{\mathrm{s}}(+) = \{\bot^{\mathsf{nat}} \times \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}, \top^{\mathsf{nat}} \times \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}\}$.

**Definition 4.19 (Strictness type assignment system)** *We write $\Sigma \vdash_{\mathrm{s}} M : \phi$ to mean that $\Sigma \vdash M : \phi$ can be derived by the rules in Fig. 4.3.*

Note that, being $\vdash_{\mathrm{s}}$ an inference system, the same terms can have different s-types.

**Example 4.20** *Let $\vdash_{\mathbf{T}}$ twice : $\rho$ where $\rho = (\sigma \to \sigma) \to \sigma \to \sigma$ and* twice $= \lambda f^{\sigma\to\sigma}.\ \lambda x^{\sigma}.\ f(f\,x)$. *It is easy to check that, for $i \in \{1, 2\}$, $\vdash$ twice : $\phi_i$, where $\phi_1 = (\bot^{\sigma} \to \bot^{\sigma}) \to \bot^{\sigma} \to \bot^{\sigma}$ and $\phi_2 = (\top^{\sigma} \to \bot^{\sigma}) \to \top^{\sigma} \to \bot^{\sigma}$.*

The proof of the following fact is immediate.

$$(\leq)\ \frac{\Sigma \vdash M : \phi \quad \phi \leq \psi}{\Sigma \vdash M : \psi}$$

$$(\text{Var})\ \Sigma, x : \phi \vdash x : \phi \qquad (\text{Con})\ \frac{\phi \in \mathbf{L}^{\mathrm{s}}(\mathsf{c})}{\Sigma \vdash \mathsf{c} : \phi}$$

$$(\to \text{I})\ \frac{\Sigma, x : \phi \vdash M : \psi}{\Sigma \vdash \lambda x.M : \phi \to \psi} \qquad (\to \text{E})\ \frac{\Sigma \vdash M : \phi \to \psi \quad \Sigma \vdash N : \phi}{\Sigma \vdash MN : \psi}$$

$$(\times \text{I})\ \frac{\Sigma \vdash M_1 : \phi_1 \quad \Sigma \vdash M_2 : \phi_2}{\Sigma \vdash \langle M_1, M_2 \rangle : \phi_1 \times \phi_2} \qquad (\times \text{E}_i)\ \frac{\Sigma \vdash M : \phi_1 \times \phi_2}{\Sigma \vdash \mathsf{proj}_i M : \phi_i}\ i \in \{1, 2\}$$

$$(\text{Fix}_\bot)\ \frac{\Sigma, x : \phi_1 \vdash M : \phi_2 \ \cdots\ \Sigma, x : \phi_k \vdash M : \phi_1}{\Sigma \vdash \mathsf{fix}\, x.M : \phi_i}\ 1 \leq i \leq k$$

$$(\text{If}_\bot)\ \frac{\Sigma \vdash N : \bot^{\mathsf{bool}} \qquad \vdash_{\mathbf{T}} M_1 : \rho \qquad \vdash_{\mathbf{T}} M_2 : \rho}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \bot^{\rho}}$$

$$(\text{If}_\top)\ \frac{\Sigma \vdash N : \top^{\mathsf{bool}} \qquad \Sigma \vdash M_1 : \phi \qquad \Sigma \vdash M_2 : \phi}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \phi}$$

Figure 4.3: Rules for strictness type assignment (system $\vdash_{\mathrm{s}}$)

**Fact 4.21** *1.* $\Sigma \vdash_{\mathrm{s}} M : \phi$ *implies* $\vdash_{\mathbf{T}} M : \epsilon(\phi)$ *and* $\epsilon(\Sigma) \supseteq \mathrm{FV}(M)$.

*2.* $\vdash_{\mathbf{T}} M : \rho$ *implies, for all basis* $\Sigma$ *such that* $\Sigma = \{x^{\epsilon(\psi)} : \psi \mid x^{\epsilon(\psi)} \in \mathrm{FV}(M)$ *for some* $\psi \in \mathbf{L}^{\mathrm{s}}\}$, $\Sigma \vdash_{\mathrm{s}} M : \top^{\rho}$. □

**Remark 4.22** *The language* $\mathbf{L}^{\mathrm{s}}$ *is a subset of the language of properties of the* conjunctive strictness logic *introduced in [Jensen, 1992] and (independently) in [Benton, 1992a]. This subset is obtained from the formulas of the conjunctive strictness logic, as presented in [Benton, 1992a] page 36, by:*

*1. using* $\bot$ *for* $\mathbf{f}$ *and* $\top$ *for* $\mathbf{t}$, *and*

*2. forbidding the use of the conjunction operator.*

*The first point is simply a syntactic variation, while the second restriction greatly reduces the power of the logic.*

*In particular, the rule for fixpoint,* $(\text{Fix}_\bot)$ *in Definition 4.19, is a restriction of the fixpoint rule for the conjunctive strictness logic in [Benton, 1992a] page 45 (see also [Jensen, 1992] page 47) that follows:*

$$(\text{fix})\ \frac{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash M : \phi_1 \wedge \cdots \wedge \phi_n}{\Sigma \vdash \mathsf{fix}\, x.M : \phi_1 \wedge \cdots \wedge \phi_n}\ .$$

*Indeed if in our system* $\Sigma \vdash_{\mathrm{s}} \mathsf{fix}\, x.M : \phi_i$ *then*

$$\Sigma, x : \phi_1 \vdash M : \phi_2 \quad \cdots \quad \Sigma, x : \phi_n \vdash M : \phi_1$$

*in the conjunctive strictness logic. Therefore, since* $\bigwedge_{1 \le i \le n} \phi_i \le \phi_j$ *for all* $1 \le j \le n$*, by conjunction introduction it is possible to derive*

$$\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash M : \phi_1 \wedge \cdots \wedge \phi_n$$

*and, from the rule* (fix) *and conjunction elimination,* $\Sigma \vdash \mathsf{fix}\, x.M : \phi_i$ *for all* $1 \le i \le n$*. Our rule* ($\mathrm{Fix}_\perp$) *introduces an implicit (and limited) use of conjunction, by allowing (in its antecedent) the derivation of more than one s-type for* $M$*.*

The following example shows an application of the ($\mathrm{Fix}_\perp$) rule.

**Example 4.23** *The function* $F = \mathsf{fix}\, f^{\rho}.M$*, where* $\rho = \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ *and*

$$M = \lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\lambda z^{\mathsf{nat}}.\mathsf{if}\ z = 0\ \mathsf{then}\ x + y\ \mathsf{else}\ fyx(z-1) \quad ,$$

*was used in [Kuo and Mishra, 1989] to show the limitation of a type system without conjunction.*

*Using* $\vdash_{\mathrm{s}}$ *we are able to prove that* $F$ *is strict in each of its 3 arguments, i.e.* $\emptyset \vdash_{\mathrm{s}} F : \phi_1$*,* $\emptyset \vdash_{\mathrm{s}} F : \phi_2$*, and* $\emptyset \vdash_{\mathrm{s}} F : \phi_3$*, where*

- $\phi_1 = \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$*,*
- $\phi_2 = \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$*, and*
- $\phi_3 = \top^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$*.*

*Note that to assign* $\phi_3$ *to* $F$ *one premise is sufficient, in fact* $\{f : \phi_3\} \vdash_{\mathrm{s}} M : \phi_3$*, and so it is possible to apply the rule* ($\mathrm{Fix}_\perp$) *as follows:*

$$(\mathrm{Fix}_\perp)\ \frac{\{f : \phi_3\} \vdash_{\mathrm{s}} M : \phi_3}{\emptyset \vdash_{\mathrm{s}} \mathsf{fix}\, f^{\rho}.M : \phi_3} \quad .$$

*To assign* $\phi_1$ *or* $\phi_2$ *to* $F$*, instead, two premises are required. In fact,* $\{f : \phi_1\} \vdash_{\mathrm{s}} M : \phi_2$ *and* $\{f : \phi_2\} \vdash_{\mathrm{s}} M : \phi_1$*, and so two premises are sufficient, since (for* $i \in \{1, 2\}$*):*

$$(\mathrm{Fix}_\perp)\ \frac{\{f : \phi_1\} \vdash_{\mathrm{s}} M : \phi_2 \quad \{f : \phi_2\} \vdash_{\mathrm{s}} M : \phi_1}{\emptyset \vdash_{\mathrm{s}} \mathsf{fix}\, f^{\rho}.M : \phi_i} \ ,$$

*but (as it is easy to check) they are also needed. So with the rule for fixpoint of [Kuo and Mishra, 1989] (that allows just one premise) neither* $\phi_1$ *nor* $\phi_2$ *can be assigned to* $F$*.*

Soundness of the s-types assignment system (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$) is stated by the following theorem.

**Theorem 4.24 (Soundness of $\vdash_{\mathrm{s}}$)** *Let* $\Sigma \vdash_{\mathrm{s}} M : \phi$*. Then* $[\![M]\!]_e \in [\![\phi]\!]$ *for all* $e$ *such that for all* $x : \psi \in \Sigma$*,* $e(x) \in [\![\psi]\!]$*.*

**Proof.** By induction on the derivation of $\Sigma \vdash_{\mathrm{s}} M : \phi$. □

### 4.2.3 Strictness analysis and ground contextual equivalence

The entailment relation presented in Section 4.2.1 can be modified to fit better the ground contextual equivalence, $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$, instead of the lazy contextual equivalence, $\simeq^{\mathrm{lazy}}_{\mathrm{obs}}$. In this perspective a s-type $\phi$ is interpreted as a subset of the interpretation of the type $\epsilon(\phi)$ in the model $\mathcal{M}^{\mathrm{gnd}}$ of Section 2.4.1. For every s-type $\phi \in \mathbf{L}^{\mathrm{s}}$, let $[\![\phi]\!]^{\mathcal{M}^{\mathrm{gnd}}}$ be the interpretation of $\phi$ in the model $\mathcal{M}^{\mathrm{gnd}}$ defined according to clauses of Definition 4.2.

The ground observational equivalence, $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$, is such that $\mathsf{fix}\, x^{\rho\to\sigma}.x \simeq^{\mathrm{gnd}}_{\mathrm{obs}} \lambda y^{\rho}.\mathsf{fix}\, z^{\sigma}.z$, so the associated entailment relation between s-types will identify all the s-types corresponding to functions that diverge for all inputs. Such s-types, called b-s-types are defined as follows.

**Definition 4.25 (b-s-types)** *The set $\mathbf{L}^{\mathrm{s}}_{\mathrm{b}}$ of b-s-types is defined by:*

- $\bot^{\rho} \in \mathbf{L}^{\mathrm{s}}_{\mathrm{b}}$, *for* $\rho \in \mathbf{T}$
- $\phi \to \psi \in \mathbf{L}^{\mathrm{s}}_{\mathrm{b}}$, *for* $\phi \in \mathbf{L}^{\mathrm{s}}_{\top}$ *and* $\psi \in \mathbf{L}^{\mathrm{s}}_{\mathrm{b}}$
- $\phi_1 \times \phi_2 \in \mathbf{L}^{\mathrm{s}}_{\mathrm{b}}$, *for* $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{s}}_{\mathrm{b}}$.

We can now introduce the entailment relation $\leq^{\mathrm{g}}_{\mathrm{s}}$; $\phi \leq^{\mathrm{g}}_{\mathrm{s}} \psi$ reflects the set-theoretic inclusion $[\![\phi]\!]^{\mathcal{M}^{\mathrm{gnd}}} \subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{gnd}}}$.

**Definition 4.26 (Entailment relation $\leq^{\mathrm{g}}_{\mathrm{s}}$)** *Let $\phi, \psi \in \mathbf{L}^{\mathrm{s}}$. We write $\phi \leq^{\mathrm{g}}_{\mathrm{s}} \psi$ to mean that $\phi \leq \psi$ is derivable by using the rules in Fig. 4.1 in which the axiom $(\bot)$ is replaced by the following*

$$(\bot_{\mathrm{g}})\ \frac{\phi \in \mathbf{L}^{\mathrm{s}}_{\mathrm{b}} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}\ .$$

*By $\cong^{\mathrm{g}}_{\mathrm{s}}$ we denote the equivalence relation induced by $\leq^{\mathrm{g}}_{\mathrm{s}}$.*

The entailment relation $\leq^{\mathrm{g}}_{\mathrm{s}}$ is sound and complete w.r.t. the semantics $\mathcal{M}^{\mathrm{gnd}}$. To our knowledge the completeness result is new.

**Theorem 4.27 (Soundness of $\leq^{\mathrm{g}}_{\mathrm{s}}$)** *Let $\phi, \psi \in \mathbf{L}^{\mathrm{s}}$. $\phi \leq^{\mathrm{g}}_{\mathrm{s}} \psi$ implies $[\![\phi]\!]^{\mathcal{M}^{\mathrm{gnd}}} \subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{gnd}}}$.*

**Proof.** By induction on the derivation of the entailment judgment. □

**Theorem 4.28 (Completeness of $\leq^{\mathrm{g}}_{\mathrm{s}}$ w.r.t. $\mathcal{M}^{\mathrm{gnd}}$)** *Let $\phi, \psi \in \mathbf{L}^{\mathrm{s}}$. $[\![\phi]\!]^{\mathcal{M}^{\mathrm{gnd}}} \subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{gnd}}}$ imply $\phi \leq^{\mathrm{g}}_{\mathrm{s}} \psi$.*

**Proof.** Similar to the proof of Theorem 4.11. □

**Definition 4.29 (Assignment system $\vdash^{\mathrm{g}}_{\mathrm{s}}$)** *We write $\Sigma \vdash^{\mathrm{g}}_{\mathrm{s}} M : \phi$ to mean that $\Sigma \vdash M : \phi$ can be derived by the rules in Fig. 4.3 using the entailment $\leq^{\mathrm{g}}_{\mathrm{s}}$ instead of $\leq_{\mathrm{s}}$.*

Soundness of the s-types assignment system $\vdash^{\mathrm{g}}_{\mathrm{s}}$ w.r.t. $\mathcal{M}^{\mathrm{gnd}}$ is stated by the following theorem.

**Theorem 4.30 (Soundness of $\vdash^{\mathrm{g}}_{\mathrm{s}}$)** *Let $\Sigma \vdash_{\mathrm{s}} M : \phi$. Then $[\![M]\!]^{\mathcal{M}^{\mathrm{gnd}}}_{e} \in [\![\phi]\!]^{\mathcal{M}^{\mathrm{gnd}}}$ for all $e$ such that for all $x : \psi \in \Sigma$, $e(x) \in [\![\psi]\!]^{\mathcal{M}^{\mathrm{gnd}}}$.*

**Proof.** By induction on the derivation of $\Sigma \vdash^{\mathrm{g}}_{\mathrm{s}} M : \phi$. □

**Remark 4.31** *The $\mathcal{M}^{\mathrm{gnd}}$ semantics identifies terms such as $\mathsf{fix}\, x^{\mathsf{nat}\to\mathsf{nat}}.x$ and $\lambda\, y^{\mathsf{nat}}.\ \mathsf{fix}\, z^{\mathsf{nat}}.z$. This choice is justified, as explained in [Benton, 1992a] page 19, by taking a view in which*

(∗) programs *are closed terms of basic type (*$\mathsf{nat}$ *or* $\mathsf{bool}$*) and the only behaviours that a program can exhibit are to diverge or to converge to an integer or boolean value.*

*So the only way in which we can observe terms of higher types is to plug them into a complete program. The semantics $\mathcal{M}^{\mathrm{lazy}}$, instead, does not identify $\mathsf{fix}\, x^{\mathsf{nat}\to\mathsf{nat}}.x$ and $\lambda\, y^{\mathsf{nat}}.\ \mathsf{fix}\, z^{\mathsf{nat}}.z$.*

*Since $M \simeq^{\mathrm{lazy}}_{\mathrm{obs}} N$ implies $M \simeq^{\mathrm{gnd}}_{\mathrm{obs}} N$, all optimizations preserving $\simeq^{\mathrm{lazy}}_{\mathrm{obs}}$ preserve also $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$. The vice versa is not true. For instance, consider the term $M = \lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda y^{\mathsf{nat}}.g\, y$. W.r.t. $\mathcal{M}^{\mathrm{gnd}}$ this term is strict in the first argument, since we can assign it (with both $\vdash_{\mathrm{s}}$ and $\vdash^{\mathrm{g}}_{\mathrm{s}}$) the s-type $(\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$, where $\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$ is a* b*-s-type. For any PCFP term $P$, the application $MP$ can be done "by value" (first evaluating $P$ and then doing the replacement) preserving the ground observational equivalence. W.r.t. $\mathcal{M}^{\mathrm{lazy}}$ this is not the case, since observing $MP$ at the type $\mathsf{nat} \to \mathsf{nat}$, when $P = \mathsf{fix}\, x^{\mathsf{nat}\to\mathsf{nat}}.x$, we have that $MP \Downarrow$ whereas the evaluation of the application $MP$ "by value" leads to a divergent computation, i.e. the lazy observational equivalence is not preserved.*

*The above considerations suggest that, under the assumption (∗), $\mathcal{M}^{\mathrm{gnd}}$ is indeed a more convenient model (rather than $\mathcal{M}^{\mathrm{lazy}}$) w.r.t. prove the soundness of strictness analysis. On the other hand (as we will see in Section 4.3.3) the model $\mathcal{M}^{\mathrm{gnd}}$ is not convenient when the kind of totality properties considered in this thesis are concerned.*

*As we will see in Section 4.4, the model $\mathcal{M}^{\mathrm{lazy}}$ allows the study of the interactions between strictness and totality.*

## 4.3 Totality analysis

In this section we carry on the totality analysis introduced in Section 3.2.2. Note that the set of *totality types* (*t-types* for short) $\mathbf{L}^{\mathrm{t}}$ (introduced in Section 3.2.2) is exactly the subset of the st-types (Definition 4.1) that do not contain the constructor $\bot$.

### 4.3.1 An entailment relation for totality types

**Definition 4.32 (Entailment relation for totality types)** *Let $\phi, \psi \in \mathbf{L}^{\mathrm{t}}$. We write $\phi \leq_{\mathrm{t}} \psi$ to mean that $\phi \leq \psi$ is derivable by the rules in Fig. 4.4. By $\cong_{\mathrm{t}}$ we denote the equivalence relation induced by $\leq_{\mathrm{t}}$.*

$$(\not\perp)\ \frac{\phi \in \mathbf{L}^{\mathrm{t}} - \mathbf{L}^{\mathrm{t}}_{\top}}{\phi \leq \not\perp^{\epsilon(\phi)}} \quad (\top)\ \frac{\psi \in \mathbf{L}^{\mathrm{t}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}$$

$$(\mathrm{Ref})\ \phi \leq \phi \quad (\to)\ \frac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_2 \to \psi_1 \leq \phi_1 \to \psi_2} \quad (\times)\ \frac{\phi_1 \leq \psi_1 \quad \phi_2 \leq \psi_2}{\phi_1 \times \phi_2 \leq \psi_1 \times \psi_2}$$

Figure 4.4: Entailment rules for totality types (system $\leq_{\mathrm{t}}$)

Also $\leq_{\mathrm{t}}$ (as $\leq_{\mathrm{s}}$) is a preorder relation, since the rule (Trans) in Fig. 3.4 is admissible. Comparing the entailment rules for $\leq_{\mathrm{t}}$ with the ones for $\leq_{\mathrm{s}}$ (in Figure 4.1) we see that they differ just for the axiom $(\not\perp)$, which is used instead of $(\perp)$. Axiom $(\not\perp)$ states that $\not\perp^{\rho}$ is the greatest t-type in

$$\mathbf{L}^{\mathrm{t}}(\rho) = \{\phi \mid \phi \in \mathbf{L}^{\mathrm{t}} - \mathbf{L}^{\mathrm{t}}_{\top} \text{ and } \epsilon(\phi) = \rho\}.$$

This means that the property $\not\perp^{\rho}$ is the least informative between the informative (i.e. $\notin \mathbf{L}^{\mathrm{t}}_{\top}$) termination properties, in fact it only says that the term has a w.h.n.f..

Soundness of the entailment between t-types (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$) is stated by the following theorem.

**Theorem 4.33 (Soundness of $\leq_{\mathrm{t}}$)** *Let $\phi, \psi \in \mathbf{L}^{\mathrm{t}}$. $\phi \leq_{\mathrm{t}} \psi$ implies $[\![\phi]\!] \subseteq [\![\psi]\!]$.*

**Proof.** By induction on the derivation of the entailment judgment. □

**Proposition 4.34** *Let $\phi \in \mathbf{L}^{\mathrm{t}}$. $[\![\phi]\!] = [\![\top^{\epsilon(\phi)}]\!]$ if and only if $\phi \in \mathbf{L}^{\mathrm{t}}_{\top}$ if and only if $\phi \cong_{\mathrm{t}} \top^{\epsilon(\phi)}$.*

**Proof.** $[\![\phi]\!] = [\![\top^{\epsilon(\phi)}]\!]$ implies (by Proposition 4.4, since $\mathbf{L}^{\mathrm{t}}_{\top} = \mathbf{L}^{\mathrm{t}} \cap \mathbf{L}^{\mathrm{st}}_{\top}$) $\phi \in \mathbf{L}^{\mathrm{t}}_{\top}$, which implies (by rule $(\top)$ of $\leq_{\mathrm{t}}$) $\phi \cong_{\mathrm{t}} \top^{\epsilon(\phi)}$, which implies (by Theorem 4.33) $[\![\phi]\!] = [\![\top^{\epsilon(\phi)}]\!]$. □

As for $\leq_{\mathrm{s}}$, the entailment $\leq_{\mathrm{t}}$ is not complete w.r.t. $\mathcal{M}^{\mathrm{lazy}}$.

**Example 4.35 (Incompleteness of $\leq_{\mathrm{t}}$ w.r.t. $\mathcal{M}^{\mathrm{lazy}}$)** *As explained in Example 4.10, in $\Lambda_{\mathbf{T}}$ the functional terms with w.h.n.f. cannot be discriminated. So, even though*

$$[\![\not\perp^{\mathsf{nat}\to\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} \neq [\![\top^{\mathsf{nat}\to\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}}$$

*(in particular $[\![\top^{\mathsf{nat}\to\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} = [\![\perp^{\mathsf{nat}\to\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} \cup [\![\not\perp^{\mathsf{nat}\to\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}}$), we have that*

$$[\![\not\perp^{\mathsf{nat}\to\mathsf{nat}} \to \not\perp^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} = [\![\top^{\mathsf{nat}\to\mathsf{nat}} \to \not\perp^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}}$$

*while*

$$\not\perp^{\mathsf{nat}\to\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \not\leq_{\mathrm{t}} \top^{\mathsf{nat}\to\mathsf{nat}} \to \not\perp^{\mathsf{nat}} .$$

Completeness of $\leq_{\mathrm{t}}$ w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$ can be proved with a technique similar to that used in Section 4.2.1 for the entailment $\leq_{\mathrm{s}}$. To our knowledge this completeness result is new.

- If $\phi \cong_{\mathrm{t}} \top^{\rho}$, then
  - $\mathrm{M}^{\phi} = \mathsf{fix}\, x^{\rho}.x$
  - $\mathrm{T}^{\phi} = \lambda z^{\rho}.\mathsf{true}$
- If $\phi = \not\perp^{\rho}$, then
  - $\mathrm{M}^{\phi} = \begin{cases} 0 & \text{if } \rho = \mathsf{nat} \\ \mathsf{true} & \text{if } \rho = \mathsf{bool} \\ \lambda x^{\rho_1}.\mathsf{fix}\, y^{\rho_2}.y & \text{if } \rho = \rho_1 \to \rho_2 \\ \langle \mathsf{fix}\, x^{\rho_1}.x, \mathsf{fix}\, y^{\rho_2}.y \rangle & \text{if } \rho = \rho_1 \times \rho_2 \end{cases}$
  - $\mathrm{T}^{\phi} = \lambda z^{\rho}.\mathsf{isdef}(z)$
- If $\phi = \phi_1 \to \phi_2 \ncong_{\mathrm{t}} \top^{\epsilon(\phi)}$, then
  - $\mathrm{M}^{\phi} = \lambda x^{\epsilon(\phi_1)}.\mathsf{if}\ \mathrm{T}^{\phi_1} x\ \mathsf{then}\ \mathrm{M}^{\phi_2}\ \mathsf{else}\ \sharp$
  - $\mathrm{T}^{\phi} = \lambda z^{\epsilon(\phi)}.\mathrm{T}^{\phi_2}(z\mathrm{M}^{\phi_1})$
- If $\phi = \phi_1 \times \phi_2 \ncong_{\mathrm{t}} \top^{\epsilon(\phi)}$, then
  - $\mathrm{M}^{\phi} = \langle \mathrm{M}^{\phi_1}, \mathrm{M}^{\phi_2} \rangle$
  - $\mathrm{T}^{\phi} = \lambda z^{\epsilon(\phi)}.\mathrm{T}^{\phi_1}(\mathsf{proj}_1 z)\ \mathsf{and}\ \mathrm{T}^{\phi_2}(\mathsf{proj}_2 z)$

Figure 4.5: Terms $\mathrm{M}^{\phi}$ and $\mathrm{T}^{\phi}$

**Theorem 4.36 (Completeness of $\leq_{\mathrm{t}}$ w.r.t. $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$)** *Let $\phi, \psi \in \mathbf{L}^{\mathrm{t}}$. $[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ implies $\phi \leq_{\mathrm{t}} \psi$.* □

To prove completeness we define for each t-type $\phi$ a term $\mathrm{M}^{\phi}$ that characterizes the property $\phi$, i.e. such that, for every t-type $\psi$, $[\mathrm{M}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ if and only if $\phi \leq_{\mathrm{t}} \psi$. We also need to define a test term $\mathrm{T}^{\phi}$ that returns $\mathsf{true}$ whenever applied to a characteristic term of a t-type which entails ($\leq_{\mathrm{t}}$) $\phi$. Such terms are defined by mutual induction on t-types.

**Definition 4.37** *Let $\phi$ be a t-type. The terms $\mathrm{M}^{\phi} \in \Lambda^{\mathsf{isdef}}_{\mathbf{T}}$ and $\mathrm{T}^{\phi} \in \Lambda^{\mathsf{isdef}}_{\mathbf{T}}$ are defined according to the clauses in Fig. 4.5, where (for sake of readability) the* $\mathsf{else}$ *branches associated to conditions that never evaluates to* $\mathsf{false}$ *(i.e. either divergent or evaluating to* $\mathsf{true}$*) are filled with the symbol $\sharp$.*

The following proposition can be proved by induction on t-types.

**Proposition 4.38** *Let $\phi$ be a t-type.*

1. $[\mathrm{M}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$.

2. $[\mathrm{T}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} \in [\![\phi \to \not\perp^{\mathsf{bool}}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}}$ *and, for every* $[M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} \in \mathbf{I}(\epsilon(\phi))^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}}$, $[\mathrm{T}^{\phi}M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} \in \{[\mathsf{fix}\, x^{\epsilon(\phi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}}, [\mathsf{true}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}}\}$. □

The proof of completeness rely on the following lemmas.

**Lemma 4.39** *Let* $\phi \in \mathbf{L}^{\mathrm{t}}$.

1. $\phi \cong_{\mathrm{t}} \top^{\epsilon(\phi)}$ *if and only if* $[\mathsf{fix}\, x^{\epsilon(\phi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} \in [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}}$.

2. *If* $\phi \cong_{\mathrm{t}} \top^{\epsilon(\phi)}$ *if and only if* $\mathrm{T}^{\phi}(\mathsf{fix}\, x^{\epsilon(\phi)}.x) \Downarrow \mathsf{true}$.

**Proof.**

1. *(only if)* Immediate.
   *(if)* Since $[\![\bot^{\epsilon(\phi)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} = \{[\mathsf{fix}\, x^{\epsilon(\phi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}}\}$ we get that (by Proposition 4.6.1) $[\mathsf{fix}\, x^{\epsilon(\phi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} \in [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}}$ imply $\phi \in \mathbf{L}^{\mathrm{st}}_{\bot\top}$. But $\phi \in \mathbf{L}^{\mathrm{t}}$, so $\phi \in \mathbf{L}^{\mathrm{t}}_{\top} = \mathbf{L}^{\mathrm{t}} \cap \mathbf{L}^{\mathrm{st}}_{\bot\top}$, and the result follows by Lemma 4.34.

2. *(only if)* Immediate, since $\phi \cong_{\mathrm{t}} \top^{\epsilon(\phi)}$ imply $\mathrm{T}^{\phi} = \lambda z^{\epsilon(\phi)}.\mathsf{true}$.
   *(if)* The case $\phi \cong_{\mathrm{t}} \top^{\epsilon(\phi)}$ is immediate. The other cases are by structural induction on $\phi \in \mathbf{L}^{\mathrm{t}}$.
   The case $\phi = \not\perp^{\epsilon(\phi)}$ is immediate.
   Let $\phi = \phi_1 \to \phi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\phi)}$, then $\phi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\phi_2)}$ (by Proposition 4.34). Since

$$[\mathrm{T}^{\phi_1\to\phi_2}(\mathsf{fix}\, x^{\epsilon(\phi_1\to\phi_2)}.x)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} = [\mathrm{T}^{\phi_2}(\mathsf{fix}\, x^{\epsilon(\phi_2)}.x)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}},$$

   the result follows by induction hypothesis.
   The case $\phi = \phi_1 \times \phi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\phi)}$ is similar. □

**Lemma 4.40** *Let* $\phi, \psi \in \mathbf{L}^{\mathrm{t}}(\rho)$.

1. *If* $[\mathrm{M}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} \in [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}}$ *then* $\phi \leq_{\mathrm{t}} \psi$.

2. *If* $\mathrm{T}^{\psi}\mathrm{M}^{\phi} \Downarrow \mathsf{true}$ *then* $\phi \leq_{\mathrm{t}} \psi$.

**Proof.** We first consider the case in which either $\psi \cong_{\mathrm{t}} \top^{\epsilon(\psi)}$ or $\phi \cong_{\mathrm{t}} \top^{\epsilon(\phi)}$.
Let $\underline{\psi \cong_{\mathrm{t}} \top^{\epsilon(\psi)}}$. Both point 1. and point 2. are immediate.
Let $\underline{\phi \cong_{\mathrm{t}} \top^{\epsilon(\phi)}}$. The case $\psi \cong_{\mathrm{t}} \top^{\epsilon(\psi)}$ has been already considered, so let $\psi \not\cong_{\mathrm{t}} \top^{\epsilon(\psi)}$. Both point 1. and point 2. are obvious since (by Lemma 4.39) $[\mathrm{M}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} = [\mathsf{fix}\, x^{\epsilon(\phi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} \notin [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}}$ and $\mathrm{T}^{\psi}(\mathsf{fix}\, x^{\epsilon(\phi)}.x) \Uparrow$.
If $\psi \not\cong_{\mathrm{t}} \top^{\epsilon(\psi)}$ and $\phi \not\cong_{\mathrm{t}} \top^{\epsilon(\phi)}$ the proof is by structural induction on $\psi$ and $\phi$.
Let $\underline{\psi = \not\perp^{\epsilon(\psi)}}$. The case $\phi \cong_{\mathrm{t}} \top^{\epsilon(\phi)}$ has been already considered, and the case $\phi = \not\perp^{\epsilon(\phi)}$ is trivial.
If $\phi = \phi_1 \to \phi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\phi)}$ then $\phi \leq_{\mathrm{t}} \psi$. Both point 1. and point 2. are obvious since $\mathrm{M}^{\phi_1\to\phi_2}$ is a $\lambda$-abstraction, so $[\mathrm{M}^{\phi_1\to\phi_2}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}} \in [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathrm{isdef}}}$ and $\mathrm{T}^{\psi}\mathrm{M}^{\phi_1\to\phi_2} = (\lambda z^{\epsilon(\psi)}.\mathsf{isdef}(z))\mathrm{M}^{\phi_1\to\phi_2} \Downarrow$

$\mathsf{true}$.
The case $\phi = \phi_1 \times \phi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\phi)}$ is analogous.

Let $\underline{\phi = \not\perp^{\epsilon(\phi)}}$. The case $\psi \cong_{\mathrm{t}} \top^{\epsilon(\psi)}$ and the case $\psi = \not\perp^{\epsilon(\psi)}$ have been already considered.
If $\psi = \psi_1 \to \psi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\psi)}$ then $\phi \not\leq_{\mathrm{t}} \psi$. Point 1. is obvious since $[\mathrm{M}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathrm{C}^{\epsilon(\phi)}_{\not\perp}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \notin [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$. For point 2. note that $[\mathrm{T}^{\psi_1\to\psi_2}(\mathrm{M}^{\phi})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathrm{T}^{\psi_2}(\mathsf{fix}\, y^{\epsilon(\psi_2)}.y)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, and (by Lemma 4.39.2, since $\psi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\psi_2)}$) $\mathrm{T}^{\psi_2}(\mathsf{fix}\, y^{\epsilon(\psi_2)}.y) \Uparrow$.
The case $\psi = \psi_1 \times \psi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\psi)}$ is analogous.

Let $\underline{\psi = \psi_1 \to \psi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\psi)}}$. All the cases but $\phi = \phi_1 \to \phi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\phi)}$ have been already considered. Then 1. and 2. can be proved as follows.

1. $[\mathrm{M}^{\phi_1\to\phi_2}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\psi_1 \to \psi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ implies $[\mathrm{M}^{\phi_1\to\phi_2}\mathrm{M}^{\psi_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\psi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$. But

$$[\mathrm{M}^{\phi_1\to\phi_2}\mathrm{M}^{\psi_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathsf{if}\ \mathrm{T}^{\phi_1}\mathrm{M}^{\psi_1}\ \mathsf{then}\ \mathrm{M}^{\phi_2}\ \mathsf{else}\ \sharp]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}},$$

   and (by Lemma 4.39.1, since $\psi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\psi_2)}$) we have that both $\mathrm{T}^{\phi_1}\mathrm{M}^{\psi_1} \Downarrow \mathsf{true}$ and $[\mathrm{M}^{\phi_2}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\psi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$. So (by induction hypotheses) $\psi_1 \leq_{\mathrm{t}} \phi_1$ and $\phi_2 \leq_{\mathrm{t}} \psi_2$ which imply $\phi_1 \to \phi_2 \leq_{\mathrm{t}} \psi_1 \to \psi_2$.

2. Assume $\mathrm{T}^{\psi_1\to\psi_2}\mathrm{M}^{\phi_1\to\phi_2} \Downarrow \mathsf{true}$. Since

$$[\mathrm{T}^{\psi_1\to\psi_2}\mathrm{M}^{\phi_1\to\phi_2}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathrm{T}^{\psi_2}(\mathrm{M}^{\phi_1\to\phi_2}\mathrm{M}^{\psi_1})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\mathrm{T}^{\psi_2}(\mathsf{if}\ \mathrm{T}^{\phi_1}\mathrm{M}^{\psi_1}\ \mathsf{then}\ \mathrm{M}^{\phi_2}\ \mathsf{else}\ \sharp)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}},$$

   and (by Lemma 4.39.2, since $\psi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\psi_2)}$) we have that both $\mathrm{T}^{\phi_1}\mathrm{M}^{\psi_1} \Downarrow \mathsf{true}$ and $\mathrm{T}^{\psi_2}\mathrm{M}^{\phi_1} \Downarrow \mathsf{true}$. So (by induction hypotheses) $\psi_1 \leq_{\mathrm{t}} \phi_1$ and $\phi_2 \leq_{\mathrm{t}} \psi_2$, which imply $\phi_1 \to \phi_2 \leq_{\mathrm{t}} \psi_1 \to \psi_2$.

The proof of the only remaining case, $\underline{\psi = \psi_1 \times \psi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\psi)}}$ and $\phi = \phi_1 \times \phi_2 \not\cong_{\mathrm{t}} \top^{\epsilon(\phi)}$, is similar to the proof of the previous one. □

From the previous lemma we derive the completeness of the entailment relation.
**Proof of Theorem 4.36.** By contraposition. Assume that $\phi \not\leq_{\mathrm{t}} \psi$. Then $\psi \not\cong_{\mathrm{t}} \top^{\epsilon(\psi)}$. By Lemma 4.40.1, $[\mathrm{M}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \notin [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$. Since from Proposition 4.38.1 $[P]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, this implies that $[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \not\subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$. □

**Remark 4.41** *As showed in Example 4.35 the entailment relation $\leq_{\mathrm{t}}$ is not complete w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$. At the moment we do not know whether a sound (and decidable) extension of the $\leq_{\mathrm{t}}$ relation which is complete w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$ exists.*

### 4.3.2 An assignment system for totality types

For each constant $\mathsf{c}$ a set $\mathbf{L}^{\mathrm{t}}(\mathsf{c})$ of minimal t-types is specified. The idea is that every t-type which is entailed ($\leq_{\mathrm{t}}$) by an element of $\mathbf{L}^{\mathrm{t}}(\mathsf{c})$ can be assigned to $\mathsf{c}$. For instance, for all integers $\mathsf{n}$, $\mathbf{L}^{\mathrm{t}}(\mathsf{n}) = \{\not\perp^{\mathsf{nat}}\}$ and $\mathbf{L}^{\mathrm{t}}(+) = \{\not\perp^{\mathsf{nat}} \times \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}\}$.

$$(\leq)\ \frac{\Sigma \vdash M : \phi \quad \phi \leq \psi}{\Sigma \vdash M : \psi}$$

$$(\text{Var})\ \Sigma, x : \phi \vdash x : \phi \qquad (\text{Con})\ \frac{\phi \in \mathbf{L}^{\mathrm{t}}(\mathsf{c})}{\Sigma \vdash \mathsf{c} : \phi}$$

$$(\to \text{I})\ \frac{\Sigma, x : \phi \vdash M : \psi}{\Sigma \vdash \lambda x.M : \phi \to \psi} \qquad (\to \text{I}_{\not\downarrow})\ \frac{\vdash_{\mathbf{T}} \lambda x.M : \rho}{\Sigma \vdash \lambda x.M : \not\downarrow^{\rho}}$$

$$(\to \text{E})\ \frac{\Sigma \vdash M : \phi \to \psi \quad \Sigma \vdash N : \phi}{\Sigma \vdash MN : \psi}$$

$$(\times \text{I})\ \frac{\Sigma \vdash M_1 : \phi_1 \quad \Sigma \vdash M_2 : \phi_2}{\Sigma \vdash \langle M_1, M_2 \rangle : \phi_1 \times \phi_2} \qquad (\times \text{I}_{\not\downarrow})\ \frac{\vdash_{\mathbf{T}} \langle M_1, M_2 \rangle : \rho}{\Sigma \vdash \langle M_1, M_2 \rangle : \not\downarrow^{\rho}}$$

$$(\times \text{E}_i)\ \frac{\Sigma \vdash M : \phi_1 \times \phi_2}{\Sigma \vdash \mathsf{proj}_i M : \phi_i}\ i \in \{1, 2\}$$

$$(\text{Fix}_{\not\downarrow})\ \frac{\Sigma, x : \top^{\epsilon(\phi_1)} \vdash M : \phi_1 \quad \cdots \quad \Sigma, x : \phi_{n-1} \vdash M : \phi_n}{\Sigma \vdash \mathsf{fix}\, x.M : \phi_n}\ n \geq 1$$

$$(\text{If}_{\not\downarrow})\ \frac{\Sigma \vdash N : \not\downarrow^{\mathsf{bool}} \quad \Sigma \vdash M_1 : \phi \quad \Sigma \vdash M_2 : \phi}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \phi}$$

$$(\text{If}_{\top})\ \frac{\Sigma \vdash N : \top^{\mathsf{bool}} \quad \vdash_{\mathbf{T}} M_1 : \rho \quad \vdash_{\mathbf{T}} M_2 : \rho}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \top^{\rho}}$$

Figure 4.6: Rules for totality type assignment (system $\vdash_{\mathrm{t}}$)

**Definition 4.42 (Totality type assignment system)** *We write* $\Sigma \vdash_{\mathrm{t}} M : \phi$ *to mean that* $\Sigma \vdash M : \phi$ *can be derived by the rules in Fig. 4.6.*

Note that, being $\vdash_{\mathrm{t}}$ an inference system, the same terms can have different t-types.

**Example 4.43** *Let* $\vdash_{\mathbf{T}} \mathrm{twice} : \rho$ *where* $\rho = (\sigma \to \sigma) \to \sigma \to \sigma$ *and* $\mathrm{twice} = \lambda f^{\sigma\to\sigma}.\ \lambda x^{\sigma}.\ f(f\,x)$*. It is easy to check that, for* $i \in \{3, 4\}$, $\vdash \mathrm{twice} : \phi_i$*, where* $\phi_3 = (\not\perp^{\sigma} \to \not\perp^{\sigma}) \to \not\perp^{\sigma} \to \not\perp^{\sigma}$*, and* $\phi_4 = (\top^{\sigma} \to \not\perp^{\sigma}) \to \top^{\sigma} \to \not\perp^{\sigma}$*.*

As for the system $\vdash_{\mathrm{s}}$ we can show the following fact.

**Fact 4.44** *1.* $\Sigma \vdash_{\mathrm{t}} M : \phi$ *implies* $\vdash_{\mathbf{T}} M : \epsilon(\phi)$ *and* $\epsilon(\Sigma) \supseteq \mathrm{FV}(M)$*.*

*2.* $\vdash_{\mathbf{T}} M : \rho$ *implies, for all basis* $\Sigma$ *such that* $\Sigma = \{x^{\epsilon(\psi)} : \psi \mid x^{\epsilon(\psi)} \in \mathrm{FV}(M)$ *for some* $\psi \in \mathbf{L}^{\mathrm{t}}\}$*,* $\Sigma \vdash_{\mathrm{t}} M : \top^{\rho}$*.* □

The following example[1] shows an application of rule $(\mathrm{Fix}_{\not\perp})$.

**Example 4.45** *Let* $\vdash_{\mathbf{T}} \mathrm{pair} : \mathsf{nat} \to \mathsf{nat} \to \rho$*,* $\vdash_{\mathbf{T}} \mathrm{fst} : \rho \to \mathsf{nat}$*,* $\vdash_{\mathbf{T}} \mathrm{snd} : \rho \to \mathsf{nat}$*, and* $\vdash_{\mathbf{T}} \mathsf{fix}\, p^{\rho}.M : \rho$*, where*

- $\rho = (\mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}) \to \mathsf{nat}$*,*
- $\mathrm{pair} = \lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\lambda z^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.zxy$*,*
- $\mathrm{fst} = \lambda p^{\rho}.p(\lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.x)$*,*
- $\mathrm{snd} = \lambda p^{\rho}.p(\lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.y)$*, and*
- $M = \mathrm{pair}(\mathrm{snd}\, p)3$*.*

*Let* $\phi_1 = (\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \not\perp^{\mathsf{nat}}$*, and* $\phi_2 = (\not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \not\perp^{\mathsf{nat}}$*.*
*It is easy to check that:* $\{p : \top^{\rho}\} \vdash_{\mathrm{t}} M : \phi_1$ *and* $\{p : \phi_1\} \vdash_{\mathrm{t}} M : \phi_2$*. So, by rule* $(\mathrm{Fix}_{\not\perp})$*, we have:* $\emptyset \vdash_{\mathrm{t}} \mathsf{fix}\, p^{\rho}.M : \phi_2$*.*

*Note that, to assign* $\phi_2$ *to* $M$*, two premises are needed, i.e. it is not possible to derive* $\{p : \top^{\rho}\} \vdash_{\mathrm{t}} M : \phi_2$*.*

Soundness of the t-type assignment system (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$) is stated by the following theorem.

**Theorem 4.46 (Soundness of $\vdash_{\mathrm{t}}$)** *Let* $\Sigma \vdash_{\mathrm{t}} M : \phi$*. Then* $[\![M]\!]_e \in [\![\phi]\!]$ *for all* $e$ *such that for all* $x : \psi \in \Sigma$*,* $e(x) \in [\![\psi]\!]$*.*

**Proof.** By induction on the derivation of $\Sigma \vdash_{\mathrm{t}} M : \phi$. □

[1] This example is used in [Solberg, 1995a] page 79 to illustrate the rule (fix) of the system for combined strictness and totality type inference $\vdash_{\mathrm{ST}}$ (discussed in Section 4.4.4 of this thesis).

### 4.3.3 Totality analysis and ground contextual equivalence

Ground contextual equivalence, $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$, is such that, for all PCFP terms $M$ of functional type (say $\rho_1 \to \rho_2$), $M \simeq^{\mathrm{gnd}}_{\mathrm{obs}} \lambda x^{\rho_1}.Mx$ (provided that $x^{\rho_1} \notin \mathrm{FV}(M)$). Consequently, under the $\mathcal{M}^{\mathrm{gnd}}$ semantics of t-types (defined according to clauses of Definition 4.2), there is no totality type which characterizes the terms having w.h.n.f.. In particular $\not\perp^{\rho}$ is interpreted as the set of $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$ equivalence classes of terms $M$ such that for *some* $N_1, \ldots, N_p$ ($p \geq 0$) $MN_1 \cdots N_p \Downarrow$ to a ground value (in $\mathsf{nat}$ or $\mathsf{bool}$). This is a more restrictive property, with a much more extensional nature than the interpretation in $\mathcal{M}^{\mathrm{lazy}}$ given in Definition 4.2. In particular a totality type assignment system which is sound for the interpretation of t-types in $\mathcal{M}^{\mathrm{gnd}}$ cannot have the rules ($\to \mathrm{I}_{\not\perp}$) and ($\times \mathrm{I}_{\not\perp}$) in Fig. 4.6, and the lack of these rules decreases significantly the set of properties that can be assigned to terms. The property of "having a w.h.n.f." is instead much more effective and more suitable for applications in program optimization.

Note that the totality analysis based on abstract interpretation in [Abramsky, 1990a] and the comportment analysis (that includes totality analysis) in [Mycroft and Solberg, 1995] (see also [Solberg, 1995a] Chapter 6), are both based on the ground observational equivalence.

## 4.4 Combined strictness and totality analysis

In this section we return to the language of strictness and totality types $\mathbf{L}^{\mathrm{st}}$ introduced in Section 4.1 and consider the problem of defining entailment rules and assignment rules for the combined language.

The aim of this section is to show that, even if we spend some effort in defining an assignment system for a language of combined strictness and totality properties (obtaining a new system which extends the one presented in [Solberg *et al.*, 1994; Solberg *et al.*, ] and [Solberg, 1995a] Chapter 2), the weakness of type assignment rules does not allow to use in an essential way the (greater) expressive power of the combined language of properties.

As an example, if we prove that a term $M$ has st-type $(\not\perp^{\rho_1} \to \not\perp^{\rho_2}) \to \perp^{\sigma} \to \perp^{\tau}$ we have that $M$, seen as a function of two arguments, is strict in its second argument if we can prove that the first argument is a total function. We will see, however, that this implies that $M$ has also st-type $\top^{\rho_1 \to \rho_2} \to \perp^{\sigma} \to \perp^{\tau}$, i.e. it is strict in its second argument independently of the first one.

### 4.4.1 An entailment relation for strictness and totality types

A simple relation of entailment for st-types, $\leq_{\mathrm{st}}$, is defined by taking the rules in Fig. 4.1 (or 4.4), by removing the axiom ($\perp$) (resp. ($\not\perp$)) and ($\top$), and by adding the following 3 axioms

$$(\perp') \ \frac{\phi \in \mathbf{L}^{\mathrm{st}}_{\perp\top}}{\perp^{\epsilon(\phi)} \leq_{\mathrm{st}} \phi} \ , \qquad (\not\perp') \ \frac{\phi \in \mathbf{L}^{\mathrm{st}} - \mathbf{L}^{\mathrm{st}}_{\perp\top}}{\phi \leq_{\mathrm{st}} \not\perp^{\epsilon(\phi)}} \ , \qquad (\top') \ \frac{\psi \in \mathbf{L}^{\mathrm{st}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq_{\mathrm{st}} \psi} \ .$$

As for the entailment $\leq_{\mathrm{s}}$ and $\leq_{\mathrm{t}}$, the entailment relation $\leq_{\mathrm{st}}$ is reflexive and transitive, since rule (Trans) in Fig. 3.4 is admissible. Let $\cong_{\mathrm{st}}$ denote the equivalence relation induced by $\leq_{\mathrm{st}}$. The soundness of $\leq_{\mathrm{st}}$ (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$) can be proved by induction on derivations.

**Theorem 4.47 (Soundness of $\leq_{\mathrm{st}}$)** *Let $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{st}}$. $\phi_1 \leq_{\mathrm{st}} \phi_2$ implies $[\![\phi_1]\!] \subseteq [\![\phi_2]\!]$.* □

Also in this case the entailment rules are not complete w.r.t. $\mathcal{M}^{\mathrm{lazy}}$. The argument that shows the failure of completeness for the entailment for strictness (see Example 4.10) and for the one for totality (see Example 4.35), in fact, can be applied to the combined entailment as well. However, $\leq_{\mathrm{st}}$ is not complete even w.r.t. $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$. There are in fact other reasons for which completeness fails, due to the interaction of strictness and totality properties. One of these reasons is that the entailment rules do not take into account the monotonicity of the evaluation rules, as the following example shows.

**Example 4.48 (Incompleteness of $\leq_{\mathrm{st}}$ w.r.t. $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$)** *Remember that, for every $\rho \in \mathbf{T}$, $[\![\top^\rho]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\![\bot^\rho]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \cup [\![\not\bot^\rho]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, $[\![\bot^\rho]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = \{[\mathsf{fix}\, x^\rho.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}\}$, and $[\![\not\bot^\rho]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = \mathbf{I}(\rho)^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} - \{[\mathsf{fix}\, x^\rho.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}\}$.*

1. *Take $\phi = \not\bot^\rho \to \bot^\sigma$ and $\psi = \top^\rho \to \bot^\sigma$. Since any monotone function that maps all the non-bottom elements in bottom* must *map bottom to bottom, we have that $[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$. However, $\phi \not\leq_{\mathrm{st}} \psi$.*

2. *Take also $\phi = \bot^\rho \to \not\bot^\sigma$ and $\psi = \top^\rho \to \not\bot^\sigma$. Since any monotone function that maps the bottom element to a value different from bottom* must *map any other value in a value different from bottom, we have that $[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$. However $\phi \not\leq_{\mathrm{st}} \psi$.*

### 4.4.2 A stronger entailment relation for strictness and totality types

In this section, to be able to handle the weakness of the system $\leq_{\mathrm{st}}$ shown in Example 4.48, we define a stronger entailment relation for combined strictness and totality types. We obtain it by adding new rules for proving entailments between "arrow" st-types that take in account monotonicity and continuity properties of the model $\mathcal{M}^{\mathrm{lazy}}$. These rules have a more complex structure than those defining $\leq_{\mathrm{st}}$.

To avoid useless details we will restrict to the language PCF′ (see Section 2.5), the subset of PCFP without the type operator "times", which is also the language considered in [Solberg *et al.*, 1994; Solberg *et al.*, ; Solberg, 1995a].

The new entailment relation ($\leq'_{\mathrm{st}}$ in Definition 4.57) is stronger than the one proposed in *op. cit.* and correct some drawbacks of the original system (see Section 4.4.4).

Let $\mathbf{L}'^{\mathrm{st}}$ be the subset of the st-types for PCF′, i.e. the set of the st-types that does not contain the type constructor $\times$. For $a \in \{\top, \bot, \not\bot\}$, let $\mathbf{L}'^{\mathrm{st}}_a = \mathbf{L}^{\mathrm{st}}_a \cap \mathbf{L}'^{\mathrm{st}}$, where $\mathbf{L}^{\mathrm{st}}_\top$, $\mathbf{L}^{\mathrm{st}}_\bot$, and $\mathbf{L}^{\mathrm{st}}_{\not\bot}$ are the sets introduced in Definition 4.3. Note that

$$\mathbf{L}'^{\mathrm{st}}_a = \{\phi_1 \to \cdots \to \phi_n \to a^\rho \mid n \geq 0, \phi_1, \ldots, \phi_n \in \mathbf{L}'^{\mathrm{st}}, \text{ and } \rho \in \mathbf{T}\}.$$

Moreover $\mathbf{L}^{\prime\mathrm{st}}_{\top}$, $\mathbf{L}^{\prime\mathrm{st}}_{\bot}$, and $\mathbf{L}^{\prime\mathrm{st}}_{\not\bot}$ form a partition of $\mathbf{L}^{\prime\mathrm{st}}$, i.e. they are pairwise disjoint and $\mathbf{L}^{\prime\mathrm{st}} = \mathbf{L}^{\prime\mathrm{st}}_{\top} \cup \mathbf{L}^{\prime\mathrm{st}}_{\bot} \cup \mathbf{L}^{\prime\mathrm{st}}_{\not\bot}$.

**Definition 4.49** *Let* $\mathcal{M} \in \{\mathcal{M}^{\mathrm{lazy}}, \mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}, \mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}\}$ *and* $\mathrm{A} \subseteq \mathbf{I}(\rho)^{\mathcal{M}}$. *The* downward closure *of* A *is the set*

$$\hat{\mathrm{A}} = \{[M]^{\mathcal{M}} \mid M \preceq^{\mathrm{lazy}}_{\mathrm{obs}} N \text{ for some } [N]^{\mathcal{M}} \in \mathrm{A}\},$$

*and the* upward closure *is the set*

$$\breve{\mathrm{A}} = \{[M]^{\mathcal{M}} \mid N \preceq^{\mathrm{lazy}}_{\mathrm{obs}} M \text{ for some } [N]^{\mathcal{M}} \in \mathrm{A}\}.$$

Obviously $\mathrm{A} \subseteq \hat{\mathrm{A}}$ and $\mathrm{A} \subseteq \breve{\mathrm{A}}$.

**Proposition 4.50** *1.* $\phi \in \mathbf{L}^{\prime\mathrm{st}}_{\top}$ *implies* $[\![\phi]\!] = [\![\hat{\phi}]\!] = [\![\breve{\phi}]\!]$,

*2.* $\phi \in \mathbf{L}^{\prime\mathrm{st}}_{\bot}$ *implies* $[\![\phi]\!] = [\![\hat{\phi}]\!]$, *and*

*3.* $\phi \in \mathbf{L}^{\prime\mathrm{st}}_{\not\bot}$ *implies* $[\![\phi]\!] = [\![\breve{\phi}]\!]$.

**Proof.**

1. Obvious, since $\phi \in \mathbf{L}^{\prime\mathrm{st}}_{\top}$ implies $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$.

2. Let $\phi = \phi_1 \to \cdots \to \phi_n \to \bot^{\rho} \in \mathbf{L}^{\prime\mathrm{st}}_{\bot}$ $(n \geq 0)$. Assume that $M' \preceq^{\mathrm{lazy}}_{\mathrm{obs}} M$ for some $[M] \in [\![\phi]\!]$ and $[M'] \notin [\![\phi]\!]$. Then there are $[N_i] \in [\![\phi_i]\!]$, $1 \leq i \leq n$, such that $[M'\, N_1 \cdots N_n] \notin [\![\bot^{\rho}]\!]$. However this implies that $M'\, N_1 \cdots N_n \Downarrow$ whereas $M\, N_1 \cdots N_n \Uparrow$, contradicting the fact that $M' \preceq^{\mathrm{lazy}}_{\mathrm{obs}} M$.

3. Similar to the proof of 2. □

We now introduce, as in [Solberg *et al.*, 1994; Solberg *et al.*, ; Solberg, 1995a], two *syntactic* operations on st-types: up and dn, that capture at a syntactic level the upward and downward closure.

**Remark 4.51** *The operations* up *and* dn *are the operations* $\uparrow$ *and* $\downarrow$ *introduced in [Solberg* et al.*, 1994; Solberg* et al.*, ] and [Solberg, 1995a] Chapter 2. We use new names (*dn *and* up*) for this operations to avoid any confusion with the* non-standard type constructors $\uparrow$ *and* $\downarrow$ *of Definition 5.54.*

**Definition 4.52 (**up **and** dn**)** *The functions* up : $\mathbf{L}^{\prime\mathrm{st}} \to (\mathbf{L}^{\prime\mathrm{st}}_{\top} \cup \mathbf{L}^{\prime\mathrm{st}}_{\not\bot})$ *and* dn : $\mathbf{L}^{\prime\mathrm{st}} \to (\mathbf{L}^{\prime\mathrm{st}}_{\top} \cup \mathbf{L}^{\prime\mathrm{st}}_{\bot})$ *are defined by the clauses in Fig. 4.7.*

| | | | | | |
|---|---|---|---|---|---|
| $\mathrm{up}(\bot^\rho)$ | $=$ | $\top^\rho$ | $\mathrm{dn}(\bot^\rho)$ | $=$ | $\bot^\rho$ |
| $\mathrm{up}(\not\bot^\rho)$ | $=$ | $\not\bot^\rho$ | $\mathrm{dn}(\not\bot^\rho)$ | $=$ | $\top^\rho$ |
| $\mathrm{up}(\top^\rho)$ | $=$ | $\top^\rho$ | $\mathrm{dn}(\top^\rho)$ | $=$ | $\top^\rho$ |
| $\mathrm{up}(\phi \to \psi)$ | $=$ | $\phi \to \mathrm{up}(\psi)$ | $\mathrm{dn}(\phi \to \psi)$ | $=$ | $\phi \to \mathrm{dn}(\psi)$ |

Figure 4.7: Functions up and dn

For instance $\mathrm{up}(\bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) = \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}}$, and $\mathrm{dn}(\not\bot^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}}) = \not\bot^{\mathsf{nat}} \to \top^{\mathsf{nat}}$.

The operation up behaves as the identity on $\mathbf{L}'^{\mathrm{st}}_\top \cup \mathbf{L}'^{\mathrm{st}}_{\not\bot}$ and maps $\mathbf{L}'^{\mathrm{st}}_\bot$ to $\mathbf{L}'^{\mathrm{st}}_\top$. For every st-type in $\mathbf{L}'^{\mathrm{st}}$, the operation up corresponds exactly to the upward closure (see Lemma 4.54.1).

Dually, dn behaves as the identity on $\mathbf{L}'^{\mathrm{st}}_\top \cup \mathbf{L}'^{\mathrm{st}}_\bot$ and maps $\mathbf{L}'^{\mathrm{st}}_{\not\bot}$ to $\mathbf{L}'^{\mathrm{st}}_\top$. Let $\mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$ be the subset of $\mathbf{L}'^{\mathrm{st}}_{\not\bot}$ defined by $\mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot} = \{\not\bot^\rho \mid \rho \in \mathbf{T}\} \cup \mathbf{L}'^{\mathrm{st}\to}_{\not\bot}$, where

$$\mathbf{L}'^{\mathrm{st}\to}_{\not\bot} = \{\phi_1 \to \cdots \to \phi_n \to \not\bot^{\rho\to\sigma} \mid \rho \to \sigma \in \mathbf{T}, n \geq 1, \text{ and } \phi_1, \ldots, \phi_n \in \mathbf{L}'^{\mathrm{st}}\}.$$

Note that

$$\mathbf{L}'^{\mathrm{st}}_{\not\bot} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot} = \{\phi_1 \to \cdots \to \phi_n \to \not\bot^\iota \mid \iota \in \{\mathsf{nat}, \mathsf{bool}\}, n \geq 1, \text{ and } \phi_1, \ldots, \phi_n \in \mathbf{L}'^{\mathrm{st}}\}.$$

As we will see in the following (Lemma 4.54.2), for the st-types in $\mathbf{L}'^{\mathrm{st}}_\top \cup \mathbf{L}'^{\mathrm{st}}_{\not\bot} \cup \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$, the operation dn corresponds to the downward closure. While, for the st-types $\phi \in \mathbf{L}'^{\mathrm{st}}_{\not\bot} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$, $\mathrm{dn}(\phi)$ is the smallest (w.r.t. inclusion of interpretations) st-type whose interpretation contains $[\![\hat{\phi}]\!]$ (see Lemma 4.54.3). Indeed, our st-types are not enough expressive to capture exactly the downward closure of the st-types $\in \mathbf{L}'^{\mathrm{st}}_{\not\bot} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$, as the following example shows.

**Example 4.53** *Consider the st-type $\phi = \top^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}}$ ($\phi \in \mathbf{L}'^{\mathrm{st}}_{\not\bot} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$), which characterizes the constant functions of type* $\mathsf{nat} \to \mathsf{nat}$. *We have* $\mathrm{dn}(\top^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}}) = \top^{\mathsf{nat}} \to \top^{\mathsf{nat}}$, *and so* $[\![\mathrm{dn}(\top^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}})]\!] = \mathbf{I}(\mathsf{nat} \to \mathsf{nat})$, *whereas the downward closure of* $[\![\top^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}}]\!]$ *is the set of equivalence classes of the closed terms which are less defined than constant functions. As we will see (in Lemma 4.54.3) for no st-type $\psi$, $[\![\hat{\phi}]\!] = [\![\psi]\!]$.*

**Lemma 4.54** *Let $\phi \in \mathbf{L}'^{\mathrm{st}}$. Then $[\![\phi]\!] \subseteq [\![\mathrm{up}(\phi)]\!]$ and $[\![\phi]\!] \subseteq [\![\mathrm{dn}(\phi)]\!]$. More specifically*

1. $[\![\breve{\phi}]\!] = [\![\mathrm{up}(\phi)]\!]$,
2. $\phi \in \mathbf{L}'^{\mathrm{st}}_\top \cup \mathbf{L}'^{\mathrm{st}}_\bot \cup \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$ *implies* $[\![\hat{\phi}]\!] = [\![\mathrm{dn}(\phi)]\!]$, *and*
3. $\phi \in \mathbf{L}'^{\mathrm{st}}_{\not\bot} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$ *implies* $[\![\mathrm{dn}(\phi)]\!] = [\![\top^{\epsilon(\phi)}]\!]$ *and* $\{\psi \mid [\![\hat{\phi}]\!] \subseteq [\![\psi]\!]\} = \mathbf{L}'^{\mathrm{st}}_\top$.

**Proof.**

1. The result follows from Proposition 4.50 and the fact that, for $\phi \in \mathbf{L}'^{\mathrm{st}}_\bot$, $[\![\breve{\phi}]\!] = [\![\top^{\epsilon(\phi)}]\!]$. To see that $[\![\breve{\phi}]\!] = [\![\top^{\epsilon(\phi)}]\!]$, observe that $[\mathsf{fix}\, x^{\epsilon(\phi)}.x] \in [\![\phi]\!]$ and, for all $[M] \in [\![\top^{\epsilon(\phi)}]\!]$, $\mathsf{fix}\, x^{\epsilon(\phi)}.x \preceq^{\mathrm{lazy}}_{\mathrm{obs}} M$.

2. The case $\phi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\bot}$ follows from Proposition 4.50. For the case $\phi \in \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$, notice that $\mathrm{dn}(\phi)$ is an $\top$-st-type.
   Let $\phi \in \mathbf{L}'^{\mathrm{st}\rightarrow}_{\not\bot}$, i.e. $\phi = \phi_1 \to \cdots \to \phi_n \to \not\bot^{\sigma\to\tau}$ $(n \geq 1)$. For all $[M] \in [\![\top^{\epsilon(\phi)}]\!]$, consider $N = \lambda\, x_1^{\epsilon(\phi_1)}. \cdots \lambda\, x_n^{\epsilon(\phi_n)}. \lambda\, y^{\sigma}. M\, x_1 \cdots x_n\, y$. It is easy to check that $M \preceq^{\mathrm{lazy}}_{\mathrm{obs}} N$ and $[N] \in [\![\phi]\!]$. Therefore $[\![\hat{\phi}]\!] = [\![\top^{\epsilon(\phi)}]\!]$.
   The case $\phi \in \{\not\bot^{\rho} \mid \rho \in \mathbf{T}\}$ is trivial.

3. Let $\phi \in \mathbf{L}'^{\mathrm{st}}_{\not\bot} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$, i.e. $\phi = \phi_1 \to \cdots \to \phi_n \to \not\bot^{\iota}$ $(n \geq 1)$. Let $\psi$ be such that $[\![\hat{\phi}]\!] \subseteq [\![\psi]\!]$, then $\epsilon(\phi) = \epsilon(\psi)$ and $\psi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\bot}$ (since $[\![\psi]\!]$ must include $[\![\bot^{\epsilon(\phi)}]\!]$). Assume that $\psi \in \mathbf{L}'^{\mathrm{st}}_{\bot}$, then it is of the shape $\psi_1 \to \cdots \to \psi_n \to \bot^{\rho_1 \to \cdots \rho_m \to \iota}$ where $\epsilon(\psi) = \epsilon(\psi_1) \to \cdots \to \epsilon(\psi_n) \to \rho_1 \to \cdots \to \rho_m \to \iota$, $n + m \geq 1$, and $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$. So, for $M = \lambda\, x_1^{\epsilon(\psi_1)}. \cdots \lambda\, x_n^{\epsilon(\psi_n)}. \lambda\, y_1^{\rho_1}. \cdots \lambda\, y_m^{\rho_m}. \mathsf{c}$, where $\mathsf{c}$ is a constant of type $\iota$, we have $[M] \in [\![\phi]\!]$ and $[M] \notin [\![\psi]\!]$, which is absurd. Therefore we can conclude that $\psi \in \mathbf{L}'^{\mathrm{st}}_{\top}$. This proves the result. □

For the set, $\mathbf{L}'^{\mathrm{st}}_{\not\bot} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$, of the st-types for which dn does not coincide with the downward closure we define an operation that approximates the downward closure. The underlying idea is that, since terms denote monotone functions, terms that converge on some arguments are more defined than terms that always diverge.

**Definition 4.55** (bw) *The function* $\mathrm{bw} : (\mathbf{L}'^{\mathrm{st}}_{\not\bot} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}) \to \mathbf{L}'^{\mathrm{st}}_{\bot}$ *is defined by:*

$$\mathrm{bw}(\phi_1 \to \cdots \to \phi_n \to \not\bot^{\iota}) = \top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_n)} \to \bot^{\iota} \ .$$

For instance, for $\phi = \bot^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}}$, $\mathrm{dn}(\phi) = \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}}$, whereas $\mathrm{bw}(\phi) = \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$.

**Lemma 4.56** *Let* $\phi \in \mathbf{L}'^{\mathrm{st}}_{\not\bot} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}$. *Then* $[\![\mathrm{bw}(\phi)]\!] \cap [\![\phi]\!] = \emptyset$ *and* $[\![\mathrm{bw}(\phi)]\!] \subset [\![\hat{\phi}]\!]$.

**Proof.** $[\![\mathrm{bw}(\phi)]\!] \cap [\![\phi]\!] = \emptyset$ is immediate. Let $\phi = \phi_1 \to \cdots \to \phi_n \to \not\bot^{\iota}$. For every $[M] \in [\![\mathrm{bw}(\phi)]\!]$, $M$ is a closed term that, whenever applied to $n$ arguments, result in a divergent computation. Consider $M' = \lambda\, x_1^{\epsilon(\phi_1)}. \cdots \lambda\, x_n^{\epsilon(\phi_n)}. \mathsf{c}$, where $\mathsf{c}$ is any constant of type $\iota$. We have that $[M'] \in [\![\phi]\!]$ and $M \preceq^{\mathrm{lazy}}_{\mathrm{obs}} M'$. Therefore $[\![\mathrm{bw}(\phi)]\!] \subseteq [\![\hat{\phi}]\!]$. We conclude that $[\![\mathrm{bw}(\phi)]\!] \subset [\![\hat{\phi}]\!]$. □

We now introduce the new entailment relation between st-types, denoted by $\leq'_{\mathrm{st}}$.

**Definition 4.57 (Entailment relation $\leq'_{\mathrm{st}}$)** *Let* $\phi, \psi \in \mathbf{L}'^{\mathrm{st}}$. *We write* $\phi \leq'_{\mathrm{st}} \psi$ *to mean that* $\phi \leq \psi$ *is derivable by the rules in Fig. 4.8. By* $\cong'_{\mathrm{st}}$ *we denote the equivalence relation induced by* $\leq'_{\mathrm{st}}$.

For every $\rho \in \mathbf{T}$, rule $(\bot)$ states that $\bot^{\rho}$ is the smallest st-type in $\{\phi \mid \phi \in \mathbf{L}'^{\mathrm{st}}_{\bot} \cup \mathbf{L}'^{\mathrm{st}}_{\top}$ and $\epsilon(\phi) = \rho\}$, and therefore the most informative between the strictness properties. Indeed the property $\bot^{\rho}$ is given only to the everywhere undefined term. Rule $(\not\bot)$ states that $\not\bot^{\rho}$ is the greatest st-type in $\{\phi \mid \phi \in \mathbf{L}'^{\mathrm{st}}_{\not\bot}$ and $\epsilon(\phi) = \rho\}$. This means that the property $\not\bot^{\rho}$ is the least informative between the totality properties, in fact it only says

$$(\bot)\ \frac{\phi \in \mathbf{L}'^{\mathrm{st}}_{\bot} \cup \mathbf{L}'^{\mathrm{st}}_{\top}}{\bot^{\epsilon(\phi)} \leq \phi} \qquad (\not\bot)\ \frac{\phi \in \mathbf{L}'^{\mathrm{st}}_{\not\bot}}{\phi \leq \not\bot^{\epsilon(\phi)}} \qquad (\top)\ \frac{\psi \in \mathbf{L}'^{\mathrm{st}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}$$

$$(\mathrm{Ref})\ \phi \leq \phi \qquad (\to)\ \frac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_2 \to \psi_1 \leq \phi_1 \to \psi_2}$$

$$(\mathrm{Mon}_0)\ \frac{\phi \in \mathbf{L}'^{\mathrm{st}}_{\not\bot} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}}{\phi \to \psi \leq \mathrm{bw}(\phi) \to \mathrm{dn}(\psi)} \qquad (\mathrm{Mon}_1)\ \frac{\phi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\bot} \cup \mathbf{L}'^{\mathrm{st}\bullet}_{\not\bot}}{\phi \to \psi \leq \mathrm{dn}(\phi) \to \mathrm{dn}(\psi)}$$

$$(\mathrm{Mon}_2)\ \phi \to \psi \leq \mathrm{up}(\phi) \to \mathrm{up}(\psi) \qquad (\mathrm{Trans})\ \frac{\phi \leq \psi \quad \psi \leq \chi}{\phi \leq \chi}$$

Figure 4.8: Entailment rules for strictness and totality types (system $\leq'_{\mathrm{st}}$)

that the term has a w.h.n.f.. Rule $(\top)$ says that the $\top$-st-types are the maximal elements of $\{\phi \mid \epsilon(\phi) = \rho\}$ and are all equivalent. ($\top$-st-types do not give any information about terms.) Rules (Ref), $(\to)$ and (Trans) are standard. Rules $(\mathrm{Mon}_0)$, $(\mathrm{Mon}_1)$, and $(\mathrm{Mon}_2)$ rely on the monotonicity of the evaluation rules of PCF$'$, as the proof of Theorem 4.59 shows. These rules allow the comparison of st-types having both strictness and totality constructors.

**Example 4.58** *Using $\leq'_{\mathrm{st}}$, the the weakness of the system $\leq_{\mathrm{st}}$ shown Example 4.48 can be correctly handled. In fact*

- $\not\bot^{\rho} \to \bot^{\sigma} \leq'_{\mathrm{st}} \top^{\rho} \to \bot^{\sigma} = \mathrm{dn}(\not\bot^{\rho}) \to \mathrm{dn}(\bot^{\sigma})$, *and this entailment cannot be proved without using rule* $(\mathrm{Mon}_1)$.
- *Similarly, it not possible to prove* $\bot^{\rho} \to \not\bot^{\sigma} \leq'_{\mathrm{st}} \top^{\rho} \to \not\bot^{\rho} = \mathrm{up}(\bot^{\rho}) \to \mathrm{up}(\not\bot^{\rho})$ *without using rule* $(\mathrm{Mon}_2)$.

*Moreover, the entailment* $(\not\bot^{\rho} \to \not\bot^{\sigma}) \to \bot^{\rho} \leq'_{\mathrm{st}} (\top^{\rho} \to \bot^{\sigma}) \to \bot^{\rho} = \mathrm{bw}(\not\bot^{\rho} \to \not\bot^{\sigma}) \to \mathrm{dn}(\bot^{\rho})$ *cannot be proved without rule* $(\mathrm{Mon}_0)$.

The soundness of the entailment relation $\leq'_{\mathrm{st}}$ (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$) is stated by the following theorem.

**Theorem 4.59 (Soundness of $\leq'_{\mathrm{st}}$)** *Let $\phi, \psi \in \mathbf{L}'^{\mathrm{st}}$. $\phi \leq'_{\mathrm{st}} \psi$ implies $[\![\phi]\!] \subseteq [\![\psi]\!]$.*

**Proof.** By induction on the derivation of $\phi \leq'_{\mathrm{st}} \psi$. The only interesting rules are the $(\mathrm{Mon}_2)$, $(\mathrm{Mon}_1)$, and $(\mathrm{Mon}_0)$.

Consider $(\mathrm{Mon}_2)$. We have to show that $[\![\phi \to \psi]\!] \subseteq [\![\mathrm{up}(\phi) \to \mathrm{up}(\psi)]\!]$.
For $\phi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\bot}$ or $\psi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\bot}$ it is immediate. In the first case since $\mathrm{up}(\phi) = \phi$, and $[\![\psi]\!] \subseteq [\![\mathrm{up}(\psi)]\!]$. In the second since $[\![\mathrm{up}(\phi) \to \mathrm{up}(\psi)]\!] = \mathbf{I}(\epsilon(\phi))$.
So let $\psi = \psi_1 \to \cdots \psi_m \to \not\bot^{\sigma}$ $(m \geq 0)$, and $\phi \in \mathbf{L}'^{\mathrm{st}}_{\bot}$. Then $\mathrm{up}(\phi) = \top^{\epsilon(\phi)}$ and $\mathrm{up}(\psi) = \psi$. For each $[F] \in [\![\phi \to \psi]\!]$, and $[Q_i] \in [\![\psi_i]\!]$ $(1 \leq i \leq m)$ we have $[F\,(\mathsf{fix}\, x^{\epsilon(\phi)}.x)\, Q_1 \cdots Q_m]$

$\in [\![\not\perp^\sigma]\!]$. But, for each $[N] \in [\![\top^{\epsilon(\phi)}]\!]$ we have $\mathsf{fix}\, x^{\epsilon(\phi)}.x \preceq^{\mathrm{lazy}}_{\mathrm{obs}} N$, and so for monotonicity $[F\, N\, Q_1 \cdots Q_m] \in [\![\not\perp^\sigma]\!]$, i.e. $[F] \in [\![\mathrm{up}(\phi) \to \mathrm{up}(\psi)]\!]$.

For $(\mathrm{Mon}_1)$, we have to show that for $\phi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\perp} \cup \mathbf{L}'^{\mathrm{st}\bullet}_{\not\perp}$ we have that $[\![\phi \to \psi]\!] \subseteq [\![\mathrm{dn}(\phi) \to \mathrm{dn}(\psi)]\!]$. For $\phi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\perp}$ or $\psi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\perp}$ it is immediate.
So let $\phi \in \mathbf{L}'^{\mathrm{st}\bullet}_{\not\perp} = \{\not\perp^\tau \mid \tau \in \mathbf{T}\} \cup \mathbf{L}'^{\mathrm{st}\to}_{\not\perp}$ and $\psi = \psi_1 \to \cdots \psi_m \to \perp^\sigma$ $(m \geq 0)$. Then $\mathrm{dn}(\phi) = \top^{\epsilon(\phi)}$, and $\mathrm{dn}(\psi) = \psi$.
For each $[F] \in [\![\phi \to \psi]\!]$, $[M] \in [\![\phi]\!]$, and $[Q_i] \in [\![\psi_i]\!]$ $(1 \leq i \leq m)$ we have $[F\, M\, Q_1 \cdots Q_m]$ $\in [\![\perp^\sigma]\!]$. But, for each $[N] \in [\![\top^\phi]\!]$ there exists (see the proof of Lemma 4.54.2) $[M'] \in [\![\phi]\!]$ such that $N \preceq^{\mathrm{lazy}}_{\mathrm{obs}} M'$, and so for monotonicity $[F\, N\, Q_1 \cdots Q_m] \in [\![\perp^\sigma]\!]$, i.e. $[F] \in [\![\mathrm{dn}(\phi) \to \mathrm{dn}(\psi)]\!]$.

Finally for $(\mathrm{Mon}_0)$, we have to show that, if $\phi \in \mathbf{L}'^{\mathrm{st}}_{\not\perp} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\perp}$ then $[\![\phi \to \psi]\!] \subseteq [\![\mathrm{bw}(\phi) \to \mathrm{dn}(\psi)]\!]$. Let $\phi = \phi_1 \to \cdots \to \phi_n \to \not\perp^\iota$ $(n \geq 1)$. For $\psi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\perp}$ it is immediate. So let $\psi = \psi_1 \to \cdots \psi_m \to \perp^\sigma$ $(m \geq 0)$. Then $\mathrm{bw}(\phi) = \top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_n)} \to \perp^\iota$ and $\mathrm{dn}(\psi) = \psi$.
For each $[F] \in [\![\phi \to \psi]\!]$, $[M] \in [\![\phi]\!]$, and $[Q_i] \in [\![\psi_i]\!]$ $(1 \leq i \leq m)$ we have $[F\, M\, Q_1 \cdots Q_m]$ $\in [\![\perp^\sigma]\!]$. But, for each $[N] \in [\![\mathrm{bw}(\phi)]\!]$ there exists (see Lemma 4.56) $[M'] \in [\![\phi]\!]$ such that $N \preceq^{\mathrm{lazy}}_{\mathrm{obs}} M'$, and so for monotonicity $[F\, N\, Q_1 \cdots Q_m] \in [\![\perp^\sigma]\!]$, i.e. $[F] \in [\![\mathrm{bw}(\phi) \to \mathrm{dn}(\psi)]\!]$.
□

As far as rule $(\mathrm{Mon}_1)$, let $\mathbf{L}'^{\mathrm{st}\bullet\bullet}_{\not\perp}$ be the subset of $\mathbf{L}'^{\mathrm{st}}_{\not\perp}$ defined by:

$$\mathbf{L}'^{\mathrm{st}\bullet\bullet}_{\not\perp} = \{\phi_1 \to \cdots \to \phi_n \to \not\perp^\iota \mid n \geq 1,\ \iota \in \{\mathsf{nat}, \mathsf{bool}\},\ \text{and } \phi_1, \ldots, \phi_n \in \mathbf{L}'^{\mathrm{st}}_{\not\perp}\}.$$

We can show that it is not sound to extend the premise of the rule $(\mathrm{Mon}_1)$ to st-types in $\mathbf{L}'^{\mathrm{st}}_{\not\perp} - (\mathbf{L}'^{\mathrm{st}\bullet}_{\not\perp} \cup \mathbf{L}'^{\mathrm{st}\bullet\bullet}_{\not\perp})$. Note that the st-types in $\mathbf{L}'^{\mathrm{st}}_{\not\perp} - (\mathbf{L}'^{\mathrm{st}\bullet}_{\not\perp} \cup \mathbf{L}'^{\mathrm{st}\bullet\bullet}_{\not\perp})$ are exactly the ones of the shape $\phi_1 \to \cdots \to \phi_n \to \not\perp^\iota$, where $n \geq 1$, $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$, and, for some $i$ $(1 \leq i \leq n)$, $\phi_i \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\perp}$.

**Proposition 4.60** *Let* $\phi \in \mathbf{L}'^{\mathrm{st}}_{\not\perp} - (\mathbf{L}'^{\mathrm{st}\bullet}_{\not\perp} \cup \mathbf{L}'^{\mathrm{st}\bullet\bullet}_{\not\perp})$ *and* $\psi \in \mathbf{L}'^{\mathrm{st}}_{\perp}$. *Then* $[\![\phi \to \psi]\!] \not\subseteq [\![\mathrm{dn}(\phi) \to \mathrm{dn}(\psi)]\!]$.

**Proof.** Let $\phi \in \mathbf{L}'^{\mathrm{st}}_{\not\perp} - (\mathbf{L}'^{\mathrm{st}\bullet}_{\not\perp} \cup \mathbf{L}'^{\mathrm{st}\bullet\bullet}_{\not\perp})$, i.e. $\phi = \phi_1 \to \cdots \to \phi_i \to \cdots \to \phi_n \to \not\perp^\iota$, for some $n \geq 1$, $\phi_1, \ldots, \phi_{i-1}, \phi_{i+1}, \ldots, \phi_n \in \mathbf{L}'^{\mathrm{st}}$, $\phi_i \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\perp}$ and $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$, where $\epsilon(\phi_i) = \tau_1 \to \cdots \to \tau_m \to \iota'$ for some $m \geq 0$, $\tau_1, \ldots, \tau_m \in \mathbf{T}$, and $\iota' \in \{\mathsf{nat}, \mathsf{bool}\}$. Let $\psi \in \mathbf{L}'^{\mathrm{st}}_{\perp}$. Take $[M] \in [\![\phi \to \psi]\!]$ defined by

$$\begin{array}{rll} M = \lambda\, f^{\epsilon(\phi)}.\mathsf{if} & f\, N_1 \cdots N_{i-1}\, (\lambda\, x_1^{\tau_1}. \cdots \lambda\, x_m^{\tau_m}.\mathsf{c}_1^{\iota'})\, N_{i+1} \cdots N_n \\ & = \\ & f\, N_1 \cdots N_{i-1}\, (\lambda\, x_1^{\tau_1}. \cdots \lambda\, x_m^{\tau_m}.\mathsf{c}_2^{\iota'})\, N_{i+1} \cdots N_n \\ & \quad \mathsf{then}\ \mathsf{fix}\, y^{\epsilon(\psi)}.y \\ & \quad \mathsf{else}\ N_0 \end{array}$$

where $[N_0] \notin [\![\psi]\!]$, $\mathsf{c}_1^{\iota'} \neq \mathsf{c}_2^{\iota'}$, and, for all $j$, $1 \leq j \neq i \leq n$, $[N_j] \in [\![\phi_j]\!]$. For all $[F] \in [\![\phi]\!]$, $[F\, N_1 \cdots N_{i-1}\, (\mathsf{fix}\, x^{\epsilon(\phi_i)}.x)\, N_{i+1} \cdots N_n] = [\mathsf{c}]$, where $\mathsf{c}$ is either an integer or a boolean

value, since $[\mathsf{fix}\, x^{\epsilon(\phi_i)}.x] \in [\![\phi_i]\!]$ for $\phi_i \in \mathbf{L}^{\mathsf{st}\prime}_{\top} \cup \mathbf{L}^{\mathsf{st}\prime}_{\bot}$. Therefore (from the monotonicity of terms), both

$$[F\, N_1 \cdots N_{i-1}\, (\lambda\, x_1^{\tau_1}. \cdots \lambda\, x_m^{\tau_m}.\mathsf{c}_1^{\iota'})\, N_{i+1} \cdots N_n] = [\mathsf{c}]\ ,$$

and

$$[F\, N_1 \cdots N_{i-1}\, (\lambda\, x_1^{\tau_1}. \cdots \lambda\, x_m^{\tau_m}.\mathsf{c}_2^{\iota'})\, N_{i+1} \cdots N_n] = [\mathsf{c}]\ ,$$

and so $[M] \in [\![\phi \to \psi]\!]$. However, $[M] \notin [\![\mathrm{dn}(\phi) \to \mathrm{dn}(\psi)]\!]$ since the term $Q = \lambda\, y_1^{\epsilon(\phi_1)}. \cdots \lambda\, y_n^{\epsilon(\phi_n)}.y_i\, Q_1 \cdots Q_m$, where $Q_1, \ldots, Q_m \in \Lambda^{\mathbf{c}}_{\mathbf{T}}$, is such that $[Q] \in [\![\mathrm{dn}(\phi)]\!]$ and $[M\, Q] \notin [\![\mathrm{dn}(\psi)]\!]$. □

**Remark 4.61** *The entailment rules of $\leq'_{\mathrm{st}}$ (in Fig. 4.8) are not complete w.r.t. the interpretation in $\mathcal{M}^{\mathrm{lazy}}$. In fact the arguments used in Examples 4.10 and 4.35 apply also for $\leq'_{\mathrm{st}}$. Moreover, we can show that the entailment rules of Definition 4.57 are not complete even w.r.t. the interpretation in $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$. The failure of completeness w.r.t. $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$ is due mainly to the fact that the rules for entailment, even though they take into account monotonicity of terms, do not take into account their continuity. That is the behaviour of convergent terms is completely defined by their behaviour on a finite number of terms. For instance let $\phi = (\not\downarrow^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}) \to \bot^{\mathsf{nat}}$ and $\psi = \top^{\mathsf{nat}\to\mathsf{nat}} \to \bot^{\mathsf{nat}}$, we have*

$$[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} = [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$$

*but*

$$\phi \not\leq'_{\mathrm{st}} \psi. \tag{4.1}$$

*To prove $[\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ we introduce the notion of* syntactic approximant of a term *(see [Pitts, 1997] for a more formal development of this notion). For every $N \in \Lambda^{\mathsf{isdef}}_{\mathbf{T}}$ an* approximant *$A_N$ of $N$ is a term in which all the occurrences of $\mathsf{fix}$ are in subterms of the shape $\mathsf{fix}\, x^{\rho}.x$ and such that $A_N \preceq^{\mathrm{lazy}}_{\mathrm{obs}} N$. Roughly $A_N$ is obtained from $N$ by first unfolding some fixpoint expressions and then by replacing each fixpoint expression $(\mathsf{fix}\, x^{\rho}.P)$ by the always divergent term of the same type $(\mathsf{fix}\, x^{\rho}.x)$. Let $\mathbf{A}(N)$ be the set of approximants of $N$. For every $N \in \Lambda^{\mathsf{isdef}}_{\mathbf{T}}$ and $A_N \in \mathbf{A}(N)$, let $\overline{A_N}$ be the term obtained from $A_N$ by replacing each occurrence of $\mathsf{fix}\, x^{\rho}.x$ by the term $C^{\rho}$, which is inductively defined by*

- $C^{\mathsf{nat}} = 0$,
- $C^{\mathsf{bool}} = \mathsf{true}$, *and*
- $C^{\rho_1 \to \rho_2} = \lambda\, x^{\rho_1}.C^{\rho_2}$.

*Note that $A_N \preceq^{\mathrm{lazy}}_{\mathrm{obs}} \overline{A_N}$ (but, obviously, $\overline{A_N} \not\preceq^{\mathrm{lazy}}_{\mathrm{obs}} N$).*
*Assume now that there is $M$ such that $[M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$ and $[M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \notin [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, where $\phi = (\not\downarrow^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}) \to \bot^{\mathsf{nat}}$ and $\psi = \top^{\mathsf{nat}\to\mathsf{nat}} \to \bot^{\mathsf{nat}}$. Then, for some $N$, $MN \Downarrow$. It is a well known property of syntactic continuity that this implies that $MA_N \Downarrow$, for some $A_N \in \mathbf{A}(N)$, and then $M\overline{A_N} \Downarrow$, since $A_N \preceq^{\mathrm{lazy}}_{\mathrm{obs}} \overline{A_N}$. But $\overline{A_N}$ is a term of the simply*

*typed λ-calculus (with arithmetic and boolean constants) and, by the strong normalization theorem for simply typed λ-calculus,* $[\overline{A_N}]^{\mathcal{M}^{\text{lazy}}_{\text{isdef}}} \in [\![\not\downarrow^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}]\!]^{\mathcal{M}^{\text{lazy}}_{\text{isdef}}}$*, and this contradicts the assumption that* $[M]^{\mathcal{M}^{\text{lazy}}_{\text{isdef}}} \in [\![\phi]\!]^{\mathcal{M}^{\text{lazy}}_{\text{isdef}}}$*.*

*With a similar argument we can prove that*

$$[\![(\top^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}) \to \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\text{lazy}}_{\text{isdef}}} \subseteq [\![(\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\text{lazy}}_{\text{isdef}}}$$

*but*

$$(\top^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \not\leq'_{\text{st}} (\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \ . \tag{4.2}$$

*At the moment we do not know whether a sound (and decidable) extension of* $\leq'_{\text{st}}$ *which is complete w.r.t.* $\mathcal{M}^{\text{lazy}}_{\text{isdef}}$ *(or* $\mathcal{M}^{\text{lazy}}$*) exists.*

*The examples given to show the incompleteness of the rules for* $\leq'_{\text{st}}$ *suggests that we have to take into account (in addition to monotonicity) also continuity of terms. So we give the following stronger version of the rules* $(\text{Mon}_0)$ *and* $(\text{Mon}_1)$*:*

$$(\text{Cont}_0)\ \frac{\phi \in \mathbf{L}'^{\text{st}}_{\not\downarrow} - (\mathbf{L}'^{\text{st}^\bullet}_{\not\downarrow} \cup \mathbf{L}'^{\text{st}^{\bullet\bullet}}_{\not\downarrow})}{\phi \to \psi \leq \text{bw}'(\phi) \to \text{dn}(\psi)}$$

$$(\text{Cont}_1)\ \frac{\phi \in \mathbf{L}'^{\text{st}}_{\top} \cup \mathbf{L}'^{\text{st}}_{\bot} \cup \mathbf{L}'^{\text{st}^\bullet}_{\not\downarrow} \cup \mathbf{L}'^{\text{st}^{\bullet\bullet}}_{\not\downarrow}}{\phi \to \psi \leq \text{dn}(\phi) \to \text{dn}(\psi)} \quad ,$$

*where* $\text{bw}' : (\mathbf{L}'^{\text{st}}_{\not\downarrow} - (\mathbf{L}'^{\text{st}^\bullet}_{\not\downarrow} \cup \mathbf{L}'^{\text{st}^{\bullet\bullet}}_{\not\downarrow})) \to \mathbf{L}'^{\text{st}}_{\bot}$ *is defined by*

$$\text{bw}'(\phi_1 \to \cdots \to \phi_n \to \not\downarrow^\iota) = \phi_1[\bot := \top] \to \cdots \to \phi_n[\bot := \top] \to \bot^\iota \ ,$$

*and for* $\phi' \in \mathbf{L}'^{\text{st}}$*,* $\phi'[\bot := \top]$ *is the st-type obtained from* $\phi'$ *by replacing the constructor* $\bot$ *by the constructor* $\top$*. Let* $\leq''_{\text{st}}$ *be the entailment relation obtained by substituting* $(\text{Mon}_0)$ *with* $(\text{Cont}_0)$*, and* $(\text{Mon}_1)$ *with* $(\text{Cont}_1)$*. Note that rule* $(\text{Cont}_1)$ *allows to handle the counterexample (4.1) above, i.e. we can prove*

$$(\not\downarrow^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \leq''_{\text{st}} \top^{\mathsf{nat}\to\mathsf{nat}} \to \bot^{\mathsf{nat}},$$

*while rule* $(\text{Cont}_0)$ *allows to handle the counterexample (4.2), i.e. we can also show that*

$$(\top^{\mathsf{nat}} \to \not\downarrow^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \leq''_{\text{st}} (\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \ .$$

*We conjecture that the entailment* $\leq''_{\text{st}}$ *is sound w.r.t. the semantics* $\mathcal{M}^{\text{lazy}}$ *and* $\mathcal{M}^{\text{lazy}}_{\text{isdef}}$*. However, as we will see in Section 4.4.5, this stronger entailment relation does not improve the results of the strictness and totality analyses in Sections 4.2 and 4.3.*

### 4.4.3 An assignment system for strictness and totality

For each PCF′ constant $\mathsf{c}$, a set $\mathbf{L}'^{\text{st}}(\mathsf{c})$ of minimal st-types in $\mathbf{L}'^{\text{s}} \cup \mathbf{L}'^{\text{t}}$ is specified. The idea is that every st-type which is entailed by a st-type in $\mathbf{L}'^{\text{st}}(\mathsf{c})$ can be assigned to $\mathsf{c}$.

For example, for all natural numbers $\mathsf{n}$, $\mathbf{L}'^{\mathrm{st}}(\mathsf{n}) = \{\not\perp^{\mathsf{nat}}\}$ and $\mathbf{L}'^{\mathrm{st}}(+) = \{\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \perp^{\mathsf{nat}}, \top^{\mathsf{nat}} \to \perp^{\mathsf{nat}} \to \perp^{\mathsf{nat}}, \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}\}$.

Note that the assumption that, for all the PCF′ constants $\mathsf{c}$, $\mathbf{L}'^{\mathrm{st}}(\mathsf{c}) \subseteq \mathbf{L}'^{\mathrm{s}} \cup \mathbf{L}'^{\mathrm{t}}$ is not restrictive. Take, for instance, a binary arithmetic operator, like $+$. A st-type for $+$ is $\not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$, i.e. the result is defined whenever both the arguments are defined. All the other st-types of $+$ express the fact that $+N_1N_2$ is undefined whenever one of $N_1$ or $N_2$ is such. But, these st-type are of the shape either $\perp^{\mathsf{nat}} \to \phi \to \perp^{\mathsf{nat}}$ or $\phi \to \perp^{\mathsf{nat}} \to \perp^{\mathsf{nat}}$, where $\phi \in \{\top^{\mathsf{nat}}, \perp^{\mathsf{nat}}, \not\perp^{\mathsf{nat}}\}$, and, in this case, it is better to choose $\phi$ as large (w.r.t. $\leq'_{\mathrm{st}}$) as possible, i.e. $\top^{\mathsf{nat}}$. In fact, $\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \perp^{\mathsf{nat}} \leq'_{\mathrm{st}} \perp^{\mathsf{nat}} \to \phi \to \perp^{\mathsf{nat}}$ and $\top^{\mathsf{nat}} \to \perp^{\mathsf{nat}} \to \perp^{\mathsf{nat}} \leq'_{\mathrm{st}} \phi \to \perp^{\mathsf{nat}} \to \perp^{\mathsf{nat}}$, for all $\phi \in \{\top^{\mathsf{nat}}, \perp^{\mathsf{nat}}, \not\perp^{\mathsf{nat}}\}$. This argument can be applied to all the strict constants, like the ones in PCF′, and can be easily extended to non strict (but monotone) first order constants, like McCarthy's ambiguity operator (see Example 4.62 below).

**Example 4.62** *Consider the st-types*

$$(\not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \ , \qquad (\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \ ,$$
$$(\not\perp^{\mathsf{nat}} \to \perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \ , \qquad (\perp^{\mathsf{nat}} \to \perp^{\mathsf{nat}} \to \perp^{\mathsf{nat}}) \ ,$$

*which (as pointed out in [Solberg, 1995a] page 28) express strictness and totality properties of McCarthy's ambiguity operator (a binary operator such that if one of the two arguments terminates so does the function call but if both the two arguments diverge so does the function call).*

*Note that, however, the information expressed by the four st-types above is entailed (with both $\leq_{\mathrm{st}}$ and $\leq'_{\mathrm{st}}$) by the s-type*

$$(\perp^{\mathsf{nat}} \to \perp^{\mathsf{nat}} \to \perp^{\mathsf{nat}}) \ ,$$

*and the t-types*

$$(\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}), \qquad (\not\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \ .$$

**Definition 4.63 (Assignment system $\vdash'_{\mathrm{st}}$)** *We write $\Sigma \vdash'_{\mathrm{st}} M : \phi$ to mean that $\Sigma \vdash M : \phi$ can be derived by the rules in Fig. 4.9.*

**Example 4.64** *Let $\vdash_{\mathbf{T}}$ twice : $\rho$ where $\rho = (\sigma \to \sigma) \to \sigma \to \sigma$ and* twice $= \lambda f^{\sigma\to\sigma}.\ \lambda x^{\sigma}.\ f(f\,x)$*. It is easy to check that, for $i \in \{1,2,3,4\}$, $\vdash'_{\mathrm{st}}$* twice : $\phi_i$*, where*

$$\begin{array}{rcl} \phi_1 & = & (\perp^{\sigma} \to \perp^{\sigma}) \to \perp^{\sigma} \to \perp^{\sigma}, \\ \phi_2 & = & (\top^{\sigma} \to \perp^{\sigma}) \to \top^{\sigma} \to \perp^{\sigma}, \\ \phi_3 & = & (\not\perp^{\sigma} \to \not\perp^{\sigma}) \to \not\perp^{\sigma} \to \not\perp^{\sigma}, \ \textit{and} \\ \phi_4 & = & (\top^{\sigma} \to \not\perp^{\sigma}) \to \top^{\sigma} \to \not\perp^{\sigma} \end{array}$$

*are the st-types in the Examples 4.20 and 4.43.*

$$(\leq)\ \frac{\Sigma \vdash M : \phi \quad \phi \leq \psi}{\Sigma \vdash M : \psi}$$

$$(\text{Var})\ \Sigma, x : \phi \vdash x : \phi \qquad (\text{Con})\ \frac{\phi \in \mathbf{L}^{\text{st}}(\mathsf{c})}{\Sigma \vdash \mathsf{c} : \phi}$$

$$(\to \text{I})\ \frac{\Sigma, x : \phi \vdash M : \psi}{\Sigma \vdash \lambda x.M : \phi \to \psi} \qquad (\to \text{I}_{\not\downarrow})\ \frac{\vdash_{\mathbf{T}} \lambda x.M : \rho}{\Sigma \vdash \lambda x.M : \not\downarrow^{\rho}}$$

$$(\to \text{E})\ \frac{\Sigma \vdash M : \phi \to \psi \quad \Sigma \vdash N : \phi}{\Sigma \vdash MN : \psi}$$

$$(\text{Fix}_{\bot})\ \frac{\Sigma, x : \phi_1 \vdash M : \phi_2 \ \cdots\ \Sigma, x : \phi_k \vdash M : \phi_1 \quad \phi_1 \in \mathbf{L}'^{\text{st}}_{\bot} \cup \mathbf{L}'^{\text{st}}_{\top}}{\Sigma \vdash \mathsf{fix}\, x.M : \phi_i}\ 1 \leq i \leq k$$

$$(\text{Fix}_{\not\downarrow})\ \frac{\Sigma, x : \top^{\rho} \vdash M : \phi_1 \quad \Sigma, x : \phi_1 \vdash M : \phi_2 \ \cdots\ \Sigma, x : \phi_{k-1} \vdash M : \phi_k}{\Sigma \vdash \mathsf{fix}\, x.M : \phi_k}\ k \geq 1$$

$$(\text{If}_{\bot})\ \frac{\Sigma \vdash N : \bot^{\mathsf{bool}} \quad \vdash_{\mathbf{T}} M_1 : \rho \quad \vdash_{\mathbf{T}} M_2 : \rho}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \bot^{\rho}}$$

$$(\text{If}_{\not\downarrow})\ \frac{\Sigma \vdash N : \not\downarrow^{\mathsf{bool}} \quad \Sigma \vdash M_1 : \phi \quad \Sigma \vdash M_2 : \phi}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \phi}$$

$$(\text{If}_{\top})\ \frac{\Sigma \vdash N : \top^{\mathsf{bool}} \quad \Sigma \vdash M_1 : \phi \quad \Sigma \vdash M_2 : \phi \quad \phi \in \mathbf{L}'^{\text{st}}_{\bot} \cup \mathbf{L}'^{\text{st}}_{\top}}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \phi}$$

Figure 4.9: Rules for strictness and totality types assignment (system $\vdash'_{\text{st}}$)

As for the systems $\vdash_{\mathrm{s}}$ and $\vdash_{\mathrm{t}}$ we have the following fact.

**Fact 4.65** *1.* $\Sigma \vdash'_{\mathrm{st}} M : \phi$ *implies* $\vdash_{\mathbf{T}} M : \epsilon(\phi)$ *and* $\epsilon(\Sigma) \supseteq \mathrm{FV}(M)$.

*2.* $\vdash_{\mathbf{T}} M : \rho$ *implies, for all basis* $\Sigma$ *such that* $\Sigma = \{x^{\epsilon(\psi)} : \psi \mid x^{\epsilon(\psi)} \in \mathrm{FV}(M)$ *for some* $\psi \in \mathbf{L}'^{\mathrm{st}}\}$, $\Sigma \vdash'_{\mathrm{st}} M : \top^{\rho}$. □

Soundness of the st-types assignment system (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$) is stated by the following theorem.

**Theorem 4.66 (Soundness of $\vdash'_{\mathrm{st}}$)** *Let* $\Sigma \vdash'_{\mathrm{st}} M : \phi$. *Then* $[\![M]\!]_e \in [\![\phi]\!]$ *for all* $e$ *such that for all* $x : \psi \in \Sigma$, $e(x) \in [\![\psi]\!]$.

**Proof.** By induction on the derivation of $\Sigma \vdash'_{\mathrm{st}} M : \phi$. □

### 4.4.4 Comparison with the system in Chapter 2 of Solberg's PhD thesis

Besides the syntactical differences of using $\bot$ for **b** and $\not\bot$ for **n**, our st-types $\mathbf{L}'^{\mathrm{st}}$ are exactly the ones introduced in [Solberg, 1995a] page 31 (see also [Solberg *et al.*, 1994; Solberg *et al.*, ]).

The main drawback of *op. cit.* is that soundness of the analysis is proved w.r.t. a quite involved (and "not intuitive") extension of the operational semantics of the language (see [Solberg, 1995a] Section 2.3) in which the set of terms is extended with terms $\mathcal{T}_P$ that are the (non strict) total extensions of the (partial) functions represented by terms $P \in \Lambda^{\mathbf{c}}_{\mathbf{T}}$. Such terms are in general non monotonic and, therefore, non computable. This choice leads to conclusions that are not intuitive. For instance consider the term:

$$M = \lambda\, f^{\mathsf{nat}\to\mathsf{nat}}.\mathsf{if}\ f(1) = f(2)\ \mathsf{then}\ \mathsf{fix}\, x^{\mathsf{nat}}.x\ \mathsf{else}\ 3\ .$$

We have that (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$) $[M] \in [\![(\top^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}}) \to \bot^{\mathsf{nat}}]\!]$. In fact a term $N$ such that $[N] \in [\![\top^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}}]\!]$ can be only a (total) constant function of type $\mathsf{nat} \to \mathsf{nat}$ (e.g. $\lambda\, x^{\mathsf{nat}}.3$) and then $MN \Uparrow$. However, extending the language with the terms $\mathcal{T}_P$, $(\top^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}}) \to \bot^{\mathsf{nat}}$ is *not* a property of $M$, since there are *extended* terms $\mathcal{T}_P$ (where $P \in \Lambda^{\mathbf{c}}_{\mathbf{T}}$ is a strict function of type $\mathsf{nat} \to \mathsf{nat}$ returning at least two different natural numbers values) having the property $\top^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}}$ which do not denote constant functions, so we have that $M\mathcal{T}_P \Downarrow$. (Notice that these functions are not monotone.)

In this thesis we take a quite different approach since our soundness proof rely directly on the "standard" operational semantics of the programming language.

#### Entailment rules

Comparing our entailment, $\leq'_{\mathrm{st}}$, with the entailment $\leq_{\mathrm{ST}}$ in [Solberg, 1995a] page 33, we can see that our rule $(\top)$ corresponds to the rules (top1) and (top2) for $\leq_{\mathrm{ST}}$, while rules

($\bot$) and ($\not\bot$) correspond respectively to the rules (bot) and (notbot) for $\leq_{\mathrm{ST}}$. In particular rule ($\not\bot$) is an extension of rule (notbot), for instance $\bot^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}} \leq'_{\mathrm{st}} \not\bot^{\mathsf{nat}\to\mathsf{nat}}$, but not $\mathbf{b}^{\mathsf{nat}} \to \mathbf{n}^{\mathsf{nat}} \leq_{\mathrm{ST}} \mathbf{n}^{\mathsf{nat}\to\mathsf{nat}}$.

Our rule ($\mathrm{Mon}_2$), see Fig. 4.8, is the rule (monotone2) introduced in [Solberg, 1995a] page 35. This rule was not proved to be sound (see [Solberg, 1995a] page 36) and, therefore, it is not included in none of the systems presented in [Solberg *et al.*, 1994; Solberg *et al.*, ; Solberg, 1995a]. Theorem 4.59 shows that ($\mathrm{Mon}_2$) is sound w.r.t. the semantics $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$.

Finally both rules ($\mathrm{Mon}_1$) and ($\mathrm{Mon}_0$) are restriction of the following rule

$$\text{(monotone)}\ \phi \to \psi \leq \mathrm{dn}(\phi) \to \mathrm{dn}(\psi)$$

of $\leq_{\mathrm{ST}}$. This rule is not, as shown by Proposition 4.60, sound w.r.t. the semantics $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$.

So we can conclude that our entailment relation $\leq'_{\mathrm{st}}$ is an extension of the sound fragment of $\leq_{\mathrm{ST}}$.

**Assignment rules**

Comparing our strictness and totality type assignment system $\vdash'_{\mathrm{st}}$ (of Definition 4.63) with the assignment system $\vdash_{\mathrm{ST}}$ (in [Solberg, 1995a] page 39), we note that $\vdash_{\mathrm{ST}}$ allows to infer top level conjunctions of st-types. However, the only rule in which a conjunction of st-types is used in a non trivial way is the rule for assigning a property to a fixpoint. In $\vdash'_{\mathrm{st}}$ we simulate such limited use of conjunction by allowing multiple premises in the antecedent of the rules ($\mathrm{Fix}_\bot$) and ($\mathrm{Fix}_{\not\bot}$) (see Fig. 4.9). If we change our system by allowing top level conjunction in the style of [Solberg, 1995a] Chapter 2, we do not change its expressiveness. More precisely, let $\vdash'_{\wedge\mathrm{st}}$ the system with top level conjunction, then $\Sigma \vdash'_{\wedge\mathrm{st}} M : \phi_1 \wedge \cdots \wedge \phi_n$ if and only if $\Sigma \vdash'_{\mathrm{st}} M : \phi_1$, …, $\Sigma \vdash'_{\mathrm{st}} M : \phi_n$.

It also easy to see that replacing the two rules ($\mathrm{Fix}_\bot$) and ($\mathrm{Fix}_{\not\bot}$) of $\vdash'_{\mathrm{st}}$ by the following rule (fix) [2] of $\vdash_{\mathrm{ST}}$:

$$\text{(fix)}\ \frac{\Sigma, x:\phi_1 \vdash M:\phi_2 \quad \cdots \quad \Sigma, x:\phi_{n-1} \vdash M:\phi_n}{\Sigma \vdash \mathsf{fix}\, x.M : \phi_n} \left( \begin{array}{l} \phi_1 \in \mathbf{L}^{\prime\mathrm{st}}_{\bot} \cup \mathbf{L}^{\prime\mathrm{st}}_{\top} \text{ and} \\ \exists\, p, q.(p < p \text{ and } \phi_q \leq \phi_p) \end{array} \right)$$

does not change the power of the system.

So we can conclude that our strictness and totality type assignment system $\vdash'_{\mathrm{st}}$ is an extension of the sound fragment of $\vdash_{\mathrm{ST}}$.

### 4.4.5 Separability of strictness and totality analyses

In this section we show how the information that can be used to replace safely a lazy application by a strict one, obtained with the system $\vdash'_{\mathrm{st}}$ (in Section 4.4.3), can be obtained from the systems $\vdash_{\mathrm{s}}$ (in Section 4.2.2) and $\vdash_{\mathrm{t}}$ (in Section 4.3.2).

[2] The rule (fix) is indeed the formulation for our syntax of the corresponding rule of $\vdash_{\mathrm{ST}}$.

### The separability result

For $\phi \in \mathbf{L}'^{\mathrm{st}}$ let $\phi[\not\perp := \top] \in \mathbf{L}'^{\mathrm{s}}$ ($\phi[\perp := \top] \in \mathbf{L}'^{\mathrm{t}}$) be the s-type (t-type) obtained from $\phi$ by replacing the constructor $\not\perp$ ($\perp$) by the constructor $\top$.

Entailment between st-types is preserved by the previous substitutions of basic property constructors.

**Proposition 4.67** *Let $\phi \leq'_{\mathrm{st}} \psi$. Then*

1. *$\phi[\perp := \top] \leq_{\mathrm{t}} \psi[\perp := \top]$, and*
2. *$\phi[\not\perp := \top] \leq_{\mathrm{s}} \psi[\not\perp := \top]$.*

**Proof.** The proof of both 1. and 2. are by induction on the derivation of $\phi \leq'_{\mathrm{st}} \psi$.

1. If $\phi \leq'_{\mathrm{st}} \psi$ was derived by the rules ($\top$), ($\not\perp$), or (Ref), then $\phi[\perp := \top] \leq_{\mathrm{t}} \psi[\perp := \top]$ is derivable, from the same rule.
   If $\phi \leq'_{\mathrm{st}} \psi$ was derived from the rule ($\perp$), then $\psi[\perp := \top] \in \mathbf{L}'^{\mathrm{t}}$ so $\phi[\perp := \top] \leq_{\mathrm{t}} \psi[\perp := \top]$ is derivable from the rule ($\top$).
   Consider rule ($\mathrm{Mon}_0$). $\phi = \phi_1 \to \phi_2$ and $\psi = \mathrm{bw}(\phi_1) \to \mathrm{dn}(\phi_2)$. For all $\phi_2 \in \mathbf{L}'^{\mathrm{st}}$, $(\mathrm{dn}(\phi_2))[\perp := \top] \in \mathbf{L}^{\mathrm{t}}_{\top}$. Therefore $\phi[\perp := \top] \leq_{\mathrm{t}} \psi[\perp := \top]$ is derivable from the rule ($\top$).
   For rule ($\mathrm{Mon}_1$) we have the same argument.
   For rule ($\mathrm{Mon}_2$) note that, for all $\phi \in \mathbf{L}'^{\mathrm{st}}$, $\phi[\perp := \top] = (\mathrm{up}(\phi))[\perp := \top]$. So the proof is immediate.
   The rules ($\to$) and (Trans) are easily handled by inductive hypotheses.

2. If $\phi \leq'_{\mathrm{st}} \psi$ was derived by the rules ($\top$), ($\perp$), or (Ref), then $\phi[\not\perp := \top] \leq_{\mathrm{s}} \psi[\not\perp := \top]$ is derivable from the same rule.
   If $\phi \leq'_{\mathrm{st}} \psi$ was derived by the rule ($\not\perp$), then $\psi[\not\perp := \top] \in \mathbf{L}^{\mathrm{s}}_{\top}$, so $\phi[\not\perp := \top] \leq_{\mathrm{s}} \psi[\not\perp := \top]$ is derivable from the rule ($\top$).
   Consider rule ($\mathrm{Mon}_0$). $\phi = \phi_1 \to \phi_2$ and $\psi = \mathrm{bw}(\phi_1) \to \mathrm{dn}(\phi_2)$. If $\phi_2 \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\perp}$, then $\phi_2[\not\perp := \top] \in \mathbf{L}^{\mathrm{s}}_{\top}$ and the proof is trivial. If $\phi_2 \in \mathbf{L}'^{\mathrm{st}}_{\perp}$, then $\mathrm{dn}(\phi_2) = \phi_2$. Moreover, since $\phi_1 \in \mathbf{L}'^{\mathrm{st}}_{\not\perp} - \mathbf{L}'^{\mathrm{st}\bullet}_{\not\perp}$, $\phi_1[\not\perp := \top] \in \mathbf{L}^{\mathrm{s}}_{\top}$, and so

$$(\to)\ \frac{(\top)\ \dfrac{\phi_1[\not\perp := \top] \in \mathbf{L}^{\mathrm{s}}_{\top}}{(\mathrm{bw}(\phi_1))[\not\perp := \top] \leq_{\mathrm{s}} \phi_1[\not\perp := \top]} \quad (\mathrm{Ref})\ \dfrac{(\mathrm{dn}(\phi_2))[\not\perp := \top] \leq_{\mathrm{s}} (\mathrm{dn}(\phi_2))[\not\perp := \top]}{}}{\phi_1[\not\perp := \top] \to \phi_2[\not\perp := \top] \leq_{\mathrm{s}} (\mathrm{bw}(\phi_1))[\not\perp := \top] \to (\mathrm{dn}(\phi_2))[\not\perp := \top]}$$

   Consider rule ($\mathrm{Mon}_1$). In this case $\phi = \phi_1 \to \phi_2$ and $\psi = \mathrm{dn}(\phi_1) \to \mathrm{dn}(\phi_2)$. If $\phi_2 \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\perp}$, then $\psi \in \mathbf{L}'^{\mathrm{st}}_{\top}$. Therefore $\phi[\not\perp := \top] \leq_{\mathrm{s}} \psi[\not\perp := \top]$ is derivable from the rule ($\top$). Otherwise ($\phi_2 \in \mathbf{L}'^{\mathrm{st}}_{\perp}$) $(\mathrm{dn}(\phi_2))[\not\perp := \top] = \phi_2[\not\perp := \top]$. If $\phi_1 \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\perp}$ then $\phi_1[\not\perp := \top] \in \mathbf{L}^{\mathrm{s}}_{\top}$, and so $\phi_1[\not\perp := \top] \to \phi_2[\not\perp := \top] \leq_{\mathrm{s}} (\mathrm{dn}(\phi_1))[\not\perp := \top] \to (\mathrm{dn}(\phi_2))[\not\perp := \top]$ is derivable from the rule ($\to$). If $\phi_1 \in \mathbf{L}'^{\mathrm{st}}_{\perp}$ then $\mathrm{dn}(\phi_1) = \phi_1$ and the result is derivable from the rule (Ref).
   Finally for rule ($\mathrm{Mon}_2$). $\phi = \phi_1 \to \phi_2$ and $\psi = \mathrm{up}(\phi_1) \to \mathrm{up}(\phi_2)$. If $\phi_2 \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\perp}$,

then $\psi \in \mathbf{L}'^{\mathrm{st}}_{\top}$. If $\phi_2 \in \mathbf{L}'^{\mathrm{st}}_{\not\bot}$, then $\phi_2[\not\bot := \top] \in \mathbf{L}'^{\mathrm{st}}_{\top}$. So in both cases $\phi[\not\bot := \top] \leq_{\mathrm{s}} \psi[\not\bot := \top]$ is derivable from the rule $(\top)$.
For rules $(\to)$ and (Trans) use the inductive hypotheses. □

For st-types in $\mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\bot}$ (all types except the one in $\mathbf{L}'^{\mathrm{st}}_{\bot}$) we can prove an even stronger result, i.e. that $\phi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\bot}$ is equivalent to a st-type not containing $\bot$, notably $\phi[\bot := \top]$ as the following proposition shows. (The analogous result does not hold for st-types in $\mathbf{L}'^{\mathrm{st}}_{\bot}$.)

**Proposition 4.68** *Let $\phi \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\bot}$. Then $\phi \cong'_{\mathrm{st}} \phi[\bot := \top]$.*

**Proof.** By structural induction on $\phi$. The cases $\phi \cong'_{\mathrm{st}} \top^{\rho}$ and $\phi = \not\bot^{\rho}$ for some $\rho$ are immediate. So let $\phi = \phi_1 \to \cdots \to \phi_n \to \not\bot^{\rho}$, $n \geq 1$. If $\phi_1, \ldots, \phi_n \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\bot}$ the result follows by induction. If $\phi_i \in \mathbf{L}'^{\mathrm{st}}_{\bot}$, then $\mathrm{up}(\phi_i) \cong'_{\mathrm{st}} \phi_i[\bot := \top] \in \mathbf{L}^{\mathrm{t}}_{\top}$ and (by using $(\mathrm{Mon}_2)$ in one direction, and $(\to)$ in the other) $\phi \cong'_{\mathrm{st}} \phi_1 \to \cdots \to \phi_i[\bot := \top] \to \cdots \to \phi_n \to \not\bot^{\rho}$. So, by applying this transformation to all the $\phi_j \in \mathbf{L}'^{\mathrm{st}}_{\bot}$ $(1 \leq j \leq n)$, we obtain $\phi \cong'_{\mathrm{st}} \phi'_1 \to \cdots \to \phi'_n \to \not\bot^{\rho}$, where $\phi'_1, \ldots, \phi'_n \in \mathbf{L}'^{\mathrm{st}}_{\top} \cup \mathbf{L}'^{\mathrm{st}}_{\not\bot}$, and we can apply the induction hypotheses. □

From Proposition 4.67 we can derive the following theorem, which states that for the derivation of properties $\in \mathbf{L}'^{\mathrm{s}}$ the distinction between the constructor $\not\bot$ and the constructor $\top$ is not relevant, and similarly for the derivation of properties $\in \mathbf{L}'^{\mathrm{t}}$ the distinction between $\bot$ and $\top$ is not relevant.

**Theorem 4.69** *Let $\Sigma \vdash'_{\mathrm{st}} M : \phi$. Then*

1. *$\Sigma[\bot := \top] \vdash_{\mathrm{t}} M : \phi[\bot := \top]$, and*
2. *$\Sigma[\not\bot := \top] \vdash_{\mathrm{s}} M : \phi[\not\bot := \top]$.*

**Proof.** By induction on the derivation of $\Sigma \vdash_{\mathrm{st}} M : \phi$.
If the last rule applied is (Var), then both 1. and 2. are obvious.
Let the last rule applied be (Con). Since constants have types that are either in $\mathbf{L}'^{\mathrm{s}}$ or in $\mathbf{L}'^{\mathrm{t}}$, when $\phi \in \mathbf{L}'^{\mathrm{s}}$ then $\phi[\bot := \top] \cong_{\mathrm{t}} \top^{\rho}$. So from Fact 4.44 we have that $\Sigma[\bot := \top] \vdash_{\mathrm{t}} M : \top^{\rho}$. Similarly for $\phi \in \mathbf{L}'^{\mathrm{t}}$.
If the last rule applied is $(\to \mathrm{I})$, or $(\to \mathrm{E})$ both 1. and 2. are by induction hypotheses.
If the last rule applied is $(\to \mathrm{I}_{\not\bot})$, then $M = \lambda x.M'$ for some $M'$, and $\phi = \not\bot^{\rho}$. So $\Sigma[\bot := \top] \vdash_{\mathrm{t}} M : \phi[\bot := \top]$ is derivable by applying the same rule. On the other hand, since $\phi[\not\bot := \top] \cong_{\mathrm{s}} \top^{\rho}$, then from Fact 4.21, $\Sigma[\not\bot := \top] \vdash_{\mathrm{s}} M : \top^{\rho}$.
If the last rule applied is $(\leq)$ both results derive from induction hypothesis and Proposition 4.67.
If the last rule applied is $(\mathrm{If}_{\bot})$. Then $\phi = \bot^{\rho}$, and $\phi[\bot := \top] = \top^{\rho}$. Therefore $\Sigma[\bot := \top] \vdash_{\mathrm{t}} M : \top^{\rho}$ from Fact 4.44. On the other hand from the induction hypothesis and applying the rule $(\mathrm{If}_{\bot})$ (which is also a rule of $\vdash_{\mathrm{s}}$), we derive $\Sigma[\not\bot := \top] \vdash_{\mathrm{s}} M : \phi[\not\bot := \top] = \bot^{\rho}$.
If the last rule applied is $(\mathrm{If}_{\not\bot})$, from the induction hypothesis and applying the rule $(\mathrm{If}_{\not\bot})$ (which is also a rule of $\vdash_{\mathrm{t}}$), we derive $\Sigma[\bot := \top] \vdash_{\mathrm{t}} M : \phi[\bot := \top]$. On the other side to derive $\Sigma[\not\bot := \top] \vdash_{\mathrm{s}} M : \phi[\not\bot := \top]$ we have to apply the induction hypothesis and the the

rule ($\text{If}_\top$), which is applicable since $\phi[\not\perp := \top] \in \mathbf{L}'^{\text{s}}$.
If the last rule applied is ($\text{If}_\top$) both 1. and 2. are by induction hypotheses followed by the application of the rule ($\text{If}_\top$).
Finally, if the last rule applied is ($\text{Fix}_\perp$) or ($\text{Fix}_{\not\perp}$) then apply induction hypotheses (for point 1. replace ($\text{Fix}_\perp$) by ($\text{Fix}_{\not\perp}$), and for point 2. replace ($\text{Fix}_{\not\perp}$) by ($\text{Fix}_\perp$)). □

### A consequence of the separability result

Theorem 4.69 shows that both $\vdash_{\text{s}}$ and $\vdash_{\text{t}}$ are conservative over $\vdash'_{\text{st}}$. The following discussion shows that, to safely replace call-by-need with call-by-value, we are indeed interested in proving statements which involve only strictness or totality types. In this case is useless to consider $\vdash'_{\text{st}}$ instead of $\vdash_{\text{s}}$ and $\vdash_{\text{t}}$.

As explained in the introduction at the beginning of Part II (page 35), given an application

$$FM_1 \cdots M_n,$$

where $n \geq 1$, we are interested in proving, for $k \in \{1, \ldots, n\}$,

1. either
$$\Sigma \vdash'_{\text{st}} F : \psi_1 \to \cdots \to \psi_{k-1} \to \perp^\rho \to \psi_{k+1} \to \cdots \to \psi_n \to \perp^\sigma \ ,$$
where for all $i \in \{1, \ldots, k-1, k+1, \ldots, n\}$ $M_i$ has the property $\psi_i$,

2. or
$$\Sigma \vdash'_{\text{st}} M_k : \not\perp^\rho \ ,$$

for some basis $\Sigma$. In fact in both cases the application to the $k$-th argument can be safely done "by value". But, by Theorem 4.69, we have that

1. $\Sigma \vdash'_{\text{st}} F : \psi_1 \to \cdots \to \psi_{k-1} \to \perp^\rho \to \psi_{k+1} \to \cdots \to \psi_n \to \perp^\sigma$ implies
   - $\Sigma[\not\perp := \top] \vdash_{\text{s}} F : \psi_1[\not\perp := \top] \to \cdots \to \psi_{k-1}[\not\perp := \top] \to \perp^\rho \to \psi_{k+1}[\not\perp := \top] \to \cdots \to \psi_n[\not\perp := \top] \to \perp^\sigma$, moreover
   - for every $i \in \{1, \ldots, k-1, k+1, \ldots, n\}$, if $\Sigma \vdash'_{\text{st}} M_i : \psi_i$ then $\Sigma[\not\perp := \top] \vdash_{\text{s}} M_i : \psi_i[\not\perp := \top]$.

   This means that, as far as we interested in properties like $\psi_1 \to \cdots \to \psi_{m-1} \to \perp^\rho \to \psi_{m+1} \to \cdots \to \psi_n \to \perp^\sigma$ above, using $\vdash'_{\text{st}}$ or $\vdash_{\text{s}}$ is not relevant.

2. $\Sigma \vdash'_{\text{st}} M_k : \not\perp^\rho$ implies
   - $\Sigma[\perp := \top] \vdash_{\text{t}} M_k : \not\perp^\rho$.

   This means that, as far as we interested in properties like $\not\perp^\rho$, using $\vdash'_{\text{st}}$ or $\vdash_{\text{t}}$ is not relevant.

We note, however, that the separability of the two analyses is due to the weakness of the $\vdash'_{\mathrm{st}}$ *type inference rules*. In fact the language of st-types allows to express properties which would *not* be expressible in the separate systems. For instance take $(\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \perp^{\mathsf{nat}}$. This is the property of the terms which map (terminating) constant functions in a divergent computation. If we consider the program:

$$M = \lambda\, f^{\mathsf{nat}\to\mathsf{nat}}.\mathsf{if}\ f1 = f2\ \mathsf{then}\ \mathsf{fix}\, x^{\mathsf{nat}}.x\ \mathsf{else}\ 3\ ,$$

then we have that $[M] \in [\![(\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \perp^{\mathsf{nat}}]\!]$, while $[M] \notin [\![(\top^{\mathsf{nat}} \to \top^{\mathsf{nat}}) \to \perp^{\mathsf{nat}}]\!]$. But, by Theorem 4.69, we *cannot* prove $\vdash'_{\mathrm{st}} M : (\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \perp^{\mathsf{nat}}$.

**Remark 4.70** *Theorem 4.69 holds also for the system $\vdash''_{\mathrm{st}}$, which is obtained from $\vdash'_{\mathrm{st}}$ by using the entailment $\leq''_{\mathrm{st}}$ introduced in Remark 4.61. So the independence result goes through also for this system.*

## 4.5 Summary

The contents of this chapter can be summarized as follows.

- The introduction of the non-standard type assignment for strictness $\vdash_{\mathrm{s}}$ (in Section 4.2.2) and of the non-standard type assignment for totality $\vdash_{\mathrm{t}}$ (in Section 4.3.2), which are sound w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$ of Section 2.4.1.

- The proof of completeness for the entailment relation for s-types $\leq_{\mathrm{s}}$ (of the system $\vdash_{\mathrm{s}}$) and for the entailment relation for t-types $\leq_{\mathrm{t}}$ (of the system $\vdash_{\mathrm{t}}$), w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}$ of Section 2.4.2.

- The introduction (in Section 4.2.3) of the non-standard type assignment for strictness $\vdash^{\mathrm{g}}_{\mathrm{s}}$ (which is sound w.r.t. the model $\mathcal{M}^{\mathrm{gnd}}$ of Section 2.4.1) with an entailment relation, $\leq^{\mathrm{g}}_{\mathrm{s}}$, which is complete w.r.t. the model $\mathcal{M}^{\mathrm{gnd}}$ of Section 2.4.1.

- The definition of the combined system for strictness and totality $\vdash'_{\mathrm{st}}$ which is sound w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$ of Section 2.4.1.

- The proof that (for the systems $\vdash'_{\mathrm{st}}$ and $\vdash''_{\mathrm{st}}$ introduced in this chapter, and so also for the system $\vdash_{\mathrm{ST}}$ presented in in [Solberg *et al.*, 1994; Solberg *et al.*, ] and [Solberg, 1995a] Chapter 2) the inference of s-types and the inference of t-types are independent. This implies that using a system only for strictness and one only for totality we can infer properties of terms that (w.r.t. the possibility of replacing safely a lazy application with a strict one) give the same information given by the properties inferred by the systems for combined strictness and totality.

# Chapter 5

# Conjunctive strictness and totality analyses

In this chapter we extend the results of Chapter 4 to strictness and totality analyses with conjunction. Given two different st-types $\phi_1$ and $\phi_2$ (such that $\epsilon(\phi_1) = \epsilon(\phi_2)$) the use conjunction allows to assign both them (i.e. $\phi_1 \wedge \phi_2$) to a same function $f$.

For a simple example[1] of use of conjunction consider the term

$$G = (\lambda f^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\mathsf{if}\ x = 0\ \mathsf{then}\ f1y\ \mathsf{else}\ fy1)(\lambda u^{\mathsf{nat}}.\lambda v^{\mathsf{nat}}.u + v)$$

of type $\rho = \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$. It is easy to see that $G$ is strict in its second argument, i.e. it satisfies the property $\phi_1 = \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$. However, with the system $\vdash_{\mathrm{s}}$ of Section 4.2 it is not possible to assign the property $\phi_1$ to the program $G$. In fact, to do this we need to assign s-type $\phi_1$ to the first occurrence of $f$ in the body of the $\lambda$-abstraction

$$\lambda f^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\mathsf{if}\ x = 0\ \mathsf{then}\ f1y\ \mathsf{else}\ fy1$$

in $G$, and to assign s-type $\phi_2 = \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$ to the second occurrence of $f$. In $\vdash_{\mathrm{s}}$ this could be done only by assuming for the $\lambda$-abstracted variable $f$ a property $\phi$ which entails both $\phi_1$ and $\phi_2$. But, in $\mathbf{L}^{\mathrm{s}}$ there is no property of $\lambda u^{\mathsf{nat}}.\lambda v^{\mathsf{nat}}.u + v$ which entails both $\phi_1$ and $\phi_2$. As we will see in Example 5.22, the conjunctive strictness type assignment system presented in Section 5.2 allows to assign $\phi_1$ to $G$.

The first section of this chapter introduces a language of strictness and totality types with conjunction and its semantics. Sections 5.2 and 5.3 introduce respectively a conjunctive strictness type assignment system and a conjunctive totality type assignment system for PCFP terms.

In Section 5.4 we introduce an assignment system for combined strictness and totality, $\vdash_{\mathrm{fcst}}$ (which is an extension of the one in [Solberg, 1995b] and [Solberg, 1995a] Chapter 3), and prove a separability result analogous to the one of Section 4.4.5.

[1] The term $G$ was introduced in [Benton, 1992a] page 32 to point out that the type system presented in [Kuo and Mishra, 1989] gives weaker results than abstract interpretation in the style of [Burn *et al.*, 1986] even for non-recursive terms. Here we are using it to show a limitation of the system $\vdash_{\mathrm{s}}$ of Section 4.2.

$$(a)\ \frac{a \in \{\bot, \not\bot, \top\} \quad \rho \in \mathbf{T}}{a^\rho \in \mathbf{L}^{\mathrm{cst}}(\rho)} \qquad (\to)\ \frac{\xi \in \mathbf{C}^{\mathrm{cst}}(\rho) \quad \psi \in \mathbf{L}^{\mathrm{cst}}(\sigma)}{\xi \to \psi \in \mathbf{L}^{\mathrm{cst}}(\rho \to \sigma)}$$

$$(\times)\ \frac{\phi_1 \in \mathbf{L}^{\mathrm{cst}}(\rho_1) \quad \phi_2 \in \mathbf{L}^{\mathrm{cst}}(\rho_2)}{\phi_1 \times \phi_2 \in \mathbf{L}^{\mathrm{cst}}(\rho_1 \times \rho_2)} \qquad (\wedge)\ \frac{\phi_1, \ldots, \phi_n \in \mathbf{L}^{\mathrm{cst}}(\rho)}{\phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{C}^{\mathrm{cst}}(\rho)}\ n \geq 1$$

Figure 5.1: Conjunctive strictness and totality types

## 5.1 Conjunctive strictness and totality types

In the cst-types of Definition 5.1 and in the systems in the next two sections we will *not* allow conjunction at the top level and in the right part of the arrow operator. This restriction on the conjunction operator is essentially the one proposed by van Bakel (see [van Bakel, 1992] and [van Bakel, 1993] Chapter 4) and then used for strictness analysis in [Hankin and Le Métayer, 1994b]. As pointed out in [Hankin and Le Métayer, 1994b] (see also [Jensen, 1992], Theorem 3.3.2), in the case of strictness analysis, these restriction makes the type manipulation easier without affecting the power of the logic. As a consequence of the separability result of Section 5.4.5 we get that this is also true for a "full conjunctive" strictness and totality type assignment (which includes new cst-type constructors to reason about monotonicity properties and allows conjunction to occur also in the right side of the arrow constructor) that extends the one proposed in [Solberg, 1995a] Chapter 3 and [Solberg, 1995b].

**Definition 5.1 (Conjunctive strictness and totality types)** *The language of* conjunctive strictness and totality types *(*cst-types *for short)* $\mathbf{L}^{\mathrm{cst}}$*, ranged over by* $\phi$ *and* $\psi$ *(with superscripts and subscripts if needed), and the set of* conjunctions of cst-types *(*cst-conjunctions *for short)* $\mathbf{C}^{\mathrm{cst}}$*, ranged over by* $\xi$*, are defined by:*

$$\mathbf{L}^{\mathrm{cst}} = \bigcup_{\rho \in \mathbf{T}} \mathbf{L}^{\mathrm{cst}}(\rho) \qquad\qquad \mathbf{C}^{\mathrm{cst}} = \bigcup_{\rho \in \mathbf{T}} \mathbf{C}^{\mathrm{cst}}(\rho)\ ,$$

*where the sets* $\mathbf{L}^{\mathrm{cst}}(\rho)$ *and* $\mathbf{C}^{\mathrm{cst}}(\rho)$ *are defined by the rules in Fig. 5.1. Note that* $\mathbf{L}^{\mathrm{cst}} \subset \mathbf{C}^{\mathrm{cst}}$*.*

Since cst-types are properties of terms, in the following we use the words cst-type and property interchangeably.

The meaning of cst-types is formalized by interpreting each cst-type $\phi$ as a subset of the interpretation of the type $\epsilon(\phi)$. In particular the interpretation of the basic properties is as in Section 4.1. I.e. $\top^\rho$ (which denotes absence of information) is interpreted as the whole interpretation of the type $\rho$ (meaning that we do not have any additional information on terms except their being of type $\rho$); $\bot^\rho$ (which denotes divergence) is interpreted as the singleton set containing the $\simeq^{\mathrm{lazy}}_{\mathrm{obs}}$-class of the closed terms of type $\rho$ not having a w.h.n.f., and $\not\bot^\rho$ (which denotes convergence) is interpreted as the set of the $\simeq^{\mathrm{lazy}}_{\mathrm{obs}}$-classes of the closed terms of type $\rho$ having a w.h.n.f..

**Definition 5.2 (Semantics of cst-types)** *The interpretation* $[\![\phi]\!]^{\mathcal{M}}$ *of a cst-type* $\phi$ *in the model* $\mathcal{M} \in \{\mathcal{M}^{\mathrm{lazy}}, \mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}\}$ *is defined by:*

$$\begin{aligned}
[\![\top^\rho]\!]^{\mathcal{M}} &= \mathbf{I}(\rho)^{\mathcal{M}} \\
[\![\bot^\rho]\!]^{\mathcal{M}} &= \{[\mathsf{fix}\, x^\rho.x]^{\mathcal{M}}\} \\
[\![\not\downarrow^\rho]\!]^{\mathcal{M}} &= \mathbf{I}(\rho)^{\mathcal{M}} - \{[\mathsf{fix}\, x^\rho.x]^{\mathcal{M}}\} \\
[\![\xi \to \psi]\!]^{\mathcal{M}} &= \{[M]^{\mathcal{M}} \mid \forall [N]^{\mathcal{M}} \in [\![\xi]\!]^{\mathcal{M}}.[MN]^{\mathcal{M}} \in [\![\psi]\!]^{\mathcal{M}}\} \\
[\![\phi_1 \times \phi_2]\!]^{\mathcal{M}} &= \{[M]^{\mathcal{M}} \mid \forall i \in \{1,2\}.[\mathsf{proj}_i M]^{\mathcal{M}} \in [\![\phi_i]\!]^{\mathcal{M}}\},
\end{aligned}$$

*where the interpretation of an cst-conjunction* $\phi_1 \wedge \cdots \wedge \phi_n$ *is defined by:*

$$[\![\phi_1 \wedge \cdots \wedge \phi_n]\!]^{\mathcal{M}} = \textstyle\bigcap_{1 \le i \le n} [\![\phi_i]\!]^{\mathcal{M}} \ .$$

In the rest of the chapter, whenever a statement holds for both the interpretations $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}$, we will not use the superscript, i.e. we will simply write $\mathbf{I}(\rho)$, $[M]$, $[\![\phi]\!]$.

The cst-types can be classified according to their rightmost (w.r.t. the arrow type constructor) basic property constructors. In particular cst-types whose rightmost constructors are non informative ($\top$) are themselves non informative, they are called $\top$-*cst-types*. The properties of being a term that, when applied to some arguments and/or projected (i.e. passed as argument to $\mathsf{proj}_1$ or $\mathsf{proj}_2$) a certain number (possibly 0) of times, result in a divergent (convergent) computation are represented by $\bot$-*cst-types* ($\not\downarrow$-*cst-types* ).

**Definition 5.3 ($\top$-cst-types, $\bot$-cst-types and $\not\downarrow$-cst-types)** *The set* $\mathbf{L}^{\mathrm{cst}}_{\top}$ *of* $\top$*-cst-types is defined by:*

- $\top^\rho \in \mathbf{L}^{\mathrm{cst}}_{\top}$*, for* $\rho \in \mathbf{T}$
- $\xi \to \psi \in \mathbf{L}^{\mathrm{cst}}_{\top}$*, for* $\xi \in \mathbf{C}^{\mathrm{cst}}$ *and* $\psi \in \mathbf{L}^{\mathrm{cst}}_{\top}$
- $\phi_1 \times \phi_2 \in \mathbf{L}^{\mathrm{cst}}_{\top}$*, for* $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{cst}}_{\top}$.

*The sets* $\mathbf{L}^{\mathrm{cst}}_{\bot}$ *of* $\bot$*-cst-types and* $\mathbf{L}^{\mathrm{cst}}_{\not\downarrow}$ *of* $\not\downarrow$*-cst-types are defined in a similar way with respectively* $\bot$ *and* $\not\downarrow$ *instead of* $\top$.

**Proposition 5.4** $\phi \in \mathbf{L}^{\mathrm{cst}}_{\top}$ *implies* $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$.

**Proof.** By structural induction on $\phi \in \mathbf{L}^{\mathrm{cst}}_{\top}$. The case $\phi = \top^\rho$, for some $\rho \in \mathbf{T}$, is immediate by Definition 5.2.
For $\phi = \xi \to \psi$, we have that $\phi \in \mathbf{L}^{\mathrm{cst}}_{\top}$ implies (by Definition 5.3) $\psi \in \mathbf{L}^{\mathrm{cst}}_{\top}$ implies (by induction hypothesis) $[\![\psi]\!] = \mathbf{I}(\epsilon(\psi))$ implies (by Definition 5.2) $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$.
For $\phi = \phi_1 \times \phi_2$, we have that $\phi \in \mathbf{L}^{\mathrm{cst}}_{\top}$ implies (by Definition 5.3) $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{cst}}_{\top}$ implies (by induction hypothesis) $[\![\phi_1]\!] = [\![\phi_2]\!] = \mathbf{I}(\epsilon(\psi))$ implies (by Definition 5.2) $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$. □

The set of $\bot\top$-cst-types (in Definition 5.5) collects some of (but not all, as explained in Remark 5.7) the properties that are satisfied by terms without w.h.n.f..

**Definition 5.5 ($\bot\top$-cst-types)** *The set $\mathbf{L}^{\mathrm{cst}}_{\bot\top}$ of $\bot\top$-cst-types is defined by:*

- $\phi \in \mathbf{L}^{\mathrm{cst}}_{\bot\top}$, *for* $\phi \in \mathbf{L}^{\mathrm{cst}}_{\bot} \cup \mathbf{L}^{\mathrm{cst}}_{\top}$
- $\xi \to \psi \in \mathbf{L}^{\mathrm{cst}}_{\bot\top}$, *for* $\xi \in \mathbf{C}^{\mathrm{cst}}$ *and* $\psi \in \mathbf{L}^{\mathrm{st}}_{\bot\top}$
- $\phi_1 \times \phi_2 \in \mathbf{L}^{\mathrm{cst}}_{\bot\top}$, *for* $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{cst}}_{\bot\top}$.

The sets $\mathbf{L}^{\mathrm{cst}}_{\top}$, $\mathbf{L}^{\mathrm{cst}}_{\bot}$ and $\mathbf{L}^{\mathrm{cst}}_{\not\bot}$ are pairwise disjoint, $\mathbf{L}^{\mathrm{cst}}_{\bot} \cup \mathbf{L}^{\mathrm{cst}}_{\top} \subset \mathbf{L}^{\mathrm{cst}}_{\bot\top}$, and $\mathbf{L}^{\mathrm{cst}}_{\not\bot} \cap \mathbf{L}^{\mathrm{cst}}_{\bot\top} = \emptyset$.

**Proposition 5.6** $\phi \in \mathbf{L}^{\mathrm{cst}}_{\bot\top}$ *implies* $[\![\bot^{\epsilon(\phi)}]\!] \subseteq [\![\phi]\!]$.

**Proof.** Similar to the proof of Proposition 5.4. □

**Remark 5.7 (Empty cst-conjunctions)** *It is worth mentioning that the analogous of Proposition 4.6.2 does not hold for cst-types, in fact it is* not *true that:*

$$\phi \in \mathbf{L}^{\mathrm{cst}} - \mathbf{L}^{\mathrm{cst}}_{\bot\top} \text{ implies } [\![\phi]\!] \subseteq [\![\not\bot^{\epsilon(\phi)}]\!]. \tag{5.1}$$

*This is due to the fact that there are cst-conjunctions whose interpretation is empty, like $\not\bot^{\mathsf{nat}} \wedge \bot^{\mathsf{nat}}$. It is easy to see that for every $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{st}}(\rho)$ and $\psi \in \mathbf{L}^{\mathrm{st}}(\sigma)$ we have that $[\![\phi_1]\!] \cap [\![\phi_2]\!] = \emptyset$ implies $[\![\phi_1 \wedge \phi_2 \to \psi]\!] = [\![\top^{\rho\to\sigma}]\!]$. For instance: $[\![(\bot^\rho \wedge \not\bot^\rho) \to \not\bot^\sigma]\!] = \mathbf{I}(\rho \to \sigma)$, and so $[\![(\bot^\rho \wedge \not\bot^\rho) \to \not\bot^\sigma]\!] \not\subseteq [\![\not\bot^{\rho\to\sigma}]\!]$. For the same reason the reverse of the implications in Propositions 5.4 and 5.6, and the reverse of the implication (5.1) do not hold.*

*Notice that cs-conjunctions (the elements of $\mathbf{C}^{\mathrm{cst}}$ which do not contain the constructor $\not\bot$) and ct-conjunctions (the elements of $\mathbf{C}^{\mathrm{cst}}$ which do not contain the constructor $\bot$) are always not empty. In fact all cs-types (the elements of $\mathbf{L}^{\mathrm{cst}}$ which do not contain the constructor $\not\bot$) contain the equivalence class of the always divergent computation of the corresponding type, and all ct-types (the elements of $\mathbf{L}^{\mathrm{cst}}$ which do not contain the constructor $\bot$) contain the observational equivalence classes of the always convergent terms of the corresponding type. By "always convergent terms" of type $\rho$ we mean all the terms $M$ such that $[M] \in [\![\mathbf{min}_{\mathrm{t}}(\rho)]\!]$, where $\mathbf{min}_{\mathrm{t}}(\rho)$ is the ct-type inductively defined as follows:*

- $\mathbf{min}_{\mathrm{t}}(\iota) = \not\bot^\iota$, *if* $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$,
- $\mathbf{min}_{\mathrm{t}}(\rho_1 \to \rho_2) = \top^{\rho_1} \to \mathbf{min}_{\mathrm{t}}(\rho_2)$, *and*
- $\mathbf{min}_{\mathrm{t}}(\rho_1 \times \rho_2) = \mathbf{min}_{\mathrm{t}}(\rho_1) \times \mathbf{min}_{\mathrm{t}}(\rho_2)$.

As we will see in Section 5.4, this language of non-standard types is indeed quite difficult to study, owing to the rather complex interactions of the strictness and totality properties expressed by the non-standard types. Moreover, in Section 5.4.5, we will show that this interactions are rather weak when considering the cst-types that can be assigned to terms. For this reason in the next two sections we study strictness and totality properties, and their assignment to PCFP terms, separately.

$$(\bot)\ \bot^{\epsilon(\phi)} \leq_{\mathrm{cs}} \phi \quad (\top)\ \frac{\psi \in \mathbf{L}^{\mathrm{cs}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq_{\mathrm{cs}} \psi}$$

$$(\mathrm{Ref})\ \phi \leq_{\mathrm{cs}} \phi \quad (\to)\ \frac{\xi_2 \leq_{\mathrm{cs}} \xi_1 \quad \phi_1 \leq_{\mathrm{cs}} \phi_2}{\xi_1 \to \phi_1 \leq_{\mathrm{cs}} \xi_2 \to \phi_2} \quad (\times)\ \frac{\phi_1 \leq_{\mathrm{cs}} \psi_1 \quad \psi_1 \leq_{\mathrm{cs}} \psi_2}{\phi_1 \times \phi_2 \leq_{\mathrm{cs}} \psi_2 \times \psi_2}$$

$$(\wedge)\ \frac{\forall i \in \{1,\ldots,m\}.\exists j \in \{1,\ldots,n\}.\phi_j \leq_{\mathrm{cs}} \psi_i}{\phi_1 \wedge \cdots \wedge \phi_n \leq_{\mathrm{cs}} \psi_1 \wedge \cdots \wedge \psi_m}$$

Figure 5.2: Entailment rules for cs-types (system $\leq_{\mathrm{cs}}$)

## 5.2 Conjunctive strictness analysis

In this section we extend the strictness analysis of Section 4.2 by allowing the use of conjunction.

Let $\mathbf{L}^{\mathrm{cs}}$ and $\mathbf{C}^{\mathrm{cs}}$ be respectively the sets of cst-types and cst-conjunctions that do not contain the constructor $\not\perp$. We call the properties in $\mathbf{L}^{\mathrm{cs}}$ *conjunctive strictness types* (*cs-types* for short) and the properties in $\mathbf{C}^{\mathrm{cs}}$ *conjunctions of cs-types* (*cs-conjunctions*). Note that $\mathbf{L}^{\mathrm{cs}} \subset \mathbf{C}^{\mathrm{cs}}$.

### 5.2.1 An entailment relation for conjunctive strictness types

**Definition 5.8 (Entailment relation for strictness types)** *Let $\xi_1, \xi_2 \in \mathbf{C}^{\mathrm{cs}}$. We write $\xi_1 \leq_{\mathrm{cs}} \xi_2$ to mean that $\xi_1 \leq_{\mathrm{cs}} \xi_2$ is derivable by the rules in Fig. 5.2. By $\cong_{\mathrm{cs}}$ we denote the equivalence relation induced by $\leq_{\mathrm{cs}}$.*

Notice that the relation $\leq_{\mathrm{cs}}$ is reflexive and transitive, since rule (Trans) in Fig. 3.4 is admissible.

Soundness of entailment for cs-types (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$) is stated by the following theorem.

**Theorem 5.9 (Soundness of $\leq_{\mathrm{cs}}$)** *Let $\xi_1, \xi_2 \in \mathbf{C}^{\mathrm{cs}}$. $\xi_1 \leq_{\mathrm{cs}} \xi_2$ implies $[\![\xi_1]\!] \subseteq [\![\xi_2]\!]$.*

**Proof.** By induction on the derivation of the entailment judgment. □

**Proposition 5.10** *$[\![\xi]\!] = [\![\top^{\epsilon(\xi)}]\!]$ if and only if $\phi_1, \ldots, \phi_n \in \mathbf{L}^{\mathrm{cs}}_{\top}$ if and only if $\xi \cong_{\mathrm{cs}} \top^{\epsilon(\xi)}$.*

**Proof.**

- $[\![\xi]\!] = [\![\top^{\epsilon(\xi)}]\!]$ implies $\phi_1, \ldots, \phi_n \in \mathbf{L}^{\mathrm{cs}}_{\top}$.
  We have that $[\![\xi]\!] = \mathbf{I}(\epsilon(\xi))$ implies $[\![\phi_1]\!] = \cdots = [\![\phi_n]\!] = \mathbf{I}(\epsilon(\xi))$, so it suffices to prove that, for $\phi \in \mathbf{L}^{\mathrm{cs}}$, $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$ implies $\phi \in \mathbf{L}^{\mathrm{cs}}_{\top}$.

By structural induction on $\phi \in \mathbf{L}^{\mathrm{cs}}$. The case $\phi \in \{\top^\rho, \bot^\rho\}$, for some $\rho \in \mathbf{T}$, is immediate.
For $\phi = \nu \to \psi$, we have that $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$ implies (by Definition 5.2) $[\![\psi]\!] = \mathbf{I}(\epsilon(\psi))$ implies (by induction hypothesis) $\psi \in \mathbf{L}^{\mathrm{cs}}_\top$ implies $\phi \in \mathbf{L}^{\mathrm{cs}}_\top$.
For $\phi = \phi_1 \times \phi_2$, we have that $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$ implies (by Definition 5.2), $[\![\phi_1]\!] = \mathbf{I}(\epsilon(\phi_1))$ and $[\![\phi_2]\!] = \mathbf{I}(\epsilon(\phi_2))$ imply (by induction hypothesis) $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{cs}}_\top$ implies $\phi \in \mathbf{L}^{\mathrm{cs}}_\top$.

- $\underline{\phi_1, \ldots, \phi_n \in \mathbf{L}^{\mathrm{cs}}_\top \text{ implies } \xi \cong_{\mathrm{cs}} \top^{\epsilon(\xi)}}$.
  Immediate by rules $(\top)$ and $(\wedge)$ of $\leq_{\mathrm{cs}}$.

- $\underline{\xi \cong_{\mathrm{cs}} \top^{\epsilon(\xi)} \text{ implies } [\![\xi]\!] = [\![\top^{\epsilon(\xi)}]\!]}$.
  Immediate by Theorem 5.9. □

The entailment rules of Definition 5.8 are not complete w.r.t. the semantics $\mathcal{M}^{\mathrm{lazy}}$, as the following example shows.

**Example 5.11 (Incompleteness of $\leq_{\mathrm{cs}}$ w.r.t. $\mathcal{M}^{\mathrm{lazy}}$)** *Let* $\phi_1 = \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$, $\phi_2 = \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$, *and* $\psi = \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$. *First note that* $[\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}} = [\![\phi_1]\!]^{\mathcal{M}^{\mathrm{lazy}}} \cup [\![\phi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}}$. *That is: for all term* $M$ *of type* $\mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$,
$M(\mathsf{fix}\, x^{\mathsf{nat}}.x)(\mathsf{fix}\, x^{\mathsf{nat}}.x) \Uparrow$ *if and only if*

- *either for all term* $Q$ *of type* $\mathsf{nat}$, $M(\mathsf{fix}\, x^{\mathsf{nat}}.x)Q \Uparrow$
- *or for all term* $P$ *of type* $\mathsf{nat}$, $MP(\mathsf{fix}\, x^{\mathsf{nat}}.x) \Uparrow$

*This can be proved in a way similar to the proof of the activity lemma of [Plotkin, 1977]. Now* $[\![\phi_1 \to \bot^{\mathsf{nat}} \wedge \phi_2 \to \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} = [\![\phi_1 \to \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} \cap [\![\phi_2 \to \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}}$, *and* $[\![\phi_1 \to \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} \cap [\![\phi_2 \to \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} = \{[M]^{\mathcal{M}^{\mathrm{lazy}}} \mid \forall\, [N]^{\mathcal{M}^{\mathrm{lazy}}} \in [\![\phi_1]\!]^{\mathcal{M}^{\mathrm{lazy}}} \cup [\![\phi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}}.[MN]^{\mathcal{M}^{\mathrm{lazy}}} \in [\![\bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}}\}$, *and since* $[\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}} = [\![\phi_1]\!]^{\mathcal{M}^{\mathrm{lazy}}} \cup [\![\phi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}}$ *we have that*

$$[\![\phi_1 \to \bot^{\mathsf{nat}} \wedge \phi_2 \to \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}} = [\![\psi \to \bot^{\mathsf{nat}}]\!]^{\mathcal{M}^{\mathrm{lazy}}}.$$

*However,* $\psi \not\leq_{\mathrm{cs}} \phi_1$ *and* $\psi \not\leq_{\mathrm{cs}} \phi_2$. *So* $\phi_1 \to \bot^{\mathsf{nat}} \not\leq_{\mathrm{cs}} \psi \to \bot^{\mathsf{nat}}$ *and* $\phi_2 \to \bot^{\mathsf{nat}} \not\leq_{\mathrm{cs}} \psi \to \bot^{\mathsf{nat}}$. *Therefore it is easy to see that*

$$\phi_1 \to \bot^{\mathsf{nat}} \wedge \phi_2 \to \bot^{\mathsf{nat}} \not\leq_{\mathrm{cs}} \psi \to \bot^{\mathsf{nat}}.$$

*The failure of completeness is then due to the "sequential" nature of the evaluation rules of the programming language PCFP.*

Another reason for incompleteness is that in $\Lambda_{\mathbf{T}}$ the functional terms with w.h.n.f. cannot be discriminated, i.e. the argument of Example 4.10 apply also to $\leq_{\mathrm{cs}}$.

The entailment $\leq_{\mathrm{cs}}$ is complete w.r.t. the term model $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$ of Section 2.4.2. To our knowledge this completeness result is new.

**Theorem 5.12 (Completeness of $\leq_{\mathrm{cs}}$ w.r.t. $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}$)** *Let $\xi_1, \xi_2 \in \mathbf{C}^{\mathrm{cs}}$. $[\![\xi_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \subseteq [\![\xi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ implies that $\xi_1 \leq_{\mathrm{cs}} \xi_2$.* □

To prove completeness we define, for each cs-type $\phi$ (and for each conjunction of $n \geq 2$ cs-types $\xi$), a term $\mathrm{M}^{\phi}$ (resp. $\mathrm{M}^{\xi}$), that characterizes the property $\phi$ ($\xi$).

**Notation 5.13** *1. We say that a conjunction of $n \geq 1$ cs-types, $\phi_1 \wedge \cdots \wedge \phi_n$, is* normal *if either $n = 1$ or $n \geq 2$ and, for all $i$, $1 \leq i \leq n$, $\phi_i \not\cong_{\mathrm{cs}} \top^{\rho}$ and $\phi_i \not\cong_{\mathrm{cs}} \bot^{\rho}$. Note that every cs-conjunction $\xi$ is equivalent ($\cong_{\mathrm{cs}}$) to a normal cs-conjunction since it is possible to eliminate the components equivalent ($\cong_{\mathrm{cs}}$) to $\top^{\rho}$, and if $\bot^{\rho}$ is a member of a cs-conjunction then the whole conjunction is equivalent ($\cong_{\mathrm{cs}}$) to $\bot^{\rho}$. Define* $\mathbf{norm}(\xi)$ *be the cs-conjunction obtained from $\xi$ in this way.*

*2. In the following definition we will use an $n$-ary strict* and *operator,* $\mathsf{and}(M_1, \ldots, M_n)$*, ($n \geq 0$) which can be easily defined by using the binary strict operator* $\mathsf{and}^{\mathsf{bool}\times\mathsf{bool}\to\mathsf{bool}}$ *introduced as a constant of the language PCFP in Section 2.1. The* and *of 0 arguments,* $\mathsf{and}()$*, evaluates to* true*.*

*3. Let $\mathrm{C}^{\rho}_{\top}$ be defined by: $\mathrm{C}^{\mathsf{nat}}_{\top} = 0$, $\mathrm{C}^{\mathsf{bool}}_{\top} = \mathsf{true}$, $\mathrm{C}^{\rho_1\to\rho_2}_{\top} = \lambda z^{\rho_1}.C^{\rho_2}$, and $\mathrm{C}^{\rho_1\times\rho_2}_{\top} = \langle \mathrm{C}^{\rho_1}_{\top}, \mathrm{C}^{\rho_2}_{\top} \rangle$.*

The characteristic terms $\mathrm{M}^{\phi}$ and $\mathrm{M}^{\xi}$ are such that, for every cs-type $\psi$ and cs-conjunction $\nu$,

- $[\mathrm{M}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ if and only if $\phi \leq_{\mathrm{cs}} \psi$,
- $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ if and only if $\xi \leq_{\mathrm{cs}} \nu$.

We also need to define

- a test term $\mathrm{T}^{\phi}$ ($\mathrm{T}^{\xi}$) that diverges whenever applied to the characteristic term of a cs-type (cs-conjunction) which does not entail ($\leq_{\mathrm{cs}}$) $\phi$ ($\xi$),
- a term $\mathrm{B}^{\psi}$ which is used, in the definition of the characteristic terms for cs-types (in Fig. 5.3), to delay the evaluation of the application of the test terms to the arguments of a characteristic term until all these arguments have been collected and can be evaluated in "parallel or" (por), and
- a term $\mathrm{D}^{\sigma;\psi_1,\ldots,\psi_n}$ which is used, in the definition of the characteristic terms for normal conjunctions $\xi_1 \to \psi_1 \wedge \cdots \wedge \xi_n \to \psi_n$ of $n \geq 2$ cs-types (in Fig. 5.4 point 2.), to delay the evaluation of the application of the characteristic terms (of each cs-type $\xi_i \to \psi_i$, $1 \leq i \leq n$ of the conjunction) until all the arguments required (by some of these characteristic terms) have been collected.

Such terms are defined by mutual induction on cs-types.

**Definition 5.14** *Let $\phi$ be a cs-type and $\xi$ be a conjunction of $n \geq 2$ cs-types. The terms $\mathrm{M}^{\phi}$, $\mathrm{T}^{\phi}$, $\mathrm{M}^{\xi}$, and $\mathrm{T}^{\xi}$ are defined according to the clauses in Fig.s 5.3 and 5.4, where (for sake of readability) the* else *branches associated to conditions that never evaluates to* false *(i.e. either divergent or evaluating to* true*) are filled with the symbol $\sharp$.*

**Example 5.15** *Take the (normal) conjunction $\xi = \phi_1 \wedge \phi_2 \wedge \phi_3$, where*

1. $\phi_1 = (\bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \psi_1$,
   *and* $\psi_1 = \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$;
2. $\phi_2 = (\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \psi_2$,
   *and* $\psi_2 = \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$;
3. $\phi_3 = (\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \psi_3$,
   *and* $\psi_3 = \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}\to\mathsf{nat}}$.

*In the following we will write (for short) $\bot$ instead of $\bot^{\mathsf{nat}}$ and $\top$ instead of $\top^{\mathsf{nat}}$.*
*According to Definition 5.14 we have*

$$\mathrm{M}^{\xi} = \lambda x^{\mathsf{nat}\to\mathsf{nat}}.\mathrm{D}^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat};\psi_1,\psi_2,\psi_3}(\mathrm{M}^{\phi_1}x)(\mathrm{M}^{\phi_2}x)(\mathrm{M}^{\phi_3}x)$$

*where*

1. $\mathrm{M}^{\phi_1} = \lambda x^{\mathsf{nat}\to\mathsf{nat}}.\mathrm{B}^{\psi_1}(\mathrm{T}^{\bot\to\bot}x)$
   - $\mathrm{B}^{\psi_1} = \lambda b^{\mathsf{bool}}.\lambda x^{\mathsf{nat}}.\mathrm{B}^{\top\to\bot}(\mathsf{por}(b, \mathrm{T}^{\bot}x))$
     *and* $\mathrm{B}^{\top\to\bot} = \lambda b^{\mathsf{bool}}.\lambda x^{\mathsf{nat}}.\mathrm{B}^{\bot}(\mathsf{por}(b, \mathrm{T}^{\top}x))$
   - $\mathrm{T}^{\bot\to\bot} = \lambda z^{\mathsf{nat}}.\mathrm{T}^{\bot}(z\mathrm{M}^{\bot})$;
2. $\mathrm{M}^{\phi_2} = \lambda x^{\mathsf{nat}\to\mathsf{nat}}.\mathrm{B}^{\psi_2}(\mathrm{T}^{\top\to\bot}x)$
   - $\mathrm{B}^{\psi_2} = \lambda b^{\mathsf{bool}}.\lambda x^{\mathsf{nat}}.\mathrm{B}^{\bot\to\bot}(\mathsf{por}(b, \mathrm{T}^{\top}x))$
     *and* $\mathrm{B}^{\bot\to\bot} = \lambda b^{\mathsf{bool}}.\lambda x^{\mathsf{nat}}.\mathrm{B}^{\bot}(\mathsf{por}(b, \mathrm{T}^{\bot}x))$
   - $\mathrm{T}^{\top\to\bot} = \lambda z^{\mathsf{nat}}.\mathrm{T}^{\bot}(z\mathrm{M}^{\top})$;
3. $\mathrm{M}^{\phi_3} = \lambda x^{\mathsf{nat}\to\mathsf{nat}}.\mathrm{B}^{\psi_3}(\mathrm{T}^{\bot^{\mathsf{nat}\to\mathsf{nat}}}x)$
   - $\mathrm{B}^{\psi_2} = \lambda b^{\mathsf{bool}}.\lambda x^{\mathsf{nat}}.\mathrm{B}^{\bot^{\mathsf{nat}\to\mathsf{nat}}}(\mathsf{por}(b, \mathrm{T}^{\top}x))$;
4. $\mathrm{D}^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat};\psi_1,\psi_2,\psi_3} = \lambda p_1^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.\lambda p_2^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.\lambda p_3^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.$ if $\mathsf{and}()$
   then $\lambda x^{\mathsf{nat}}.\mathrm{D}^{\mathsf{nat}\to\mathsf{nat};\top\to\bot,\bot\to\bot,\bot^{\mathsf{nat}\to\mathsf{nat}}}(p_1x)(p_2x)(p_3x)$ else $\sharp$
   - $\mathrm{D}^{\mathsf{nat}\to\mathsf{nat};\top\to\bot,\bot\to\bot,\bot^{\mathsf{nat}\to\mathsf{nat}}} = \lambda q_1^{\top\to\bot}.\lambda q_2^{\bot\to\bot}.\lambda q_3^{\bot^{\mathsf{nat}\to\mathsf{nat}}}.$ if $\mathsf{and}(\mathsf{isdef}(q_3))$
     then $\lambda y^{\mathsf{nat}}.\mathrm{D}^{\mathsf{nat};}$ else $\sharp$.

- If $\phi \cong_{\mathrm{cs}} \top^{\rho}$, then
  - $\mathrm{M}^{\phi} = \mathrm{C}^{\rho}_{\top}$
  - $\mathrm{T}^{\phi} = \lambda z^{\rho}.\mathsf{fix}\, x^{\rho}.x$
- If $\phi = \bot^{\rho}$, then
  - $\mathrm{M}^{\phi} = \mathsf{fix}\, x^{\rho}.x$
  - $\mathrm{T}^{\phi} = \lambda z^{\rho}.\mathsf{isdef}(z)$
- If $\phi = \phi_1 \times \phi_2 \ncong_{\mathrm{cs}} \top^{\epsilon(\phi)}$, then
  - $\mathrm{M}^{\phi} = \langle \mathrm{M}^{\phi_1}, \mathrm{M}^{\phi_2} \rangle$
  - $\mathrm{T}^{\phi} = \lambda z^{\epsilon(\phi)}.\mathsf{por}(\mathrm{T}^{\phi_1}(\mathsf{proj}_1 z), \mathrm{T}^{\phi_2}(\mathsf{proj}_2 z))$
- If $\phi = \xi \to \psi \ncong_{\mathrm{cs}} \top^{\epsilon(\phi)}$, then
  - $\mathrm{M}^{\phi} = \lambda x^{\epsilon(\xi)}.\mathrm{B}^{\psi}(\mathrm{T}^{\xi} x)$
  - $\mathrm{T}^{\phi} = \lambda z^{\epsilon(\phi)}.\mathrm{T}^{\psi}(z\mathrm{M}^{\xi})$

  where the term $\mathrm{B}^{\psi}$ is inductively defined as follows.
- If $\psi \cong_{\mathrm{cs}} \top^{\sigma}$, then
  - $\mathrm{B}^{\psi} = \lambda b^{\mathsf{bool}}.\mathrm{C}^{\sigma}_{\top}$
- If $\psi = \bot^{\sigma}$, then
  - $\mathrm{B}^{\psi} = \lambda b^{\mathsf{bool}}.\mathsf{if}\ b\ \mathsf{then}\ \mathrm{C}^{\sigma}_{\top}\ \mathsf{else}\ \sharp$
- If $\psi = \psi_1 \times \psi_2 \ncong_{\mathrm{cs}} \top^{\epsilon(\psi)}$, then
  - $\mathrm{B}^{\psi} = \lambda b^{\mathsf{bool}}.\langle \mathrm{B}^{\psi_1} b, \mathrm{B}^{\psi_2} b \rangle$
- If $\psi = \nu \to \chi \ncong_{\mathrm{cs}} \top^{\epsilon(\psi)}$, then
  - $\mathrm{B}^{\psi} = \lambda b^{\mathsf{bool}}.\lambda x^{\epsilon(\nu)}.\mathrm{B}^{\chi}(\mathsf{por}(b, \mathrm{T}^{\nu} x))$

Figure 5.3: Terms $\mathrm{M}^{\phi}$ and $\mathrm{T}^{\phi}$

Let $\mathbf{norm}(\xi) = \phi_1 \wedge \cdots \wedge \phi_n$. If $n = 1$ then $\mathrm{M}^{\xi} = \mathrm{M}^{\phi_1}$ and $\mathrm{T}^{\xi} = \mathrm{T}^{\phi_1}$.
Otherwise ($n \geq 2$)

$$\mathrm{T}^{\xi} = \lambda z^{\epsilon(\xi)}.\mathsf{por}((\mathrm{T}^{\phi_1} z), \ldots, (\mathrm{T}^{\phi_n} z))$$

and for $\mathrm{M}^{\xi}$ there are 2 cases.

1. If $\phi_i = \psi_i \times \psi_i'$ $(1 \leq i \leq n)$ then
   - $\mathrm{M}^{\xi} = \langle \mathrm{M}^{\psi_1 \wedge \cdots \wedge \psi_n}, \mathrm{M}^{\psi_1' \wedge \cdots \wedge \psi_n'} \rangle$
2. If $\phi_i = \xi_i \to \psi_i$ $(1 \leq i \leq n)$ then
   - $\mathrm{M}^{\xi} = \lambda x^{\epsilon(\xi_1)}.\mathrm{D}^{\epsilon(\psi_1);\psi_1,\ldots,\psi_n}(\mathrm{M}^{\xi_1 \to \psi_1} x) \cdots (\mathrm{M}^{\xi_n \to \psi_n} x)$

where the term $\mathrm{D}^{\sigma;\psi_1,\ldots,\psi_n}$ ($n \geq 0$ and $\sigma = \epsilon(\psi_1) = \cdots = \epsilon(\psi_n)$) is inductively defined as follows.

- If $n = 0$, then
  - $\mathrm{D}^{\sigma;} = \mathrm{C}^{\sigma}_{\top}$
- Otherwise ($n \geq 1$) let

$$\begin{array}{rcl} \{i_1, \ldots, i_h\} & = & \{i \mid 1 \leq i \leq n \text{ and } \psi_i = \bot^{\sigma}\} \\ \{j_1, \ldots, j_k\} & = & \{j \mid 1 \leq j \leq n \text{ and } \psi_j \neq \bot^{\sigma} \text{ and } \psi_j \not\cong_{\mathrm{cs}} \top^{\sigma}\} \end{array}$$

  (note that either $k = 0$ or $\psi_{j_1} \wedge \cdots \wedge \psi_{j_k} = \mathbf{norm}(\psi_1 \wedge \cdots \wedge \psi_n)$) in
  - $\mathrm{D}^{\iota;\psi_1,\ldots,\psi_n} = \lambda p_1^{\iota}. \cdots \lambda p_n^{\iota}.\mathsf{if}\ \mathsf{and}_{i \in \{i_1,\ldots,i_h\}} \mathsf{isdef}(p_i)\ \mathsf{then}\ \mathrm{C}^{\iota}_{\top}\ \mathsf{else}\ \sharp$,
    where $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$
  - $\mathrm{D}^{\sigma_1 \times \sigma_2;\psi_1,\ldots,\psi_n} = \lambda p_1^{\sigma_1 \times \sigma_2}. \cdots \lambda p_n^{\sigma_1 \times \sigma_2}.$

$$\begin{array}{ll} \mathsf{if} & \mathsf{and}_{i \in \{i_1,\ldots,i_h\}} \mathsf{isdef}(p_i) \\ \mathsf{then} & \langle \mathrm{D}^{\sigma_1;\chi_{j_1},\ldots,\chi_{j_k}}(\mathsf{proj}_1 p_{j_1}) \cdots (\mathsf{proj}_1 p_{j_k}), \\ & \quad \mathrm{D}^{\sigma_2;\chi_{j_1}',\ldots,\chi_{j_k}'}(\mathsf{proj}_2 p_{j_1}) \cdots (\mathsf{proj}_2 p_{j_k}) \rangle \\ \mathsf{else}\ \sharp & \end{array}$$

    where, for $j \in \{j_1, \ldots, j_k\}$, $\psi_j = \chi_j \times \chi_j'$
  - $\mathrm{D}^{\sigma_1 \to \sigma_2;\psi_1,\ldots,\psi_n} = \lambda p_1^{\sigma_1 \to \sigma_2}. \cdots \lambda p_n^{\sigma_1 \to \sigma_2}.\mathsf{if}\ \mathsf{and}_{i \in \{i_1,\ldots,i_h\}} \mathsf{isdef}(p_i)$

$$\mathsf{then}\ \lambda x^{\sigma_1}.\mathrm{D}^{\sigma_2;\chi_{j_1},\ldots,\chi_{j_k}}(p_{j_1} x) \cdots (p_{j_k} x)\ \mathsf{else}\ \sharp$$

    where, for $j \in \{j_1, \ldots, j_k\}$, $\psi_j = \nu_j \to \chi_j$

Figure 5.4: Terms $\mathrm{M}^{\xi}$ and $\mathrm{T}^{\xi}$

The following proposition can be easily proved by induction on cs-types.

**Proposition 5.16** *Let $\xi = \phi_1 \wedge \cdots \wedge \phi_n$ be a conjunction of $n \geq 1$ cs-types (note that, for $n = 1$, $\xi \in \mathbf{L}^{\mathrm{cs}}$), $\sigma = \epsilon(\xi)$ and $\phi$ be a cs-type.*

1. $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\xi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$.
2. $[\mathrm{T}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\xi \to \bot^{\mathsf{bool}}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ *and, for every* $[M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in \mathbf{I}(\epsilon(\xi))^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$, $[\mathrm{T}^{\xi} M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in \{[\mathsf{fix}\, x^{\epsilon(\xi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}, [\mathsf{true}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}\}$.
3. $[\mathrm{B}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\bot^{\mathsf{bool}} \to \phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ *and* $[\mathrm{B}^{\phi}\mathsf{true}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} = [\mathrm{C}^{\epsilon(\phi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$.
4. $[\mathrm{D}^{\sigma;}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\top^{\sigma}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$,
   $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\wedge_{i\in\{1,\ldots,n\}} \top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{i-1})} \to \phi_i \to \top^{\epsilon(\phi_{i+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \phi_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$, *and*
   $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}\mathrm{C}^{\epsilon(\phi_1)}_{\top}\cdots\mathrm{C}^{\epsilon(\phi_n)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} = [\mathrm{C}^{\sigma}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$. □

The proof of completeness rely on the following lemmas.

**Lemma 5.17** *Let $\xi = \phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{C}^{\mathrm{cs}}$.*

1. $\xi \cong_{\mathrm{cs}} \top^{\epsilon(\xi)}$ *if and only if* $[\mathrm{C}^{\epsilon(\xi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\xi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$.
2. $\xi \cong_{\mathrm{cs}} \top^{\epsilon(\xi)}$ *if and only if* $\mathrm{T}^{\xi}\mathrm{C}^{\epsilon(\xi)}_{\top} \Uparrow$.

**Proof.** First notice that

- (by Definition 5.2) $[\mathrm{C}^{\epsilon(\xi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\xi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ if and only if, for all $i \in \{1,\ldots,n\}$, $[\mathrm{C}^{\epsilon(\xi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\phi_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$, and
- (by Definition 5.14) $\mathrm{T}^{\xi}\mathrm{C}^{\epsilon(\xi)}_{\top} \Uparrow$ if and only if, for all $i \in \{1,\ldots,n\}$, $\mathrm{T}^{\phi_i}\mathrm{C}^{\epsilon(\xi)}_{\top} \Uparrow$.

So, by Proposition 5.10, it suffice to prove the property for $n = 1$.

1. *(only if)* Immediate by soundness (Theorem 5.9).
   *(if)* The case $\phi \cong_{\mathrm{cs}} \top^{\epsilon(\phi)}$ is immediate. The other cases are by structural induction on $\phi \in \mathbf{L}^{\mathrm{cs}}$.
   The case $\phi = \bot^{\epsilon(\phi)}$ is immediate.
   Let $\phi = \nu \to \psi \ncong_{\mathrm{cs}} \top^{\epsilon(\phi)}$, then $\psi \ncong_{\mathrm{cs}} \top^{\epsilon(\psi)}$ (by Proposition 5.10). By induction hypothesis $[\mathrm{C}^{\epsilon(\psi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \notin [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$. Now note that $[\mathrm{C}^{\epsilon(\nu\to\psi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} = [\lambda z^{\epsilon(\nu)}.\mathrm{C}^{\epsilon(\psi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$, so, for every $[M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}} \in [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, $[\mathrm{C}^{\epsilon(\nu\to\psi)}_{\top} M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} = [\mathrm{C}^{\epsilon(\psi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \notin [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$, which implies $[\mathrm{C}^{\epsilon(\phi)}_{\top}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \notin [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef}}}$.
   The case $\phi = \phi_1 \times \phi_2 \ncong_{\mathrm{cs}} \top^{\epsilon(\phi)}$ is similar.

2. *(only if)* Immediate since, $\phi \cong_{\mathrm{cs}} \top^{\epsilon(\phi)}$ implies $\mathrm{T}^{\phi} = \lambda z^{\epsilon(\phi)}.\mathsf{fix}\, x^{\mathsf{bool}}.x$.
   *(if)* The case $\phi \cong_{\mathrm{cs}} \top^{\epsilon(\phi)}$ is immediate. The other cases are by structural induction on $\phi \in \mathbf{L}^{\mathrm{cs}}$.
   The case $\phi = \bot^{\epsilon(\phi)}$ is immediate.
   Let $\phi = \nu \to \psi \ncong_{\mathrm{cs}} \top^{\epsilon(\phi)}$, then $\psi \ncong_{\mathrm{cs}} \top^{\epsilon(\psi)}$ (by Proposition 5.10). Since

$$[\mathrm{T}^{\nu\to\psi}\mathrm{C}_{\top}^{\epsilon(\nu\to\psi)}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathrm{T}^{\psi}\mathrm{C}_{\top}^{\epsilon(\psi)}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},$$

   the result follows by induction hypotheses.
   The case $\phi = \phi_1 \times \phi_2 \ncong_{\mathrm{cs}} \top^{\epsilon(\phi)}$ is similar. □

**Lemma 5.18** *Let $\xi = \phi_1 \wedge \cdots \wedge \phi_n$ and $\nu = \psi_1 \wedge \cdots \wedge \psi_m$ be cs-conjunctions such that $\epsilon(\xi) = \epsilon(\nu) = \sigma$.*

1. *If $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$ then $\xi \leq_{\mathrm{cs}} \nu$.*
2. *If $\mathrm{T}^{\nu}\mathrm{M}^{\xi} \Uparrow$ then $\xi \leq_{\mathrm{cs}} \nu$.*
3. *If $n = m = 1$ and $[\mathrm{B}^{\phi_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\bot^{\mathsf{bool}} \to \psi_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, then $\phi_1 \leq_{\mathrm{cs}} \psi_1$.*
4. *If $\forall i \in \{1,\ldots,m\}.\exists j_i \in \{1,\ldots,n\}$. $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in$*

$$[\![\top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{j_i-1})} \to \phi_{j_i} \to \top^{\epsilon(\phi_{j_i+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \psi_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},$$

   *then $\xi \leq_{\mathrm{cs}} \nu$.*

**Proof.** First notice that $\xi \leq_{\mathrm{cs}} \nu$ if and only if $\forall i \in \{1,\ldots,m\}.\exists j_i \in \{1,\ldots,n\}.\phi_{j_i} \leq_{\mathrm{cs}} \psi_i$.
We first consider the case in which either $\nu \cong_{\mathrm{cs}} \top^{\epsilon(\nu)}$ or $\xi \cong_{\mathrm{cs}} \top^{\epsilon(\xi)}$.
Let $\underline{\nu \cong_{\mathrm{cs}} \top^{\epsilon(\nu)}}$. All the 4 points are immediate.
Let $\underline{\xi \cong_{\mathrm{cs}} \top^{\epsilon(\xi)}}$. The case $\nu \cong_{\mathrm{cs}} \top^{\epsilon(\nu)}$ has been already considered, so let $\nu \ncong_{\mathrm{cs}} \top^{\epsilon(\nu)}$. Points 1. and 2. are obvious since (by Lemma 5.17) $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathrm{C}_{\top}^{\epsilon(\xi)}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$ and $\mathrm{T}^{\nu}\mathrm{C}_{\top}^{\epsilon(\xi)} \Downarrow \mathsf{true}$. For point 3. observe that $[\mathrm{B}^{\phi_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\lambda b^{\mathsf{bool}}.\mathrm{C}_{\top}^{\epsilon(\phi_1)}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\psi_1']\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, while, for point 4.,

$$\begin{aligned}[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} &= [\lambda p_1^{\epsilon(\phi_1)}.\cdots\lambda p_n^{\epsilon(\phi_n)}.\mathrm{C}_{\top}^{\epsilon(\xi)}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \\ &\notin [\![\wedge_{i\in\{1,\ldots,n\}}\bot^{\epsilon(\phi_1)} \to \cdots \to \bot^{\epsilon(\phi_n)} \to \psi_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}.\end{aligned}$$

If $\nu \ncong_{\mathrm{cs}} \top^{\epsilon(\nu)}$ and $\xi \ncong_{\mathrm{cs}} \top^{\epsilon(\xi)}$ the proof is by structural induction on $\mathbf{norm}(\nu)$ and $\mathbf{norm}(\xi)$.
Let $\underline{\mathbf{norm}(\nu) = \bot^{\epsilon(\nu)}}$. The case $\mathbf{norm}(\xi) \cong_{\mathrm{cs}} \top^{\epsilon(\xi)}$ has been already considered, and the case $\mathbf{norm}(\xi) = \bot^{\epsilon(\xi)}$ is trivial.
If $\mathbf{norm}(\xi) = \xi_1 \to \phi_1' \wedge \cdots \wedge \xi_p \to \phi_p'$ then $\xi \not\leq_{\mathrm{cs}} \nu$. The 4 points are obvious since, for $p = 1$,

1. $[\mathrm{M}^{\xi_1\to\phi_1'}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\bot^{\epsilon(\nu)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,

2. $[\mathrm{T}^{\nu}(\mathrm{M}^{\xi_1\to\phi'_1})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{isdef}(\mathrm{M}^{\xi_1\to\phi'_1})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{true}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,
3. $[\mathrm{B}^{\phi_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\bot^{\mathsf{bool}} \to \psi_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, and
4. $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\bot^{\epsilon(\phi_1)} \to \cdots \to \bot^{\epsilon(\phi_n)} \to \bot^{\epsilon(\nu)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,

while, for $p \geq 2$,

1. $[\mathrm{M}^{\xi_1\to\phi'_1\wedge\cdots\wedge\xi_p\to\phi'_p}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\bot^{\epsilon(\nu)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,
2. $[\mathrm{T}^{\nu}(\mathrm{M}^{\xi_1\to\phi'_1\wedge\cdots\wedge\xi_p\to\phi'_p})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{isdef}(\mathrm{M}^{\xi_1\to\phi'_1\wedge\cdots\wedge\xi_p\to\phi'_p})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{true}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,
3. is trivial, and
4. is as for the case $p = 1$.

The case $\mathbf{norm}(\xi) = \phi_1 \times \phi'_1 \wedge \cdots \wedge \phi_p \times \phi'_p$ is similar.

Let $\underline{\mathbf{norm}(\xi) = \bot^{\epsilon(\xi)}}$. The case $\mathbf{norm}(\nu) = \top^{\epsilon(\nu)}$ and the case $\mathbf{norm}(\nu) = \bot^{\epsilon(\nu)}$ have been already considered.
If $\mathbf{norm}(\nu) = \nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q$ then $\xi \leq_{\mathrm{cs}} \nu$. The 4 points are obvious since, for $q = 1$,

1. $[\mathrm{M}^{\bot^{\epsilon(\xi)}}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{fix}\, x^{\epsilon(\xi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,
2. $[\mathrm{T}^{\nu_1\to\psi'_1}(\mathrm{M}^{\bot^{\epsilon(\xi)}})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathrm{T}^{\psi'_1}(\mathsf{fix}\, y^{\epsilon(\psi'_1)}.y)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{true}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$ (by Lemma 5.17.2, since $\psi'_1 \not\cong_{\mathrm{cs}} \top^{\epsilon(\nu)}$),
3. $[\mathrm{B}^{\bot^{\epsilon(\xi)}}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\bot^{\mathsf{bool}} \to \bot^{\epsilon(\xi)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, and
4. $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{j-1})} \to \phi_j \to \top^{\epsilon(\phi_{j+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \bot^{\epsilon(\nu)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \subseteq [\![\top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{j-1})} \to \phi_j \to \top^{\epsilon(\phi_{j+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \nu_1 \to \psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, for some $j$ such that $\phi_j = \bot^{\epsilon(\phi_j)}$,

while, for $q \geq 2$,

1. $[\mathrm{M}^{\bot^{\epsilon(\xi)}}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{fix}\, x^{\epsilon(\xi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,
2. $[\mathrm{T}^{\nu}(\mathrm{M}^{\bot^{\epsilon(\xi)}})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{por}(\mathrm{T}^{\psi_1}(\mathrm{M}^{\bot^{\epsilon(\xi)}}), \ldots, \mathrm{T}^{\psi_n}(\mathrm{M}^{\bot^{\epsilon(\xi)}}))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,
3. is trivial, and
4. $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{j-1})} \to \phi_j \to \top^{\epsilon(\phi_{j+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \bot^{\epsilon(\nu)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \subseteq [\![\wedge_{i\in\{1,\ldots,q\}} \top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{j-1})} \to \phi_j \to \top^{\epsilon(\phi_{j+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \nu_i \to \psi'_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, for some $j$ such that $\phi_j = \bot^{\epsilon(\phi_j)}$.

The case $\mathbf{norm}(\nu) = \psi_1 \times \psi'_1 \wedge \cdots \wedge \psi_q \times \psi'_q$ is analogous.

Let $\mathbf{norm}(\nu) = \nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q$. All the cases but $\mathbf{norm}(\xi) = \xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p$ have been already considered. In this case $\xi \leq_{\mathrm{cs}} \nu$ if and only if $\forall i \in \{1,\ldots,q\}.\exists j \in \{1,\ldots,p\}$. both $\nu'_j \leq_{\mathrm{cs}} \xi'_i$ and $\phi'_j \leq_{\mathrm{cs}} \psi'_i$.
Then the 4 points can be proved as follows.
Case $p = 1$ and $q = 1$.

1. $[\mathrm{M}^{\xi_1 \to \phi'_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\nu_1 \to \psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ implies $[\mathrm{M}^{\xi_1 \to \phi'_1}\mathrm{M}^{\nu_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$. But

$$[\mathrm{M}^{\xi_1 \to \phi'_1}\mathrm{M}^{\nu_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} = [\mathrm{B}^{\phi'_1}(\mathrm{T}^{\xi_1}\mathrm{M}^{\nu_1})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}},$$

and (by Proposition 5.16 and Lemma 5.17.2, since $\psi'_1 \not\cong_{\mathrm{cs}} \top^{\epsilon(\psi'_1)}$) we have that both $\mathrm{T}^{\xi_1}\mathrm{M}^{\nu_1} \Uparrow$ and $[\mathrm{B}^{\phi'_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\bot^{\mathsf{bool}} \to \psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$, so (by induction hypotheses) we get $\nu_1 \leq_{\mathrm{cs}} \xi_1$ and $\phi'_1 \leq_{\mathrm{cs}} \psi'_1$, which imply $\xi_1 \to \phi'_1 \leq_{\mathrm{cs}} \nu_1 \to \psi'_1$.

2. Assume $\mathrm{T}^{\nu_1 \to \psi'_1}\mathrm{M}^{\xi_1 \to \phi'_1} \Uparrow$. Since

$$\begin{aligned}[\mathrm{T}^{\nu_1 \to \psi'_1}\mathrm{M}^{\xi_1 \to \phi'_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} &= [\mathrm{T}^{\psi'_1}(\mathrm{M}^{\xi_1 \to \phi'_1}\mathrm{M}^{\nu_1})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \\ &= [\mathrm{T}^{\psi'_1}(\mathrm{B}^{\phi'_1}(\mathrm{T}^{\xi_1}\mathrm{M}^{\nu_1}))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}},\end{aligned}$$

we get (by Proposition 5.16 and Lemma 5.17, since $\psi'_1 \not\cong_{\mathrm{cs}} \top^{\epsilon(\psi'_1)}$, using induction hypotheses) $\nu_1 \leq_{\mathrm{cs}} \xi_1$ and $\phi'_1 \leq_{\mathrm{cs}} \psi'_1$ which imply $\xi_1 \to \phi'_1 \leq_{\mathrm{cs}} \nu_1 \to \psi'_1$.

3. Assume $[\mathrm{B}^{\xi_1 \to \phi'_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\bot^{\mathsf{bool}} \to \nu_1 \to \psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$. Since

$$\begin{aligned}[\mathrm{B}^{\xi_1 \to \phi'_1}(\mathsf{fix}\, x^{\mathsf{bool}}.x)\mathrm{M}^{\nu_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} &= [\mathrm{B}^{\phi'_1}(\mathsf{por}((\mathsf{fix}\, x^{\mathsf{bool}}.x), (\mathrm{T}^{\xi_1}\mathrm{M}^{\nu_1})))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \\ &= [\mathrm{B}^{\phi'_1}(\mathrm{T}^{\xi_1}\mathrm{M}^{\nu_1})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}},\end{aligned}$$

we get (by Proposition 5.16 and Lemma 5.17, since $\psi'_1 \not\cong_{\mathrm{cs}} \top^{\epsilon(\psi'_1)}$, using induction hypotheses) $\nu_1 \leq_{\mathrm{cs}} \xi_1$ and $\phi'_1 \leq_{\mathrm{cs}} \psi'_1$ which imply $\xi_1 \to \phi'_1 \leq_{\mathrm{cs}} \nu_1 \to \psi'_1$.

4. Assume that there is $j_1 \in \{1,\ldots,n\}$ such that $[\mathrm{D}^{\epsilon(\xi);\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in$

$$[\![\top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{j_1-1})} \to \phi_{j_1} \to \top^{\epsilon(\phi_{j_1+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \nu_1 \to \psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}.$$

Note that by Proposition 5.16.4 and Lemma 5.17.1, since $\nu_1 \to \psi'_1 \not\cong_{\mathrm{cs}} \top^{\epsilon(\nu)}$, follows that $\phi_{j_1} = \xi_1 \to \phi'_1$. Since

$$\begin{aligned}[\mathrm{D}^{\epsilon(\xi);\phi_1,\ldots,\phi_n}\mathrm{M}^{\phi_1}\cdots\mathrm{M}^{\phi_n}\mathrm{M}^{\nu_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} &= [\mathrm{D}^{\epsilon(\phi'_1);\phi'_1}(\mathrm{M}^{\xi_1 \to \phi'_1}\mathrm{M}^{\nu_1})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \\ &= [\mathrm{D}^{\epsilon(\phi'_1);\phi'_1}(\mathrm{B}^{\phi'_1}(\mathrm{T}^{\xi_1}\mathrm{M}^{\nu_1}))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}},\end{aligned}$$

we get (by Proposition 5.16 and Lemma 5.17, since $\psi'_1 \not\cong_{\mathrm{cs}} \top^{\epsilon(\psi'_1)}$, using induction hypotheses) $\nu_1 \leq_{\mathrm{cs}} \xi_1$ and $\phi'_1 \leq_{\mathrm{cs}} \psi'_1$, which imply $\xi_1 \to \phi'_1 \leq_{\mathrm{cs}} \nu_1 \to \psi'_1$.

Case $p, q \geq 2$.

1. $[\mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$ implies, for all $i \in \{1, \ldots, q\}$, $[\mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p} \mathrm{M}^{\nu_i}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\psi'_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$. But

$$\begin{aligned}
&[\mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p} \mathrm{M}^{\nu_i}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = \\
&[\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathrm{M}^{\xi_1 \to \phi'_1} \mathrm{M}^{\nu_i}) \cdots (\mathrm{M}^{\xi_p \to \phi'_p} \mathrm{M}^{\nu_i})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = \\
&[\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathrm{B}^{\phi'_1}(\mathrm{T}^{\xi_1} \mathrm{M}^{\nu_i})) \cdots (\mathrm{B}^{\phi'_p}(\mathrm{T}^{\xi_p} \mathrm{M}^{\nu_i}))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},
\end{aligned}$$

so (by Proposition 5.16 and Lemma 5.17, since $\psi'_i \not\cong_{\mathrm{cs}} \top^{\epsilon(\psi'_i)}$, using induction hypotheses) we get $\forall i \in \{1, \ldots, q\}.\exists j \in \{1, \ldots, p\}.\xi_j \to \phi'_j \leq_{\mathrm{cs}} \nu_i \to \psi'_i$, which imply $\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p \leq_{\mathrm{cs}} \nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q$.

2. Assume $\mathrm{T}^{\nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q} \mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p} \Uparrow$. Since

$$\begin{aligned}
&[\mathrm{T}^{\nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q} \mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = \\
&[\mathsf{por}(\mathrm{T}^{\nu_1 \to \psi'_1} \mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p}, \ldots, \mathrm{T}^{\nu_q \to \psi'_q} \mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},
\end{aligned}$$

and, for all $i \in \{1, \ldots, q\}$,

$$\begin{aligned}
&[\mathrm{T}^{\nu_i \to \psi'_i} \mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = \\
&[\mathrm{T}^{\psi'_i}(\mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p} \mathrm{M}^{\nu_i})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = \\
&[\mathrm{T}^{\psi'_1}(\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathrm{M}^{\xi_1 \to \phi'_1} \mathrm{M}^{\nu_i}) \cdots (\mathrm{M}^{\xi_p \to \phi'_p} \mathrm{M}^{\nu_i}))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = \\
&[\mathrm{T}^{\psi'_1}(\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathrm{B}^{\phi'_1}(\mathrm{T}^{\xi_1} \mathrm{M}^{\nu_i})) \cdots (\mathrm{B}^{\phi'_p}(\mathrm{T}^{\xi_p} \mathrm{M}^{\nu_i})))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},
\end{aligned}$$

we get (by Proposition 5.16 and Lemma 5.17, since $\psi'_i \not\cong_{\mathrm{cs}} \top^{\epsilon(\psi'_i)}$, using induction hypotheses) $\forall i \in \{1, \ldots, q\}.\exists j \in \{1, \ldots, p\}.\xi_j \to \phi'_j \leq_{\mathrm{cs}} \nu_i \to \psi'_i$, which imply $\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p \leq_{\mathrm{cs}} \nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q$.

3. Trivial.

4. Assume that $\forall i \in \{1, \ldots, q\}.\exists j_i \in \{1, \ldots, n\}$. $[\mathrm{D}^{\epsilon(\xi);\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in$

$$[\![\top^{\epsilon(\phi_1)} \to \ldots \top^{\epsilon(\phi_{j_i-1})} \to \phi_{j_i} \to \top^{\epsilon(\phi_{j_i+1})} \to \ldots \top^{\epsilon(\phi_n)} \to \nu_i \to \psi'_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}.$$

Note that by Proposition 5.16.4 and Lemma 5.17.1, since, for every $i \in \{1, \ldots, q\}$, $\nu_i \to \psi'_i \not\cong_{\mathrm{cs}} \top^{\epsilon(\nu)}$, follows that $\phi_{j_1}, \ldots, \phi_{j_q} \in \{\xi_1 \to \phi'_1, \ldots, \xi_p \to \phi'_p\}$. Since, for all $i \in \{1, \ldots, q\}$,

$$\begin{aligned}
&[\mathrm{D}^{\epsilon(\xi);\phi_1,\ldots,\phi_n} \mathrm{M}^{\phi_1} \cdots \mathrm{M}^{\phi_n} \mathrm{M}^{\nu'_i}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = \\
&[\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathrm{M}^{\xi_1 \to \phi'_1} \mathrm{M}^{\nu_i}) \cdots (\mathrm{M}^{\xi_p \to \phi'_p} \mathrm{M}^{\nu_i})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = \\
&[\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathrm{B}^{\phi'_1}(\mathrm{T}^{\xi_1} \mathrm{M}^{\nu_i})) \cdots (\mathrm{B}^{\phi'_p}(\mathrm{T}^{\xi_p} \mathrm{M}^{\nu_i}))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},
\end{aligned}$$

we get (by Proposition 5.16 and Lemma 5.17, since $\psi'_i \not\cong_{\mathrm{cs}} \top^{\epsilon(\psi'_i)}$, using induction hypotheses) $\forall i \in \{1, \ldots, q\}.\exists j \in \{1, \ldots, p\}.\xi_j \to \phi'_j \leq_{\mathrm{cs}} \nu_i \to \psi'_i$, which imply $\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p \leq_{\mathrm{cs}} \nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q$.

The case $p = 1$ and $q \geq 2$, and the case $p \geq 2$ and $q = 1$ are similar.

The proof of the only remaining case, $\mathbf{norm}(\nu) = \psi_1 \times \psi'_1 \wedge \cdots \wedge \psi_q \times \psi'_q$ and $\mathbf{norm}(\xi) = \phi_1 \times \phi'_1 \wedge \cdots \wedge \phi_p \times \phi'_p$, is similar to the proof of the previous one. □

From the previous lemma we derive the completeness of the entailment relation.

**Proof of Theorem 5.12.** By contraposition. Assume that $\xi \not\leq_{\mathrm{cs}} \nu$. Then $\nu \not\cong_{\mathrm{cs}} \top^{\epsilon(\psi)}$. By Lemma 5.18.1 $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \notin [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$. Since from Proposition 5.16.1 $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\xi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$, this implies that $[\![\xi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \not\subseteq [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$. □

**Remark 5.19** *As showed in Example 5.11 the entailment relation $\leq_{\mathrm{cs}}$ is not complete w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$. At the moment we do not known whether a sound (and decidable) extension of the $\leq_{\mathrm{cs}}$ relation which is complete w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$ exists.*

### 5.2.2 An assignment system for conjunctive strictness types

If $x^{\rho}$ is a term variable of type $\rho$ an assumption for $x^{\rho}$ is an expression of the shape $x^{\rho} : \xi$, or $x : \xi$ for short, where $\xi \in \mathbf{C}^{\mathrm{cs}}$ and $\epsilon(\xi) = \rho$. A basis is a set $\Sigma$ of cs-types assumptions for term variables.

For each constant $\mathsf{c}$ a set $\mathbf{L}^{\mathrm{cs}}(\mathsf{c})$ of minimal cs-types is specified. The idea is that every cs-type which is entailed ($\leq_{\mathrm{cs}}$) by an element of $\mathbf{L}^{\mathrm{cs}}(\mathsf{c})$ can be assigned to $\mathsf{c}$. For instance, for all integers $\mathsf{n}$, $\mathbf{L}^{\mathrm{cs}}(\mathsf{n}) = \{\top^{\mathsf{nat}}\}$ and $\mathbf{L}^{\mathrm{cs}}(+) = \{\bot^{\mathsf{nat}} \times \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}, \top^{\mathsf{nat}} \times \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}\}$.

**Definition 5.20 (Strictness type assignment system)** *We write $\Sigma \vdash_{\mathrm{cs}} M : \phi$ to mean that $\Sigma \vdash M : \phi$ can be derived by the rules in Fig. 5.5.*

The proof of the following fact is immediate.

**Fact 5.21** *1. $\Sigma \vdash_{\mathrm{cs}} M : \phi$ implies $\vdash_{\mathbf{T}} M : \epsilon(\phi)$ and $\epsilon(\Sigma) \supseteq \mathrm{FV}(M)$.*

*2. $\vdash_{\mathbf{T}} M : \rho$ implies, for all basis $\Sigma$ such that $\Sigma = \{x^{\epsilon(\xi)} : \xi \mid x^{\epsilon(\xi)} \in \mathrm{FV}(M)$ for some $\xi \in \mathbf{C}^{\mathrm{cs}}\}$, $\Sigma \vdash_{\mathrm{cs}} M : \top^{\rho}$.* □

The system $\vdash_{\mathrm{cs}}$ is more powerful than $\vdash_{\mathrm{s}}$, as the following example shows.

**Example 5.22** *Let $\vdash_{\mathbf{T}} G : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$, where*

$$G = (\lambda f^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\mathsf{if}\ x = 0\ \mathsf{then}\ f1y\ \mathsf{else}\ fy1)(\lambda u^{\mathsf{nat}}.\lambda v^{\mathsf{nat}}.u + v).$$

*Using $\vdash_{\mathrm{cs}}$ we are able to infer that $G$ is strict in its second arguments, i.e. $\emptyset \vdash_{\mathrm{cs}} G : \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$.*

*As explained ad the beginning of the chapter, in $\vdash_{\mathrm{s}}$ it is not possible to assign to $G$ the property $\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$.*

The rule ($\mathrm{Fix}_{\bot}$) of the system $\vdash_{\mathrm{cs}}$ is more powerful of the corresponding rule of the system $\vdash_{\mathrm{s}}$ (in Figure 4.3), as the following example shows.

$$(\leq)\ \frac{\Sigma \vdash M : \phi \quad \phi \leq \psi}{\Sigma \vdash M : \psi}$$

$$(\text{Var})\ \frac{}{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash x : \phi_i}\ 1 \leq i \leq n \qquad (\text{Con})\ \frac{\phi \in \mathbf{L}^{\text{cs}}(\mathsf{c})}{\Sigma \vdash \mathsf{c} : \phi}$$

$$(\to \text{I})\ \frac{\Sigma, x : \xi \vdash M : \psi}{\Sigma \vdash \lambda x.M : \xi \to \psi}$$

$$(\to \text{E})\ \frac{\Sigma \vdash M : \phi_1 \wedge \cdots \wedge \phi_n \to \psi \quad \Sigma \vdash N : \phi_1 \ \cdots\ \Sigma \vdash N : \phi_n}{\Sigma \vdash MN : \psi}$$

$$(\times \text{I})\ \frac{\Sigma \vdash M_1 : \phi_1 \quad \Sigma \vdash M_2 : \phi_2}{\Sigma \vdash \langle M_1, M_2 \rangle : \phi_1 \times \phi_2} \qquad (\times \text{E}_i)\ \frac{\Sigma \vdash M : \phi_1 \times \phi_2}{\Sigma \vdash \mathsf{proj}_i M : \phi_i}\ i \in \{1, 2\}$$

$$(\text{Fix}_\bot)\ \frac{\forall j \in \{1, \ldots, n\}.\ \Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash M : \phi_j}{\Sigma \vdash \mathsf{fix}\, x.M : \phi_i}\ 1 \leq i \leq n$$

$$(\text{If}_\bot)\ \frac{\Sigma \vdash N : \bot^{\mathsf{bool}} \quad \vdash_{\mathbf{T}} M_1 : \rho \quad \vdash_{\mathbf{T}} M_2 : \rho}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \bot^\rho}$$

$$(\text{If}_\top)\ \frac{\Sigma \vdash N : \top^{\mathsf{bool}} \quad \Sigma \vdash M_1 : \phi \quad \Sigma \vdash M_2 : \phi}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \phi}$$

Figure 5.5: Rules for cs-types assignment (system $\vdash_{\text{cs}}$)

**Example 5.23** *Take the term* $F = \mathsf{fix}\, f^{\rho}.M$*, where* $\rho = \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$*, and*

$$\begin{array}{rcl} M & = & \lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\lambda z^{\mathsf{nat}}.\mathsf{if}\ z = 0\ \mathsf{then}\ x + y \\ & & \mathsf{else\ if}\ z \leq 5\ \mathsf{then}\ fyx(z-1)\ \mathsf{else}\ fxy(z-1). \end{array}$$

*Using* $\vdash_{\mathrm{cs}}$ *it is possible to prove that* $F$ *is strict in each of its 3 arguments, i.e.* $\emptyset \vdash_{\mathrm{cs}} F : \phi_1$*,* $\emptyset \vdash_{\mathrm{cs}} F : \phi_2$*, and* $\emptyset \vdash_{\mathrm{cs}} F : \phi_3$*, where*

- $\phi_1 = \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$*,*
- $\phi_2 = \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$*, and*
- $\phi_3 = \top^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$*.*

*Note that to assign* $\phi_3$ *to* $F$ *one premise is sufficient, in fact* $\{f : \phi_3\} \vdash_{\mathrm{cs}} M : \phi_3$*, and so it is possible to apply the rule* $(\mathrm{Fix}_{\bot})$ *as follows:*

$$(\mathrm{Fix}_{\bot})\ \frac{\{f : \phi_3\} \vdash_{\mathrm{cs}} M : \phi_3}{\emptyset \vdash_{\mathrm{cs}} \mathsf{fix}\, f^{\rho}.M : \phi_3}\quad .$$

*In particular this can also be done using* $\vdash_{\mathrm{s}}$*, i.e.* $\emptyset \vdash_{\mathrm{s}} F : \phi_3$*.*

*Instead to assign* $\phi_1$ *or* $\phi_2$ *to* $F$ *two premises and the explicit use of the* $\wedge$ *operator are required. In fact, for* $i \in \{1, 2\}$*,* $\{f : \phi_1 \wedge \phi_2\} \vdash_{\mathrm{cs}} M : \phi_i$*, and so two premises are sufficient, since:*

$$(\mathrm{Fix}_{\bot})\ \frac{\{f : \phi_1 \wedge \phi_2\} \vdash_{\mathrm{cs}} M : \phi_1 \quad \{f : \phi_1 \wedge \phi_2\} \vdash_{\mathrm{cs}} M : \phi_2}{\emptyset \vdash_{\mathrm{cs}} \mathsf{fix}\, f^{\rho}.M : \phi_i}\ ,$$

*but (as it is easy to check) with* $\vdash_{\mathrm{s}}$ *it is not possible to assign neither* $\phi_1$ *nor* $\phi_2$ *to* $F$*.*

Soundness of the cs-types assignment system (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$) is stated by the following theorem.

**Theorem 5.24 (Soundness of $\vdash_{\mathrm{cs}}$)** *Let* $\Sigma \vdash_{\mathrm{cs}} M : \phi$*. Then* $[\![M]\!]_e \in [\![\phi]\!]$ *for all* $e$ *such that for all* $x : \xi \in \Sigma$*,* $e(x) \in [\![\xi]\!]$*.*

**Proof.** By induction on the derivation of $\Sigma \vdash_{\mathrm{cs}} M : \phi$. □

### 5.2.3 Conjunctive strictness analysis and ground contextual equivalence

The entailment relation presented in Section 5.2.1 can be modified to fit better the ground contextual equivalence, $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$, instead of the lazy contextual equivalence, $\simeq^{\mathrm{lazy}}_{\mathrm{obs}}$. In this perspective a cs-type $\phi$ is interpreted as a subset of the interpretation of the type $\epsilon(\phi)$ in the model $\mathcal{M}^{\mathrm{gnd}}$ of Section 2.4.1. For every cs-type $\phi \in \mathbf{L}^{\mathrm{cs}}$, let $[\![\phi]\!]^{\mathcal{M}^{\mathrm{gnd}}}$ be the interpretation of $\phi$ in the model $\mathcal{M}^{\mathrm{gnd}}$ defined according to clauses of Definition 5.2.

The associated entailment relation between cs-types will identify all the cs-types corresponding to functions that diverge for all inputs. Such cs-types, called b-cs-types are defined as follows.

**Definition 5.25 (b-cs-types)** *The set* $\mathbf{L}_{\mathrm{b}}^{\mathrm{cs}}$ *of* b*-cs-types is defined by:*

- $\bot^{\rho} \in \mathbf{L}_{\mathrm{b}}^{\mathrm{cs}}$, *for* $\rho \in \mathbf{T}$
- $\phi_1 \wedge \cdots \wedge \phi_n \to \psi \in \mathbf{L}_{\mathrm{b}}^{\mathrm{cs}}$, *for* $n \geq 1$, $\phi_1, \ldots, \phi_n \in \mathbf{L}_{\top}^{\mathrm{cs}}$ *and* $\psi \in \mathbf{L}_{\mathrm{b}}^{\mathrm{cs}}$
- $\phi_1 \times \phi_2 \in \mathbf{L}_{\mathrm{b}}^{\mathrm{cs}}$, *for* $\phi_1, \phi_2 \in \mathbf{L}_{\mathrm{b}}^{\mathrm{cs}}$.

We can now introduce the entailment relation $\leq_{\mathrm{cs}}^{\mathrm{g}}$; $\phi \leq_{\mathrm{cs}}^{\mathrm{g}} \psi$ reflects the set-theoretic inclusion $[\![\phi]\!]^{\mathcal{M}^{\mathrm{gnd}}} \subseteq [\![\psi]\!]^{\mathcal{M}^{\mathrm{gnd}}}$.

**Definition 5.26 (Entailment relation $\leq_{\mathrm{cs}}^{\mathrm{g}}$)** *Let* $\phi, \psi \in \mathbf{L}^{\mathrm{cs}}$. *We write* $\phi \leq_{\mathrm{cs}}^{\mathrm{g}} \psi$ *to mean that* $\phi \leq \psi$ *is derivable by using the rules in Fig. 5.2 in which the axiom* $(\bot)$ *is replaced by the following*

$$(\bot_{\mathrm{g}})\ \frac{\phi \in \mathbf{L}_{\mathrm{b}}^{\mathrm{cs}} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}\ .$$

*By* $\cong_{\mathrm{cs}}^{\mathrm{g}}$ *we denote the equivalence relation induced by* $\leq_{\mathrm{cs}}^{\mathrm{g}}$.

Soundness of the $\leq_{\mathrm{cs}}^{\mathrm{g}}$ entailment w.r.t. both $\mathcal{M}^{\mathrm{gnd}}$ and $\mathcal{M}_{\mathsf{por}}^{\mathrm{gnd}}$ is stated by the following theorem.

**Theorem 5.27 (Soundness of $\leq_{\mathrm{cs}}^{\mathrm{g}}$)** *Let* $\phi, \psi \in \mathbf{L}^{\mathrm{cs}}$. $\phi \leq_{\mathrm{cs}}^{\mathrm{g}} \psi$ *implies* $[\![\phi]\!] \subseteq [\![\psi]\!]$.

**Proof.** By induction on the derivation of the entailment judgement. □

The entailment $\leq_{\mathrm{cs}}^{\mathrm{g}}$ is not complete w.r.t. the interpretation in $\mathcal{M}^{\mathrm{gnd}}$, since the argument of Example 5.11 apply also for the $\leq_{\mathrm{cs}}^{\mathrm{g}}$ entailment. However, the entailment $\leq_{\mathrm{cs}}^{\mathrm{g}}$ is complete w.r.t. the term model $\mathcal{M}_{\mathsf{por}}^{\mathrm{gnd}}$ of Section 2.4.2.

**Theorem 5.28 (Completeness of $\leq_{\mathrm{cs}}^{\mathrm{g}}$ w.r.t. $\mathcal{M}_{\mathsf{por}}^{\mathrm{gnd}}$)** *Let* $\phi, \psi \in \mathbf{L}^{\mathrm{cs}}$. $[\![\phi]\!]^{\mathcal{M}_{\mathsf{por}}^{\mathrm{gnd}}} \subseteq [\![\psi]\!]^{\mathcal{M}_{\mathsf{por}}^{\mathrm{gnd}}}$ *imply* $\phi \leq_{\mathrm{cs}}^{\mathrm{g}} \psi$.

**Proof.** Similar to the proof of Theorem 5.12. □

**Definition 5.29 (Assignment system $\vdash_{\mathrm{cs}}^{\mathrm{g}}$)** *We write* $\Sigma \vdash_{\mathrm{cs}}^{\mathrm{g}} M : \phi$ *to mean that* $\Sigma \vdash M : \phi$ *can be derived by the rules in Fig. 5.5 using the entailment* $\leq_{\mathrm{cs}}^{\mathrm{g}}$ *instead of* $\leq_{\mathrm{cs}}$.

Soundness of the cs-types assigment system $\vdash_{\mathrm{cs}}^{\mathrm{g}}$ w.r.t. both $\mathcal{M}^{\mathrm{gnd}}$ and $\mathcal{M}_{\mathsf{por}}^{\mathrm{gnd}}$ is stated by the following theorem.

**Theorem 5.30 (Soundness of $\vdash_{\mathrm{cs}}^{\mathrm{g}}$)** *Let* $\Sigma \vdash_{\mathrm{cs}} M : \phi$. *Then* $[\![M]\!]_e \in [\![\phi]\!]$ *for all* $e$ *such that for all* $x : \psi \in \Sigma$, $e(x) \in [\![\psi]\!]$.

**Proof.** By induction on the derivation of $\Sigma \vdash_{\mathrm{cs}}^{\mathrm{g}} M : \phi$. □

**Remark 5.31** *The entailment relation* $\leq_{\mathrm{cs}}^{\mathrm{g}}$ *is not complete w.r.t. the model* $\mathcal{M}^{\mathrm{gnd}}$. *It is a very interesting research problem finding a sound (and decidable) extension of the* $\leq_{\mathrm{cs}}^{\mathrm{g}}$ *relation which is complete w.r.t. the model* $\mathcal{M}^{\mathrm{gnd}}$.

### 5.2.4 Comparison with the systems of Benton and Jensen

In spite of the syntactical differences in the language of properties and a minor difference in the inclusion relation, the cs-type assignment system $\vdash^{\mathrm{g}}_{\mathrm{cs}}$ of the previous section is as powerful as the *conjunctive strictness logic* introduced in [Jensen, 1992; Jensen, 1995], and (independently) in [Benton, 1992a]. In particular the cs-types in $\mathbf{L}^{\mathrm{cs}}$ correspond to a subset of the formulae of the conjunctive strictness logic: the set of *simple formulae*, described in [Jensen, 1992] pag. 38. This subset is obtained from the formulas of the conjunctive strictness logic, as presented in [Jensen, 1992] pag. 38 and [Benton, 1992a] pag. 37, by forbidding the use of the conjunction operator in the right part of the arrow operator and at the top level. In [Jensen, 1992] it is proved (Theorem 3.2.2) that every formula of the conjunctive strictness logic can be expressed as a conjunction of simple formulae. So, as pointed out in [Hankin and Le Métayer, 1994b] (see Remark 5.32 below), restricting to simple formulae makes the type manipulation easier without affecting the expressive power of the logic.

In [Jensen, 1992] it is proved that the conjunctive strictness logic and the abstract interpretation in the style of [Burn *et al.*, 1986] are equivalent, but no attempt to prove soundness w.r.t. a standard (denotational or operational) semantics of the programming language is made (for this purpose [Jensen, 1992] relies on results in [Burn *et al.*, 1986]). Instead in [Benton, 1992a] the conjunctive strictness logic is proved sound w.r.t. the standard denotational semantics, which is adequate (but not fully abstract, see [Benton, 1992a] pag. 22) w.r.t. the ground observational equivalence $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$ (inducing the term model $\mathcal{M}^{\mathrm{gnd}}$), and fully abstract w.r.t. the ground observational equivalence of the language extended with a "parallel or" operator (inducing the term model $\mathcal{M}^{\mathrm{gnd}}_{\mathsf{por}}$). Both [Jensen, 1992] and [Benton, 1992a] present a completeness result for the entailment relation of the conjunctive strictness logic.

In this thesis we take a more direct approach, since all the results are proved directly w.r.t. the operational semantics of the programming language. Moreover we consider both lazy and ground observational equivalences, while *op. cit.* focus on the ground case.

**Remark 5.32 (A checking algorithm for strictness)** *In [Hankin and Le Métayer, 1994b] (see also [Hankin and Le Métayer, 1994a; Hankin and Le Métayer, 1995]) it is described a* checking *algorithm (i.e. an algorithm that given a program $M$ checks whether a particular strictness property can be assigned to $M$) for a version of the conjunctive strictness logic that uses only top level conjunctions of simple formulae*[2]*. The algorithm (which uses a number of optimizations such as lazy types and the analogous of frontiers for conjunctive strictness types) has the same power as usual imlementations of abstract interpretation but does not exibit the same inefficiency problems.*

[2]Which is essentially the system $\vdash^{\mathrm{g}}_{\mathrm{cs}}$ in Section 5.2.3 of this thesis.

$$(\not\downarrow)\ \frac{\phi \in \mathbf{L}^{\mathrm{ct}} - \mathbf{L}^{\mathrm{ct}}_{\top}}{\phi \leq_{\mathrm{ct}} \not\downarrow^{\epsilon(\phi)}} \quad (\top)\ \frac{\psi \in \mathbf{L}^{\mathrm{ct}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq_{\mathrm{ct}} \psi}$$

$$(\mathrm{Ref})\ \phi \leq_{\mathrm{ct}} \phi \quad (\rightarrow)\ \frac{\xi_2 \leq_{\mathrm{ct}} \xi_1 \quad \phi_1 \leq_{\mathrm{ct}} \phi_2}{\xi_1 \rightarrow \phi_1 \leq_{\mathrm{ct}} \xi_2 \rightarrow \phi_2} \quad (\times)\ \frac{\phi_1 \leq_{\mathrm{ct}} \psi_1 \quad \psi_1 \leq_{\mathrm{ct}} \psi_2}{\phi_1 \times \phi_2 \leq_{\mathrm{ct}} \psi_2 \times \psi_2}$$

$$(\wedge)\ \frac{\forall i \in \{1,\ldots,m\}.\exists j \in \{1,\ldots,n\}.\phi_j \leq_{\mathrm{ct}} \psi_i}{\phi_1 \wedge \cdots \wedge \phi_n \leq_{\mathrm{ct}} \psi_1 \wedge \cdots \wedge \psi_m}$$

Figure 5.6: Entailment rules for ct-types (system $\leq_{\mathrm{ct}}$)

## 5.3 Conjunctive totality analysis

In this section we extend the totality analysis of Section 4.3 by allowing the use of conjunction.

Let $\mathbf{L}^{\mathrm{ct}}$ and $\mathbf{C}^{\mathrm{ct}}$ be respectively the sets of cst-types and cst-conjunctions that do not contain the constructor $\bot$. We call the properties in $\mathbf{L}^{\mathrm{ct}}$ *conjunctive totality types* (*ct-types* for short) and the properties in $\mathbf{C}^{\mathrm{ct}}$ *conjunctions of ct-types* (*ct-conjunctions*). Note that $\mathbf{L}^{\mathrm{ct}} \subset \mathbf{C}^{\mathrm{ct}}$.

### 5.3.1 An entailment relation for conjunctive totality types

**Definition 5.33 (Entailment relation for totality types)** *Let $\xi_1, \xi_2 \in \mathbf{C}^{\mathrm{ct}}$. We write $\xi_1 \leq_{\mathrm{ct}} \xi_2$ to mean that $\xi_1 \leq_{\mathrm{ct}} \xi_2$ is derivable by the rules in Fig. 5.6. By $\cong_{\mathrm{ct}}$ we denote the equivalence relation induced by $\leq_{\mathrm{ct}}$.*

As for the entailment $\leq_{\mathrm{cs}}$, also $\leq_{\mathrm{ct}}$ is reflexive and transitive. Note that entailment $\leq_{\mathrm{ct}}$ differs from $\leq_{\mathrm{cs}}$ only for the axiom $(\not\downarrow)$ (used instead of $(\bot)$).

Soundness of the entailment $\leq_{\mathrm{ct}}$ (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}$) is stated by the following theorem.

**Theorem 5.34 (Soundness $\leq_{\mathrm{ct}}$)** *Let $\xi_1, \xi_2 \in \mathbf{C}^{\mathrm{ct}}$. $\xi_1 \leq_{\mathrm{ct}} \phi_2$ implies that $[\![\xi_1]\!] \subseteq [\![\xi_2]\!]$.*

**Proof.** By induction on the derivation of the entailment judgement. □

**Proposition 5.35** *Let $\xi = \phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{C}^{\mathrm{ct}}$. $[\![\xi]\!] = [\![\top^{\epsilon(\xi)}]\!]$ if and only if $\phi_1, \ldots, \phi_n \in \mathbf{L}^{\mathrm{ct}}_{\top}$ if and only if $\xi \cong_{\mathrm{ct}} \top^{\epsilon(\xi)}$.*

**Proof.**

- $\underline{[\![\xi]\!] = [\![\top^{\epsilon(\xi)}]\!] \text{ implies } \phi_1, \ldots, \phi_n \in \mathbf{L}^{\mathrm{ct}}_{\top}}$.
  We have that $[\![\xi]\!] = \mathbf{I}(\epsilon(\xi))$ implies $[\![\phi_1]\!] = \cdots = [\![\phi_n]\!] = \mathbf{I}(\epsilon(\xi))$, so it suffices to prove that, for $\phi \in \mathbf{L}^{\mathrm{ct}}$, $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$ implies $\phi \in \mathbf{L}^{\mathrm{ct}}_{\top}$.

By structural induction on $\phi \in \mathbf{L}^{\mathrm{ct}}$. The case $\phi \in \{\top^\rho, \not\perp^\rho\}$, for some $\rho \in \mathbf{T}$, is immediate.
For $\phi = \nu \to \psi$, we have that $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi))$ implies (by Definition 5.2) $[\![\psi]\!] = \mathbf{I}(\epsilon(\psi))$ implies (by induction hypothesis) $\psi \in \mathbf{L}^{\mathrm{ct}}_\top$ implies $\phi \in \mathbf{L}^{\mathrm{ct}}_\top$.
The case $\phi = \phi_1 \times \phi_2$ is similar.

- $\underline{\phi_1, \ldots, \phi_n \in \mathbf{L}^{\mathrm{ct}}_\top \text{ implies } \xi \cong_{\mathrm{ct}} \top^{\epsilon(\xi)}}$.
  Immediate by rules $(\top)$ and $(\wedge)$ of $\leq_{\mathrm{ct}}$.
- $\underline{\xi \cong_{\mathrm{ct}} \top^{\epsilon(\xi)} \text{ implies } [\![\xi]\!] = [\![\top^{\epsilon(\xi)}]\!]}$.
  Immediate by Theorem 5.34. □

As for $\leq_{\mathrm{cs}}$, the entailment $\leq_{\mathrm{ct}}$ is not complete w.r.t. $\mathcal{M}^{\mathrm{lazy}}$, owing to the sequential nature of the evaluation rules of the language PCFP.

**Example 5.36 (Incompleteness of $\leq_{\mathrm{ct}}$ w.r.t. $\mathcal{M}^{\mathrm{lazy}}$)** *Let $\phi_1 = \not\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$, $\phi_2 = \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$, and $\psi = \top^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$. We can show, with an argument similar to that of Example 5.11, that*

$$[\![\phi_1 \wedge \phi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}} = [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}}.$$

*However $\phi_1 \not\leq_{\mathrm{ct}} \psi$ and $\phi_2 \not\leq_{\mathrm{ct}} \psi$, therefore it is easy to see that*

$$\phi_1 \wedge \phi_2 \not\leq_{\mathrm{ct}} \psi.$$

Another reason for incompleteness is that in PCFP the functional terms with w.h.n.f. cannot be discriminated, i.e. the argument of Example 4.35 apply also to $\leq_{\mathrm{ct}}$.

**Lemma 5.37** *Let $\phi \in \mathbf{L}^{\mathrm{ct}}$. $\phi \in \mathbf{L}^{\mathrm{ct}} - \mathbf{L}^{\mathrm{ct}}_\top$ if and only if $[\![\phi]\!] \subseteq [\![\not\perp^{\epsilon(\phi)}]\!]$.*

**Proof.** Similar to the proof of Proposition 5.35.1. □

Completeness of $\leq_{\mathrm{ct}}$ w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}$ can be proved with a technique similar to that used in Section 5.3.1 for the entailment $\leq_{\mathrm{cs}}$.

**Theorem 5.38 (Completeness of $\leq_{\mathrm{ct}}$ w.r.t. $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}$)** *Let $\xi_1, \xi_2 \in \mathbf{C}^{\mathrm{ct}}$. Then, $[\![\xi_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \subseteq [\![\xi_2]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ implies that $\xi_1 \leq_{\mathrm{ct}} \xi_2$.* □

The completeness proof proceeds as for the $\leq_{\mathrm{cs}}$ entailment by defining, for each ct-type $\phi$ (and for each conjunction of $n \geq 2$ ct-types $\xi$), a term $\mathrm{M}^\phi$ (resp. $\mathrm{M}^\xi$) that characterizes the property $\phi$ ($\xi$).

**Notation 5.39** *1. We say that a conjunction of $n \geq 1$ ct-types $\phi_1 \wedge \cdots \wedge \phi_n$ is* normal *if either $n = 1$ or $n \geq 2$ and, for all $i$, $1 \leq i \leq n$, $\phi_i \not\cong_{\mathrm{ct}} \top^\rho$ and $\phi_i \not\cong_{\mathrm{ct}} \not\perp^\rho$. Note that every ct-conjunction $\xi$ is equivalent ($\cong_{\mathrm{ct}}$) to a normal ct-conjunction since it is possible to eliminate (first) the components equivalent ($\cong_{\mathrm{ct}}$) to $\top^\rho$ and (then) the ones equal to $\not\perp^\rho$. Define* $\mathbf{norm}(\xi)$ *be the ct-conjunction obtained from $\xi$ in this way.*

2. *For every type $\rho$ let $\mathbf{min}_{\mathrm{t}}(\rho)$ be the ct-type defined by the clauses at the end of Remark 5.7. It is easy to check that $\mathbf{min}_{\mathrm{t}}(\rho)$ is the minimum (w.r.t. $\leq_{\mathrm{ct}}$) ct-type in $\mathbf{L}^{\mathrm{ct}}(\rho)$, i.e. for every $\phi \in \mathbf{L}^{\mathrm{ct}}(\rho)$, $\mathbf{min}_{\mathrm{t}}(\rho) \leq_{\mathrm{ct}} \phi$.*

3. *In the following definition we will use an n-ary strict* and *operator, which is defined as in Notation 5.13.2.*

The characteristics terms $\mathrm{M}^{\phi}$ and $\mathrm{M}^{\xi}$ are such that, for every ct-type $\psi$ and ct-conjunction $\nu$,

- $[\mathrm{M}^{\phi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ if and only if $\phi \leq_{\mathrm{ct}} \psi$,
- $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ if and only if $\xi \leq_{\mathrm{ct}} \nu$.

We also need to define

- a test term $\mathrm{T}^{\phi}$ ($\mathrm{T}^{\xi}$) that converges whenever applied to the characteristic term of a ct-type (ct-conjunction) which entails ($\leq_{\mathrm{ct}}$) $\phi$ ($\xi$), and
- a term $\mathrm{D}^{\sigma;\psi_1,\ldots,\psi_n}$ which is used, in the definition of the characteristic terms for normal conjunctions $\xi_1 \to \psi_1 \wedge \cdots \wedge \xi_n \to \psi_n$ of $n \geq 2$ ct-types (in Fig. 5.8 point 2.), to delay the evaluation of the application of the characteristic terms (of each ct-type $\xi_i \to \psi_i$, $1 \leq i \leq n$ of the conjunction) until all the arguments required (by some of these characteristic terms) have been collected.

Note that the definition of the test terms for ct-types is to some extent dual to the corresponding definitions for cs-types. Moreover, the characteristic terms for ct-types (in Fig. 5.7) do *not* delay the evaluation of the application of the test terms to the arguments of a characteristic term until all these arguments have been collected (as the characteristic terms for cs-types do, by means of the auxiliary term $\mathrm{B}^{\phi}$, see Fig. 5.3), since they must be evaluated in and.

**Definition 5.40** *Let $\phi$ be a ct-type and $\xi$ be a conjunction of $n \geq 2$ ct-types. The terms $\mathrm{M}^{\phi}$, $\mathrm{T}^{\phi}$, $\mathrm{M}^{\xi}$, and $\mathrm{T}^{\xi}$ are defined according to the clauses in Figs. 5.7 and 5.8, where (for sake of readability) the* else *branches associated to conditions that never evaluates to* false *(i.e. either divergent or evaluating to* true*) are filled with the symbol $\sharp$.*

**Example 5.41** *Take the (normal) conjunction $\xi = \phi_1 \wedge \phi_2 \wedge \phi_3$, where*

1. $\phi_1 = (\not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \psi_1$,
   *and* $\psi_1 = \not\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$;
2. $\phi_2 = (\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \psi_2$,
   *and* $\psi_2 = \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$;
3. $\phi_3 = (\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \psi_3$,
   *and* $\psi_3 = \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}\to\mathsf{nat}}$.

*In the following we will write (for short) $\not\perp$ instead of $\not\perp^{\mathsf{nat}}$ and $\top$ instead of $\top^{\mathsf{nat}}$.*

*According to Definition 5.40 we have*

$$\mathrm{M}^{\xi} = \lambda x^{\mathsf{nat}\to\mathsf{nat}}.\mathrm{D}^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat};\psi_1,\psi_2,\psi_3}(\mathrm{M}^{\phi_1}x)(\mathrm{M}^{\phi_2}x)(\mathrm{M}^{\phi_3}x)$$

*where*

1. $\mathrm{M}^{\phi_1} = \lambda x^{\mathsf{nat}\to\mathsf{nat}}.\mathsf{if}\ \mathrm{T}^{\not\perp\to\not\perp}x\ \mathsf{then}\ \mathrm{M}^{\psi_1}\ \mathsf{else}\ \sharp$
   - $\mathrm{M}^{\psi_1} = \lambda x^{\mathsf{nat}}.\mathsf{if}\ \mathrm{T}^{\not\perp}x\ \mathsf{then}\ \mathrm{M}^{\top\to\not\perp}\ \mathsf{else}\ \sharp$
     *and* $\mathrm{M}^{\top\to\not\perp} = \lambda x^{\mathsf{nat}}.\mathsf{if}\ \mathrm{T}^{\top}x\ \mathsf{then}\ \mathrm{M}^{\not\perp}\ \mathsf{else}\ \sharp$
   - $\mathrm{T}^{\not\perp\to\not\perp} = \lambda z^{\mathsf{nat}}.\mathrm{T}^{\not\perp}(z\mathrm{M}^{\not\perp})$;
2. $\mathrm{M}^{\phi_2} = \lambda x^{\mathsf{nat}\to\mathsf{nat}}.\mathsf{if}\ \mathrm{T}^{\top\to\not\perp}x\ \mathsf{then}\ \mathrm{M}^{\psi_2}\ \mathsf{else}\ \sharp$
   - $\mathrm{M}^{\psi_2} = \lambda x^{\mathsf{nat}}.\mathsf{if}\ \mathrm{T}^{\top}x\ \mathsf{then}\ \mathrm{M}^{\not\perp\to\not\perp}\ \mathsf{else}\ \sharp$
     *and* $\mathrm{M}^{\not\perp\to\not\perp} = \lambda x^{\mathsf{nat}}.\mathsf{if}\ \mathrm{T}^{\not\perp}x\ \mathsf{then}\ \mathrm{M}^{\not\perp}\ \mathsf{else}\ \sharp$
   - $\mathrm{T}^{\top\to\not\perp} = \lambda z^{\mathsf{nat}}.\mathrm{T}^{\not\perp}(z\mathrm{M}^{\top})$;
3. $\mathrm{M}^{\phi_3} = \lambda x^{\mathsf{nat}\to\mathsf{nat}}.\mathsf{if}\ \mathrm{T}^{\not\perp^{\mathsf{nat}\to\mathsf{nat}}}x\ \mathsf{then}\ \mathrm{M}^{\psi_3}\ \mathsf{else}\ \sharp$
   - $\mathrm{M}^{\psi_3} = \lambda x^{\mathsf{nat}}.\mathsf{if}\ \mathrm{T}^{\top}x\ \mathsf{then}\ \mathrm{M}^{\not\perp^{\mathsf{nat}\to\mathsf{nat}}}\ \mathsf{else}\ \sharp$;
4. $\mathrm{D}^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat};\psi_1,\psi_2,\psi_3} = \lambda p_1^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.\lambda p_2^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.\lambda p_3^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.$

   $$\begin{array}{l}\mathsf{if}\quad \mathsf{por}(\mathsf{isdef}(p_1),\mathsf{isdef}(p_2),\mathsf{isdef}(p_3))\\ \quad\ \mathsf{then}\ \lambda x^{\mathsf{nat}}.\mathrm{D}^{\mathsf{nat}\to\mathsf{nat};\top\to\not\perp,\not\perp\to\not\perp,\not\perp^{\mathsf{nat}\to\mathsf{nat}}}(p_1x)(p_2x)(p_3x)\ \ \mathsf{else}\ \sharp\end{array}$$

   - $\mathrm{D}^{\mathsf{nat}\to\mathsf{nat};\top\to\not\perp,\not\perp\to\not\perp,\not\perp^{\mathsf{nat}\to\mathsf{nat}}} = \lambda q_1^{\top\to\not\perp}.\lambda q_2^{\not\perp\to\not\perp}.\lambda q_3^{\not\perp^{\mathsf{nat}\to\mathsf{nat}}}.$

     $$\mathsf{if}\ \mathsf{por}(\mathsf{isdef}(q_1),\mathsf{isdef}(q_2),\mathsf{isdef}(q_3))\ \ \mathsf{then}\ \lambda y^{\mathsf{nat}}.\mathrm{D}^{\mathsf{nat};}\ \ \mathsf{else}\ \sharp.$$

The following proposition can be easily proved by induction on ct-types.

**Proposition 5.42** *Let $\xi = \phi_1 \wedge \cdots \wedge \phi_n$ be a conjunction of $n \geq 1$ ct-types (note that, for $n = 1$, $\xi \in \mathbf{L}^{\mathrm{ct}}$) and $\sigma = \epsilon(\xi)$.*

1. $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\xi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$.
2. $[\mathrm{T}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\xi \to \not\perp^{\mathsf{bool}}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$ *and, for every* $[M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in \mathbf{I}(\epsilon(\xi))^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, $[\mathrm{T}^{\xi}M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in \{[\mathsf{fix}\,x^{\epsilon(\xi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}, [\mathsf{true}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}\}$.
3. $[\mathrm{D}^{\sigma;}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\top^{\sigma}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,
   $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\wedge_{i\in\{1,\ldots,n\}}\top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{i-1})} \to \phi_i \to \top^{\epsilon(\phi_{i+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \phi_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, *and*
   $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}(\mathsf{fix}\,x^{\epsilon(\phi_1)}.x)\cdots(\mathsf{fix}\,x^{\epsilon(\phi_n)}.x)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{fix}\,x^{\sigma}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$. □

- If $\phi \cong_{\mathrm{ct}} \top^{\rho}$, then
  - $\mathrm{M}^{\phi} = \mathsf{fix}\, x^{\rho}.x$
  - $\mathrm{T}^{\phi} = \lambda z^{\rho}.\mathsf{true}$
- If $\phi = \not\downarrow^{\rho}$, then
  - $\mathrm{M}^{\phi} = \begin{cases} 0 & \text{if } \rho = \mathsf{nat} \\ \mathsf{true} & \text{if } \rho = \mathsf{bool} \\ \lambda x^{\rho_1}.\mathsf{fix}\, y^{\rho_2}.y & \text{if } \rho = \rho_1 \to \rho_2 \\ \langle \mathsf{fix}\, x^{\rho_1}.x, \mathsf{fix}\, y^{\rho_2}.y \rangle & \text{if } \rho = \rho_1 \times \rho_2 \end{cases}$
  - $\mathrm{T}^{\phi} = \lambda z^{\rho}.\mathsf{isdef}(z)$
- If $\phi = \phi_1 \times \phi_2 \ncong_{\mathrm{ct}} \top^{\epsilon(\phi)}$, then
  - $\mathrm{M}^{\phi} = \langle \mathrm{M}^{\phi_1}, \mathrm{M}^{\phi_2} \rangle$
  - $\mathrm{T}^{\phi} = \lambda z^{\epsilon(\phi)}.\mathsf{and}(\mathrm{T}^{\phi_1}(\mathsf{proj}_1 z), \mathrm{T}^{\phi_2}(\mathsf{proj}_2 z))$
- If $\phi = \xi \to \psi \ncong_{\mathrm{ct}} \top^{\epsilon(\phi)}$, then
  - $\mathrm{M}^{\phi} = \lambda x^{\epsilon(\xi)}.\mathsf{if}\ \mathrm{T}^{\xi} x\ \mathsf{then}\ \mathrm{M}^{\psi}\ \mathsf{else}\ \sharp$
  - $\mathrm{T}^{\phi} = \lambda z^{\epsilon(\phi)}.\mathrm{T}^{\psi}(z\mathrm{M}^{\xi})$

Figure 5.7: Terms $\mathrm{M}^{\phi}$ and $\mathrm{T}^{\phi}$

Let $\mathbf{norm}(\xi) = \phi_1 \wedge \cdots \wedge \phi_n$. If $n = 1$ then $\mathrm{M}^{\xi} = \mathrm{M}^{\phi_1}$ and $\mathrm{T}^{\xi} = \mathrm{T}^{\phi_1}$.
Otherwise ($n \geq 2$)

$$\mathrm{T}^{\xi} = \lambda z^{\epsilon(\xi)}.\mathsf{and}(\mathrm{T}^{\phi_1} z, \ldots, \mathrm{T}^{\phi_n} z)$$

and for $\mathrm{M}^{\xi}$ there are 2 cases.

1. If $\phi_i = \psi_i \times \psi_i'$ $(1 \leq i \leq n)$ then
   - $\mathrm{M}^{\xi} = \langle \mathrm{M}^{\psi_1 \wedge \cdots \wedge \psi_n}, \mathrm{M}^{\psi_1' \wedge \cdots \wedge \psi_n'} \rangle$
2. If $\phi_i = \xi_i \to \psi_i$ $(1 \leq i \leq n)$ then
   - $\mathrm{M}^{\xi} = \lambda x^{\epsilon(\xi_1)}.\mathrm{D}^{\epsilon(\psi_1);\psi_1,\ldots,\psi_n}(\mathrm{M}^{\xi_1 \to \psi_1} x) \cdots (\mathrm{M}^{\xi_n \to \psi_n} x)$

where the term $\mathrm{D}^{\sigma;\psi_1,\ldots,\psi_n}$ ($n \geq 0$ and $\sigma = \epsilon(\psi_1) = \cdots = \epsilon(\psi_n)$) is inductively defined as follows.

- If $n = 0$, then
  - $\mathrm{D}^{\sigma;} = \mathsf{fix}\, x^{\sigma}.x$
- Otherwise ($n \geq 1$) let

$$\begin{array}{rcl} \{i_1, \ldots, i_h\} & = & \{i \mid 1 \leq i \leq n \text{ and } \psi_i \not\cong_{\mathrm{ct}} \top^{\sigma}\} \\ \{j_1, \ldots, j_k\} & = & \{j \mid 1 \leq j \leq n \text{ and } \psi_j \neq \not\perp^{\sigma} \textit{ and } \psi_j \not\cong_{\mathrm{ct}} \top^{\sigma}\} \end{array}$$

  (note that either $k = 0$ or $\psi_{j_1} \wedge \cdots \wedge \psi_{j_k} = \mathbf{norm}(\psi_1 \wedge \cdots \wedge \psi_n)$) in

  - $\mathrm{D}^{\iota;\psi_1,\ldots,\psi_n} = \lambda p_1^{\iota}. \cdots \lambda p_n^{\iota}.\mathsf{if}\ \mathsf{por}_{i \in \{i_1,\ldots,i_h\}} \mathsf{isdef}(p_i)\ \mathsf{then}\ \mathrm{C}^{\iota}_{\not\perp}\ \mathsf{else}\ \sharp$,
    where $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$
  - $\mathrm{D}^{\sigma_1 \times \sigma_2;\psi_1,\ldots,\psi_n} = \lambda p_1^{\sigma_1 \times \sigma_2}. \cdots \lambda p_n^{\sigma_1 \times \sigma_2}.\mathsf{if}\ \mathsf{por}_{i \in \{i_1,\ldots,i_h\}} \mathsf{isdef}(p_i)$

    $$\begin{array}{ll} \mathsf{then} & \langle \mathrm{D}^{\sigma_1;\chi_{j_1},\ldots,\chi_{j_k}}(\mathsf{proj}_1 p_{j_1}) \cdots (\mathsf{proj}_1 p_{j_k}), \\ & \quad \mathrm{D}^{\sigma_2;\chi_{j_1}',\ldots,\chi_{j_k}'}(\mathsf{proj}_2 p_{j_1}) \cdots (\mathsf{proj}_2 p_{j_k}) \rangle \\ \mathsf{else}\ \sharp & \end{array}$$

    where, for $j \in \{j_1, \ldots, j_k\}$, $\psi_j = \chi_j \times \chi_j'$
  - $\mathrm{D}^{\sigma_1 \to \sigma_2;\psi_1,\ldots,\psi_n} = \lambda p_1^{\sigma_1 \to \sigma_2}. \cdots \lambda p_n^{\sigma_1 \to \sigma_2}.$

    $$\mathsf{if}\ \mathsf{por}_{i \in \{i_1,\ldots,i_h\}} \mathsf{isdef}(p_i)\ \mathsf{then}\ \lambda x^{\sigma_1}.\mathrm{D}^{\sigma_2;\chi_{j_1},\ldots,\chi_{j_k}}(p_{j_1} x) \cdots (p_{j_k} x)\ \mathsf{else}\ \sharp$$

    where, for $j \in \{j_1, \ldots, j_k\}$, $\psi_j = \nu_j \to \chi_j$

Figure 5.8: Terms $\mathrm{M}^{\xi}$ and $\mathrm{T}^{\xi}$

The proof of completeness rely on the following lemmas.

**Lemma 5.43** *Let* $\xi = \phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{C}^{\mathrm{ct}}$.

1. $\xi \cong_{\mathrm{ct}} \top^{\epsilon(\xi)}$ *if and only if* $[\mathsf{fix}\, x^{\epsilon(\xi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\xi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$.
2. $\xi \cong_{\mathrm{ct}} \top^{\epsilon(\xi)}$ *if and only if* $\mathrm{T}^{\xi}(\mathsf{fix}\, x^{\epsilon(\xi)}.x) \Downarrow \mathsf{true}$.

**Proof.** First notice that

- (by Definition 5.2) $[\mathsf{fix}\, x^{\epsilon(\xi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\xi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$ if and only if, for all $i \in \{1, \ldots, n\}$, $[\mathsf{fix}\, x^{\epsilon(\xi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\phi_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, and
- (by Definition 5.40) $\mathrm{T}^{\xi}\mathrm{C}^{\epsilon(\xi)}_{\top} \Downarrow \mathsf{true}$ if and only if, for all $i \in \{1, \ldots, n\}$, $\mathrm{T}^{\phi_i}\mathrm{C}^{\epsilon(\xi)}_{\top} \Downarrow \mathsf{true}$.

So, by Proposition 5.35, it suffice to prove the property for $n = 1$.

1. *(only if)* Immediate by soundness (Theorem 5.34).
   *(if)* The case $\phi \cong_{\mathrm{ct}} \top^{\epsilon(\phi)}$ is immediate. The other cases are by structural induction on $\phi \in \mathbf{L}^{\mathrm{ct}}$.
   The case $\phi = \not\perp^{\epsilon(\phi)}$ is immediate.
   Let $\phi = \nu \to \psi \ncong_{\mathrm{ct}} \top^{\epsilon(\phi)}$, then $\psi \ncong_{\mathrm{ct}} \top^{\epsilon(\psi)}$ (by Proposition 5.35). By induction hypothesis $[\mathsf{fix}\, y^{\epsilon(\psi)}.y]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$. So, for every $[M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, $[(\mathsf{fix}\, x^{\epsilon(\phi)}.x)M]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{fix}\, y^{\epsilon(\psi)}.y]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\psi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, which implies $[\mathsf{fix}\, x^{\epsilon(\phi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\phi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$.
   The case $\phi = \phi_1 \times \phi_2 \ncong_{\mathrm{cs}} \top^{\epsilon(\phi)}$ is similar.

2. *(only if)* Immediate, since $\phi \cong_{\mathrm{ct}} \top^{\epsilon(\phi)}$ implies $\mathrm{T}^{\phi} = \lambda z^{\epsilon(\phi)}.\mathsf{true}$.
   *(if)* The case $\phi \cong_{\mathrm{ct}} \top^{\epsilon(\phi)}$ is immediate. The other cases are by structural induction on $\phi \in \mathbf{L}^{\mathrm{ct}}$.
   The case $\phi = \not\perp^{\epsilon(\phi)}$ is immediate.
   Let $\phi = \nu \to \psi \ncong_{\mathrm{ct}} \top^{\epsilon(\phi)}$, then $\psi \ncong_{\mathrm{ct}} \top^{\epsilon(\psi)}$ (by Proposition 5.35). Since

   $$[\mathrm{T}^{\nu\to\psi}(\mathsf{fix}\, x^{\epsilon(\nu\to\psi)}.x)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathrm{T}^{\psi}(\mathsf{fix}\, y^{\epsilon(\psi)}.y)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},$$

   the result follows by induction hypotheses.
   The case $\phi = \phi_1 \times \phi_2 \ncong_{\mathrm{ct}} \top^{\epsilon(\phi)}$ is similar. □

**Lemma 5.44** *Let* $\xi = \phi_1 \wedge \cdots \wedge \phi_n$ *and* $\nu = \psi_1 \wedge \cdots \wedge \psi_n$ *be ct-conjunctions such that* $\epsilon(\xi) = \epsilon(\nu) = \sigma$.

1. *If* $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$ *then* $\xi \leq_{\mathrm{ct}} \nu$.
2. *If* $\mathrm{T}^{\nu}\mathrm{M}^{\xi} \Downarrow \mathsf{true}$ *then* $\xi \leq_{\mathrm{ct}} \nu$.

*3. If* $\forall i \in \{1,\ldots,m\}.\exists j_i \in \{1,\ldots,n\}.\ [\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in$

$$[\![\top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{j_i-1})} \to \phi_{j_i} \to \top^{\epsilon(\phi_{j_i+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \psi_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},$$

*then* $\xi \leq_{\mathrm{ct}} \nu$.

**Proof.** First notice that $\xi \leq_{\mathrm{ct}} \nu$ if and only if $\forall i \in \{1,\ldots,m\}.\exists j_i \in \{1,\ldots,n\}.\phi_{j_i} \leq_{\mathrm{ct}} \psi_i$. We first consider the case in which either $\nu \cong_{\mathrm{ct}} \top^{\epsilon(\nu)}$ or $\xi \cong_{\mathrm{ct}} \top^{\epsilon(\xi)}$.
Let $\underline{\nu \cong_{\mathrm{ct}} \top^{\epsilon(\nu)}}$. All the 3 points are immediate.
Let $\underline{\xi \cong_{\mathrm{ct}} \top^{\epsilon(\xi)}}$. The case $\nu \cong_{\mathrm{ct}} \top^{\epsilon(\nu)}$ has been already considered, so let $\nu \ncong_{\mathrm{ct}} \top^{\epsilon(\nu)}$. Points 1. and 2. are obvious since (by Lemma 5.43) $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{fix}\, x^{\epsilon(\xi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$ and $\mathrm{T}^{\nu}(\mathsf{fix}\, x^{\epsilon(\xi)}.x) \Uparrow$. For point 3. observe that

$$\begin{aligned}[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} &= [\lambda p_1^{\epsilon(\phi_1)}.\cdots\lambda p_n^{\epsilon(\phi_n)}.\mathsf{fix}\, x^{\epsilon(\xi)}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \\ &\notin [\![\wedge_{i\in\{1,\ldots,n\}}\mathbf{min}_{\mathrm{t}}(\epsilon(\phi_1)) \to \cdots \to \mathbf{min}_{\mathrm{t}}(\epsilon(\phi_n)) \to \psi_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}.\end{aligned}$$

If $\nu \ncong_{\mathrm{ct}} \top^{\epsilon(\nu)}$ and $\xi \ncong_{\mathrm{ct}} \top^{\epsilon(\xi)}$ the proof is by structural induction on $\mathbf{norm}(\nu)$ and $\mathbf{norm}(\xi)$.

Let $\underline{\mathbf{norm}(\nu) = \not\perp^{\epsilon(\nu)}}$. The case $\mathbf{norm}(\xi) \cong_{\mathrm{ct}} \top^{\epsilon(\xi)}$ has been already considered, and the case $\mathbf{norm}(\xi) = \not\perp^{\epsilon(\xi)}$ is trivial.
If $\mathbf{norm}(\xi) = \xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p$ then $\xi \leq_{\mathrm{ct}} \nu$. The 3 points are obvious since, for $p = 1$,

1. $[\mathrm{M}^{\xi_1\to\phi'_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\not\perp^{\epsilon(\nu)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,
2. $[\mathrm{T}^{\nu}(\mathrm{M}^{\xi_1\to\phi'_1})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{isdef}(\mathrm{M}^{\xi_1\to\phi'_1})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{true}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, and
3. $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{j-1})} \to \phi_j \to \top^{\epsilon(\phi_{j+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \not\perp^{\epsilon(\nu)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, for some $j$ such that $\phi_j \ncong_{\mathrm{ct}} \top^{\epsilon(\phi_j)}$.

while, for $p \geq 2$,

1. $[\mathrm{M}^{\xi_1\to\phi'_1\wedge\cdots\wedge\xi_p\to\phi'_p}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\not\perp^{\epsilon(\nu)}]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,
2. $[\mathrm{T}^{\nu}(\mathrm{M}^{\xi_1\to\phi'_1\wedge\cdots\wedge\xi_p\to\phi'_p})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{isdef}(\mathrm{M}^{\xi_1\to\phi'_1\wedge\cdots\wedge\xi_p\to\phi'_p})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, and
3. is as for the case $p = 1$.

The case $\mathbf{norm}(\xi) = \phi_1 \times \phi'_1 \wedge \cdots \wedge \phi_p \times \phi'_p$ is similar.

Let $\underline{\mathbf{norm}(\xi) = \not\perp^{\epsilon(\xi)}}$. The case $\mathbf{norm}(\nu) \cong_{\mathrm{ct}} \top^{\epsilon(\nu)}$ and the case $\mathbf{norm}(\nu) = \not\perp^{\epsilon(\nu)}$ have been already considered.
If $\mathbf{norm}(\nu) = \nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q$ then $\xi \not\leq_{\mathrm{t}} \nu$. The 3 points are obvious since, for $q = 1$,

1. $[\mathrm{M}^{\not\perp^{\epsilon(\xi)}}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathrm{C}^{\epsilon(\xi)}_{\not\perp}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,

2. $[\mathrm{T}^{\nu_1\to\psi'_1}(\mathrm{M}^{\not\perp^{\epsilon(\xi)}})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathrm{T}^{\psi'_1}(\mathsf{fix}\, y^{\epsilon(\psi'_1)}.y)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{fix}\, x^{\mathsf{bool}}.x]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$ (by Lemma 5.43.2, since $\psi'_1 \not\cong_{\mathrm{ct}} \top^{\epsilon(\nu)}$), and

3. $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\mathbf{min}_{\mathrm{t}}(\epsilon(\phi_1)) \to \cdots \to \mathbf{min}_{\mathrm{t}}(\epsilon(\phi_n)) \to \nu_1 \to \psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,

while, for $q \geq 2$,

1. $[\mathrm{M}^{\not\perp^{\epsilon(\xi)}}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathrm{C}^{\epsilon(\xi)}_{\not\perp}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$,

2. $[\mathrm{T}^{\nu}(\mathrm{M}^{\not\perp^{\epsilon(\xi)}})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{and}(\mathrm{T}^{\psi_1}(\mathrm{M}^{\not\perp^{\epsilon(\xi)}}), \ldots, \mathrm{T}^{\psi_n}(\mathrm{M}^{\not\perp^{\epsilon(\xi)}}))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, and

3. $[\mathrm{D}^{\sigma;\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \notin [\![\wedge_{i\in\{1,\ldots,q\}}\mathbf{min}_{\mathrm{t}}(\epsilon(\phi_1)) \to \cdots \to \mathbf{min}_{\mathrm{t}}(\epsilon(\phi_n)) \to \nu_i \to \psi'_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$.

The case $\mathbf{norm}(\nu) = \psi_1 \times \psi'_1 \wedge \cdots \wedge \psi_q \times \psi'_q$ is analogous.

Let $\mathbf{norm}(\nu) = \nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q$. All the cases but $\mathbf{norm}(\xi) = \xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p$ have been already considered. In this case $\xi \leq_{\mathrm{ct}} \nu$ if and only if $\forall i \in \{1,\ldots,q\}.\exists j \in \{1,\ldots,p\}$. both $\nu'_j \leq_{\mathrm{ct}} \xi'_i$ and $\phi'_j \leq_{\mathrm{ct}} \psi'_i$.
Then the 3 points can be proved as follows.
Case $p = 1$ and $q = 1$.

1. $[\mathrm{M}^{\xi_1\to\phi'_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\nu_1 \to \psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$ implies $[\mathrm{M}^{\xi_1\to\phi'_1}\mathrm{M}^{\nu_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\nu_1 \to \psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$. But
$$[\mathrm{M}^{\xi_1\to\phi'_1}\mathrm{M}^{\nu_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} = [\mathsf{if}\ \mathrm{T}^{\xi_1}\mathrm{M}^{\nu_1}\ \mathsf{then}\ \mathrm{M}^{\phi'_1}\ \mathsf{else}\ \sharp]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},$$
and (by Proposition 5.42 and Lemma 5.43.2, since $\psi'_1 \not\cong_{\mathrm{ct}} \top^{\epsilon(\psi'_1)}$) we have that both $\mathrm{T}^{\xi_1}\mathrm{M}^{\nu_1} \Downarrow \mathsf{true}$ and $[\mathrm{B}^{\phi'_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in [\![\not\perp^{\mathsf{bool}} \to \psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}$, so (by induction hypotheses) we get $\nu_1 \leq_{\mathrm{ct}} \xi_1$ and $\phi'_1 \leq_{\mathrm{ct}} \psi'_1$, which imply $\xi_1 \to \phi'_1 \leq_{\mathrm{ct}} \nu_1 \to \psi'_1$.

2. Assume $\mathrm{T}^{\nu_1\to\psi'_1}\mathrm{M}^{\xi_1\to\phi'_1} \Downarrow \mathsf{true}$. Since
$$\begin{aligned}[\mathrm{T}^{\nu_1\to\psi'_1}\mathrm{M}^{\xi_1\to\phi'_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} &= [\mathrm{T}^{\psi'_1}(\mathrm{M}^{\xi_1\to\phi'_1}\mathrm{M}^{\nu_1})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \\ &= [\mathrm{T}^{\psi'_1}(\mathsf{if}\ \mathrm{T}^{\xi_1}\mathrm{M}^{\nu_1}\ \mathsf{then}\ \mathrm{M}^{\phi'_1}\ \mathsf{else}\ \sharp)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},\end{aligned}$$
then we get (by Proposition 5.42 and Lemma 5.43, since $\psi'_1 \not\cong_{\mathrm{ct}} \top^{\epsilon(\psi'_1)}$, using induction hypotheses) $\nu_1 \leq_{\mathrm{ct}} \xi_1$ and $\phi'_1 \leq_{\mathrm{ct}} \psi'_1$ which imply $\xi_1 \to \phi'_1 \leq_{\mathrm{ct}} \nu_1 \to \psi'_1$.

3. Assume that there is $j_1 \in \{1,\ldots,n\}$ such that $[\mathrm{D}^{\epsilon(\xi);\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \in$
$$[\![\top^{\epsilon(\phi_1)} \to \cdots \to \top^{\epsilon(\phi_{j_1-1})} \to \phi_{j_1} \to \top^{\epsilon(\phi_{j_1+1})} \to \cdots \to \top^{\epsilon(\phi_n)} \to \nu_1 \to \psi'_1]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}}.$$
Note that by Proposition 5.42.3 and Lemma 5.43.1, since $\nu_1 \to \psi'_1 \not\cong_{\mathrm{ct}} \top^{\epsilon(\nu)}$, follows that $\phi_{j_1} = \xi_1 \to \phi'_1$. Since
$$\begin{aligned}[\mathrm{D}^{\epsilon(\xi);\phi_1,\ldots,\phi_n}\mathrm{M}^{\phi_1}\cdots\mathrm{M}^{\phi_n}\mathrm{M}^{\nu_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} &= [\mathrm{D}^{\epsilon(\phi'_1);\phi'_1}\mathrm{M}^{\xi_1\to\phi'_1}\mathrm{M}^{\nu_1}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}} \\ &= [\mathrm{D}^{\epsilon(\phi'_1);\phi'_1}(\mathsf{if}\ \mathrm{T}^{\xi_1}\mathrm{M}^{\nu_1}\ \mathsf{then}\ \mathrm{M}^{\phi'_1}\ \mathsf{else}\ \sharp)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}},\end{aligned}$$
we get (by Proposition 5.42 and Lemma 5.43, since $\psi'_1 \not\cong_{\mathrm{ct}} \top^{\epsilon(\psi'_1)}$, using induction hypotheses) $\nu_1 \leq_{\mathrm{ct}} \xi_1$ and $\phi'_1 \leq_{\mathrm{ct}} \psi'_1$, which imply $\xi_1 \to \phi'_1 \leq_{\mathrm{ct}} \nu_1 \to \psi'_1$.

Case $p, q \geq 2$.

1. $[\mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$ implies, for all $i \in \{1, \ldots, q\}$, $[\mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p} \mathrm{M}^{\nu_i}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\psi'_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$. But $[\mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p} \mathrm{M}^{\nu_i}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} =$

   $$[\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathrm{M}^{\xi_1 \to \phi'_1} \mathrm{M}^{\nu_i}) \cdots (\mathrm{M}^{\xi_p \to \phi'_p} \mathrm{M}^{\nu_i})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$$
   $$= [\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}((\mathsf{if}\ \mathrm{T}^{\xi_1} \mathrm{M}^{\nu_i}\ \mathsf{then}\ \mathrm{M}^{\phi'_1}\ \mathsf{else}\ \sharp)) \cdots ((\mathsf{if}\ \mathrm{T}^{\xi_p} \mathrm{M}^{\nu_i}\ \mathsf{then}\ \mathrm{M}^{\phi'_p}\ \mathsf{else}\ \sharp))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}},$$

   so (by Proposition 5.42 and Lemma 5.43, since $\psi'_i \not\cong_{\mathrm{ct}} \top^{\epsilon(\psi'_i)}$, using induction hypotheses) we get $\forall i \in \{1, \ldots, q\}.\exists j \in \{1, \ldots, p\}.\xi_j \to \phi'_j \leq_{\mathrm{ct}} \nu_i \to \psi'_i$, which imply $\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p \leq_{\mathrm{ct}} \nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q$.

2. Assume $\mathrm{T}^{\nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q} \mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p} \Downarrow \mathsf{true}$. Since

   $$[\mathrm{T}^{\nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q} \mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} =$$
   $$[\mathsf{and}(\mathrm{T}^{\nu_1 \to \psi'_1} \mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p}, \ldots, \mathrm{T}^{\nu_q \to \psi'_q} \mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}},$$

   and, for all $i \in \{1, \ldots, q\}$,

   $$[\mathrm{T}^{\nu_i \to \psi'_i} \mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} =$$
   $$[\mathrm{T}^{\psi'_i}(\mathrm{M}^{\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p} \mathrm{M}^{\nu_i})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} =$$
   $$[\mathrm{T}^{\psi'_1}(\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathrm{M}^{\xi_1 \to \phi'_1} \mathrm{M}^{\nu_i}) \cdots (\mathrm{M}^{\xi_p \to \phi'_p} \mathrm{M}^{\nu_i}))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} =$$
   $$[\mathrm{T}^{\psi'_1}(\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathsf{if}\ \mathrm{T}^{\xi_1} \mathrm{M}^{\nu_i}\ \mathsf{then}\ \mathrm{M}^{\phi'_1}\ \mathsf{else}\ \sharp) \cdots (\mathsf{if}\ \mathrm{T}^{\xi_p} \mathrm{M}^{\nu_i}\ \mathsf{then}\ \mathrm{M}^{\phi'_p}\ \mathsf{else}\ \sharp))]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}},$$

   we get (by Proposition 5.42 and Lemma 5.43, since $\psi'_i \not\cong_{\mathrm{ct}} \top^{\epsilon(\psi'_i)}$, using induction hypotheses) $\forall i \in \{1, \ldots, q\}.\exists j \in \{1, \ldots, p\}.\xi_j \to \phi'_j \leq_{\mathrm{ct}} \nu_i \to \psi'_i$, which imply $\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p \leq_{\mathrm{ct}} \nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q$.

3. Assume that $\forall i \in \{1, \ldots, q\}.\exists j_i \in \{1, \ldots, n\}.\phi_{j_i} \leq_{\mathrm{ct}} \psi_i$. $[\mathrm{D}^{\epsilon(\xi);\phi_1,\ldots,\phi_n}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in$

   $$[\![\top^{\epsilon(\phi_1)} \to \ldots \top^{\epsilon(\phi_{j_i-1})} \to \phi_{j_i} \to \top^{\epsilon(\phi_{j_i+1})} \to \ldots \top^{\epsilon(\phi_n)} \to \nu_i \to \psi'_i]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}.$$

   Note that by Proposition 5.42.3 and Lemma 5.43.1, since, for each $i \in \{1, \ldots, q\}$, $\nu_i \to \psi'_i \not\cong_{\mathrm{ct}} \top^{\epsilon(\nu)}$, follows that $\phi_{j_1}, \ldots, \phi_{j_q} \in \{\xi_1 \to \phi'_1, \ldots, \xi_p \to \phi'_p\}$. Since, for all $i \in \{1, \ldots, q\}$,

   $$[\mathrm{D}^{\epsilon(\xi);\phi_1,\ldots,\phi_n} \mathrm{M}^{\phi_1} \cdots \mathrm{M}^{\phi_n} \mathrm{M}^{\nu'_i}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} =$$
   $$[\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathrm{M}^{\xi_1 \to \phi'_1} \mathrm{M}^{\nu_i}) \cdots (\mathrm{M}^{\xi_p \to \phi'_p} \mathrm{M}^{\nu_i})]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} =$$
   $$[\mathrm{D}^{\epsilon(\phi'_1);\phi'_1,\ldots,\phi'_p}(\mathsf{if}\ \mathrm{T}^{\xi_1} \mathrm{M}^{\nu_i}\ \mathsf{then}\ \mathrm{M}^{\phi'_1}\ \mathsf{else}\ \sharp) \cdots (\mathsf{if}\ \mathrm{T}^{\xi_p} \mathrm{M}^{\nu_i}\ \mathsf{then}\ \mathrm{M}^{\phi'_p}\ \mathsf{else}\ \sharp)]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}},$$

   we get (by Proposition 5.42 and Lemma 5.43, since $\psi'_i \not\cong_{\mathrm{ct}} \top^{\epsilon(\psi'_i)}$, using induction hypotheses) $\forall i \in \{1, \ldots, q\}.\exists j \in \{1, \ldots, p\}.\xi_j \to \phi'_j \leq_{\mathrm{ct}} \nu_i \to \psi'_i$, which imply $\xi_1 \to \phi'_1 \wedge \cdots \wedge \xi_p \to \phi'_p \leq_{\mathrm{ct}} \nu_1 \to \psi'_1 \wedge \cdots \wedge \nu_q \to \psi'_q$.

The case $p = 1$ and $q \geq 2$, and the case $p \geq 2$ and $q = 1$ are similar.

The proof of the only remaining case, $\mathbf{norm}(\nu) = \psi_1 \times \psi'_1 \wedge \cdots \wedge \psi_q \times \psi'_q$ and $\mathbf{norm}(\xi) = \phi_1 \times \phi'_1 \wedge \cdots \wedge \phi_p \times \phi'_p$, is similar to the proof of the previous one. □

From the previous lemma we derive the completeness of the entailment relation.

**Proof of Theorem 5.38**. By contraposition. Assume that $\xi \not\leq_{\mathrm{ct}} \nu$. Then $\nu \not\approx_{\mathrm{ct}} \top^{\epsilon(\psi)}$. By Lemma 5.44.1 $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \notin [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$. Since from Proposition 5.42.1 $[\mathrm{M}^{\xi}]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \in [\![\xi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$, this implies that $[\![\xi]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}} \not\subseteq [\![\nu]\!]^{\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}}$. □

**Remark 5.45** *As showed in Example 5.36 the entailment relation $\leq_{\mathrm{ct}}$ is not complete w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$. At the moment we do not known whether a sound (and decidable) extension of the $\leq_{\mathrm{ct}}$ relation which is complete w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$ exists.*

### 5.3.2 An assignment system for conjunctive totality types

For each constant $\mathsf{c}$ a set $\mathbf{L}^{\mathrm{ct}}(\mathsf{c})$ of minimal ct-types is specified. The idea is that every ct-type which is entailed ($\leq_{\mathrm{ct}}$) by an element of $\mathbf{L}^{\mathrm{ct}}(\mathsf{c})$ can be assigned to $\mathsf{c}$. For example, for all integers $\mathsf{n}$, $\mathbf{L}^{\mathrm{ct}}(\mathsf{n}) = \{\not\perp^{\mathsf{nat}}\}$ and $\mathbf{L}^{\mathrm{ct}}(+) = \{\not\perp^{\mathsf{nat}} \times \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}\}$.

**Definition 5.46 (Totality type assignment system)** *We write $\Sigma \vdash_{\mathrm{ct}} M : \phi$ to mean that $\Sigma \vdash M : \phi$ can be derived, using only ct-types, by the rules in Fig. 5.9.*

The proof of the following fact is immediate.

**Fact 5.47** *1. $\Sigma \vdash_{\mathrm{ct}} M : \phi$ implies $\vdash_{\mathbf{T}} M : \epsilon(\phi)$ and $\epsilon(\Sigma) \supseteq \mathrm{FV}(M)$.*

*2. $\vdash_{\mathbf{T}} M : \rho$ implies, for all basis $\Sigma$ such that $\Sigma = \{x^{\epsilon(\xi)} : \xi \mid x^{\epsilon(\xi)} \in \mathrm{FV}(M)$ for some $\xi \in \mathbf{C}^{\mathrm{ct}}\}$, $\Sigma \vdash_{\mathrm{ct}} M : \top^{\rho}$.* □

The system $\vdash_{\mathrm{ct}}$ is more powerful than $\vdash_{\mathrm{t}}$, as the following example shows.

**Example 5.48** *Let*

- $\vdash_{\mathbf{T}} P : \rho$,
- $\vdash_{\mathbf{T}} Q : (\mathsf{nat} \to \mathsf{nat}) \to \mathsf{nat} \to \mathsf{nat}$, *and*
- $\vdash_{\mathbf{T}} PQ : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$,

*where*

- $\rho = ((\mathsf{nat} \to \mathsf{nat}) \to \mathsf{nat} \to \mathsf{nat}) \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$,
- $P = \lambda g^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}\to\mathsf{nat}}.\lambda u^{\mathsf{nat}}.\lambda v^{\mathsf{nat}}.g(\lambda w^{\mathsf{nat}}.w)u + g(\lambda z^{\mathsf{nat}}.3)v$, *and*
- $Q = \lambda f^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.fx$.

*Let*

$$(\leq)\ \frac{\Sigma \vdash M : \phi \quad \phi \leq \psi}{\Sigma \vdash M : \psi}$$

$$(\text{Var})\ \frac{}{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash x : \phi_i}\ 1 \leq i \leq n \qquad (\text{Con})\ \frac{\phi \in \mathbf{L}^{\text{ct}}(\mathsf{c})}{\Sigma \vdash \mathsf{c} : \phi}$$

$$(\to \text{I})\ \frac{\Sigma, x : \xi \vdash M : \psi}{\Sigma \vdash \lambda x.M : \xi \to \psi} \qquad (\to \text{I}_{\not\downarrow})\ \frac{\vdash_{\mathbf{T}} \lambda x.M : \rho}{\Sigma \vdash \lambda x.M : \not\downarrow^{\rho}}$$

$$(\to \text{E})\ \frac{\Sigma \vdash M : \phi_1 \wedge \cdots \wedge \phi_n \to \psi \quad \Sigma \vdash N : \phi_1 \ \cdots\ \Sigma \vdash N : \phi_n}{\Sigma \vdash MN : \psi}$$

$$(\times \text{I})\ \frac{\Sigma \vdash M_1 : \phi_1 \quad \Sigma \vdash M_2 : \phi_2}{\Sigma \vdash \langle M_1, M_2 \rangle : \phi_1 \times \phi_2} \qquad (\times \text{I}_{\not\downarrow})\ \frac{\vdash_{\mathbf{T}} \langle M_1, M_2 \rangle : \rho}{\Sigma \vdash \langle M_1, M_2 \rangle : \not\downarrow^{\rho}}$$

$$(\times \text{E}_i)\ \frac{\Sigma \vdash M : \phi_1 \times \phi_2}{\Sigma \vdash \mathsf{proj}_i M : \phi_i}\ i \in \{1, 2\}$$

$$(\text{Fix}_{\not\downarrow})\ \frac{\begin{array}{c}\Sigma, x : \top^{\rho} \vdash M : \phi_1 \quad \Sigma, x : \phi_1 \vdash M : \phi_2 \\ \Sigma, x : \phi_1 \wedge \phi_2 \vdash M : \phi_3 \quad \cdots \quad \Sigma, x : \phi_1 \wedge \cdots \wedge \phi_{n-1} \vdash M : \phi_n\end{array}}{\Sigma \vdash \mathsf{fix}\, x.M : \phi_n}\ n \geq 1$$

$$(\text{If}_{\not\downarrow})\ \frac{\Sigma \vdash N : \not\downarrow^{\mathsf{bool}} \quad \Sigma \vdash M_1 : \phi \quad \Sigma \vdash M_2 : \phi}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \phi}$$

$$(\text{If}_{\top})\ \frac{\Sigma \vdash N : \top^{\mathsf{bool}} \quad \vdash_{\mathbf{T}} M_1 : \rho \quad \vdash_{\mathbf{T}} M_2 : \rho}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \top^{\rho}}$$

Figure 5.9: Rules for ct-type assignment (system $\vdash_{\text{ct}}$)

- $\psi_1 = (\not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$, *and*
- $\psi_2 = (\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$.

*It is easy to check that*

- $\vdash_{\mathrm{ct}} P : (\psi_1 \wedge \psi_2) \to \not\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$,
- *for* $i \in \{1,2\}$, $\vdash_{\mathrm{ct}} Q : \psi_i$, *and*
- $\vdash_{\mathrm{ct}} PQ : \not\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$,

*while in* $\vdash_{\mathrm{t}}$ *it is not possible to prove that* $PQ$ *has the property* $\not\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$.

The rule ($\mathrm{Fix}_{\not\perp}$) of the system $\vdash_{\mathrm{ct}}$ is more powerful of the corresponding rule of the system $\vdash_{\mathrm{t}}$ (in Figure 4.6), as following example shows.

**Example 5.49** *Let* $\vdash_{\mathbf{T}} \mathsf{fix}\, f^{\rho}.M : \rho$, *where*

- $\rho = \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to (\mathsf{nat} \times (\mathsf{nat} \times \mathsf{nat}))$, *and*
- $M = \lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\lambda z^{\mathsf{nat}}.\mathsf{if}\ z = 0$
  $\mathsf{then}\ \langle \mathsf{proj}_2(\mathsf{proj}_2(fxy(z-1))),\ \langle \mathsf{proj}_2(\mathsf{proj}_2(fyx(z-1))), x\rangle\rangle$
  $\mathsf{else}\ \langle \mathsf{proj}_2(\mathsf{proj}_1(fyx(z-1))),\ \langle y, 3\rangle\rangle$ .

*Let*

- $\phi_1 = \not\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to (\top^{\mathsf{nat}} \times (\top^{\mathsf{nat}} \times (\not\perp^{\mathsf{nat}}))$,
- $\phi_2 = \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to (\top^{\mathsf{nat}} \times (\not\perp^{\mathsf{nat}} \times (\top^{\mathsf{nat}}))$, *and*
- $\phi_3 = \not\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to (\not\perp^{\mathsf{nat}} \times (\top^{\mathsf{nat}} \times (\not\perp^{\mathsf{nat}}))$.

*We have that:* $\{f : \top^{\rho}\} \vdash_{\mathrm{ct}} M : \phi_1$, $\{p : \phi_1\} \vdash_{\mathrm{ct}} M : \phi_2$, *and* $\{p : \phi_1 \wedge \phi_2\} \vdash_{\mathrm{ct}} M : \phi_3$. *So, by rule* ($\mathrm{Fix}_{\not\perp}$), *we have:* $\emptyset \vdash_{\mathrm{ct}} \mathsf{fix}\, f^{\rho}.M : \phi_3$.

*Instead the rule* ($\mathrm{Fix}_{\not\perp}$) *of the system* $\vdash_{\mathrm{t}}$ *does not allow to assign the property* $\phi_3$ *to* $\mathsf{fix}\, f^{\rho}.M$.

Soundness of the ct-types assignment system (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$) is stated by the following theorem.

**Theorem 5.50 (Soundness of $\vdash_{\mathrm{ct}}$)** *Let* $\Sigma \vdash_{\mathrm{ct}} M : \phi$. *Then* $[\![M]\!]_e \in [\![\phi]\!]$ *for all* $e$ *such that for all* $x : \xi \in \Sigma$, $e(x) \in [\![\xi]\!]$.

**Proof.** By induction on the derivation of $\Sigma \vdash_{\mathrm{ct}} M : \phi$. □

**Remark 5.51 (A checking algorithm for totality)** *By using the technique proposed in [Hankin and Le Métayer, 1994b] (see Remark 5.32), it should be possible to build an efficient checking algorithm for the system* $\vdash_{\mathrm{ct}}$. *In [Solberg, 1995a] Chapter 4 and in [Solberg, 1995b] a checking algorithm for the system for combined strictness and totality discussed in Section 5.4.4 is presented. This algorithm uses some of the ideas introduced in [Hankin and Le Métayer, 1994b], however, due to the complexity of the combined conjunctive strictness and totality types and inference rules, many of the optimizations proposed in [Hankin and Le Métayer, 1994b] are lost.*

- $\phi \in \mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}$, if $\phi \in \mathbf{L}^{\mathrm{ct}}_{\not\perp}$
- $\phi_1 \times \phi_2 \in \mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}$, if either $\phi_1 \in \mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}$ or $\phi_2 \in \mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}$
- $(\phi_1 \wedge \cdots \wedge \phi_n) \to \psi \in \mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}$, if $\psi \in \mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}$ and either $\phi_1, \ldots, \phi_n \in \mathbf{L}^{\mathrm{cst}}_{\perp\top}$ or $\phi_1, \ldots, \phi_n \in \mathbf{L}^{\mathrm{cst}}_{\not\perp\top}$

Figure 5.10: Set $\mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}$

### 5.3.3 Conjunctive totality analysis and ground contextual equivalence

The same considerations done for totality analysis without conjunction in Section 4.3.3 apply also to conjunctive totality analysis.

## 5.4 Combined conjunctive strictness and totality analysis

In this section we return to the language of strictness and totality types $\mathbf{L}^{\mathrm{cst}}$ introduced in Section 5.1 and consider the problem of defining an entailment relation for the combined language.

The aim of this section is to show that, even if we spend some effort in defining an assignment system for a language of combined strictness and totality properties (obtaining a new system which extends the one presented in [Solberg, 1995b] and [Solberg, 1995a] Chapter 3), the weakness of type assignment rules does not allow to use in an essential way the (greater) expressive power of the combined language of properties.

### 5.4.1 An entailment relation for conjunctive strictness and totality types

A simple relation of entailment for cst-types, $\leq_{\mathrm{cst}}$, is defined by taking the rules in Fig. 5.2 (or 5.6), by removing the axioms $(\perp)$ (resp. $(\not\perp)$) and $(\top)$, and by adding the following 3 axioms:

$$(\perp') \ \frac{\phi \in \mathbf{L}^{\mathrm{cst}}_{\perp\top}}{\perp^{\epsilon(\phi)} \leq_{\mathrm{cst}} \phi} \ , \qquad (\not\perp') \ \frac{\phi \in \mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}}{\phi \leq_{\mathrm{cst}} \not\perp^{\epsilon(\phi)}} \ , \qquad (\top') \ \frac{\psi \in \mathbf{L}^{\mathrm{cst}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq_{\mathrm{cst}} \psi} \ ,$$

where $\mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}$ is a subset of $\mathbf{L}^{\mathrm{cst}}_{\not\perp}$ (defined by the clauses in Figure 5.10 to restrict the occurrences of conjunctions whose interpretation is empty) such that, for all $\phi \in \mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}$, we have that $[\![\phi]\!] \subseteq [\![\not\perp^{\epsilon(\phi)}]\!]$. The antecedent of the rule $(\not\perp')$ is $\phi \in \mathbf{L}^{\mathrm{cst}}_{\not\perp\mathrm{safe}}$ instead of the more "natural" (see rule $(\not\perp')$ in Section 4.4.1) $\phi \in \mathbf{L}^{\mathrm{cst}} - \mathbf{L}^{\mathrm{cst}}_{\perp\top}$ since, owing to the presence of

conjunctions whose interpretation is empty (see Remark 5.7), the rule with this antecedent would be unsound. For instance, for any $\rho, \sigma \in \mathbf{T}$, $[\![(\bot^\rho \wedge \not\bot^\rho) \to \not\bot^\sigma]\!] = [\![\top^{\rho\to\sigma}]\!] \not\subseteq [\![\not\bot^{\rho\to\sigma}]\!]$. On the other hand it is easy to check that, for all $\phi \in \mathbf{L}^{\mathrm{cst}}_{\not\bot\mathrm{safe}}$, we have that $[\![\phi]\!] \subseteq [\![\not\bot^{\epsilon(\phi)}]\!]$.

As for the entailment $\leq_{\mathrm{cs}}$ and $\leq_{\mathrm{ct}}$, the entailment relation $\leq_{\mathrm{cst}}$ is reflexive and transitive, since rule (Trans) in Fig. 3.4 is admissible. Let $\cong_{\mathrm{cst}}$ denote the equivalence relation induced by $\leq_{\mathrm{cst}}$. The soundness of $\leq_{\mathrm{cst}}$ (for both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$) can be proved by induction on the derivation of the entailment judgement .

**Theorem 5.52 (Soundness of $\leq_{\mathrm{cst}}$)** *Let $\xi_1, \xi_2 \in \mathbf{C}^{\mathrm{cst}}$. $\xi_1 \leq_{\mathrm{cst}} \xi_2$ implies $[\![\xi_1]\!] \subseteq [\![\xi_2]\!]$.* □

Also in this case the entailment rules of are not complete w.r.t. $\mathcal{M}^{\mathrm{lazy}}$. Both the arguments that show the failure of completeness for the entailment for strictness (see Example 5.11) and for the one for totality (see Example 5.36), in fact, can be applied to the combined entailment as well. However, $\leq_{\mathrm{cst}}$ is not complete even w.r.t. $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$. There are in fact other reasons for which completeness fails, due to the interaction of strictness and totality properties, as the following example shows.

**Example 5.53 (Incompleteness of $\leq_{\mathrm{cst}}$ w.r.t. $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$)** *As explained in Remark 5.7, in the combined language of properties we can define conjunctions whose interpretation is empty, like $\not\bot^{\mathsf{nat}} \wedge \bot^{\mathsf{nat}}$. For every $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{cst}}(\rho)$ and $\psi \in \mathbf{L}^{\mathrm{cst}}(\sigma)$ we have that $[\![\phi_1]\!] \cap [\![\phi_2]\!] = \emptyset$ implies $[\![\phi_1 \wedge \phi_2 \to \psi]\!] = [\![\top^{\rho\to\sigma}]\!]$. However, for $\psi \notin \mathbf{L}^{\mathrm{cst}}_{\top}$, we have that $\phi_1 \wedge \phi_2 \to \psi \not\cong_{\mathrm{cst}} \top^{\rho\to\sigma}$. For instance: $(\bot^\rho \wedge \not\bot^\rho) \to \not\bot^\sigma \not\cong_{\mathrm{cst}} \top^{\rho\to\sigma}$.*

Another reason for incompleteness is due to the fact that the entailment rules do not take into account the monotonicity of the evaluation rules, i.e. the arguments of Example 4.48 apply also to $\leq_{\mathrm{cst}}$.

### 5.4.2 A language of full conjunctive strictness and totality types

In this section, to be able to handle the weakness of the system $\leq_{\mathrm{cst}}$ shown in Example 4.48, we introduce a richer language of strictness and totality types in which the properties of monotonicity can be expressed. As we will see, the entailment relation for the new language is rather awkward.

As for the entailment $\leq'_{\mathrm{st}}$ of Section 4.4.2 we consider the language PCF′ (the subset of PCFP described in Section 2.5), which is also the language considered in [Solberg, 1995b] and [Solberg, 1995a] Chapter 3.

The new language and entailment relation are more expressive than those in *op. cit.* and correct some drawbacks of the original system (see Section 5.4.4).

The main tool to reason about monotonicity is the introduction of two cst-type constructors: $\downarrow$, and $\uparrow$, which are interpreted as the semantic operations of downward and upward closure (see Definition 4.49) respectively. Moreover, to capture any possible interaction between the new operators and the conjunction operator, we allow conjunctions to occur also in the right part of the arrow operator.

$$(a)\ \frac{a \in \{\bot, \not\bot, \top\} \quad \rho \in \mathbf{T}}{a^\rho \in \mathbf{L}^{\mathrm{fcst}}(\rho)} \qquad (\to)\ \frac{\nu_1 \in \mathbf{L}^{\mathrm{fcst}}(\rho_1) \quad \nu_2 \in \mathbf{L}^{\mathrm{fcst}}(\rho_2)}{\nu_1 \to \nu_2 \in \mathbf{L}^{\mathrm{fcst}}(\rho_1 \to \rho_2)}$$

$$(\wedge)\ \frac{\nu_1 \in \mathbf{L}^{\mathrm{fcst}}(\rho) \quad \nu_2 \in \mathbf{L}^{\mathrm{fcst}}(\rho)}{\nu_1 \wedge \nu_2 \in \mathbf{L}^{\mathrm{fcst}}(\rho)}$$

$$(\downarrow)\ \frac{\nu \in \mathbf{L}^{\mathrm{fcst}}(\rho)}{\downarrow(\nu) \in \mathbf{L}^{\mathrm{fcst}}(\rho)} \qquad (\uparrow)\ \frac{\nu \in \mathbf{L}^{\mathrm{fcst}}(\rho)}{\uparrow(\nu) \in \mathbf{L}^{\mathrm{fcst}}(\rho)}$$

Figure 5.11: Full conjunctive strictness and totality types

**Definition 5.54 (Full conjunctive strictness and totality types)** *The language of* full conjunctive strictness and totality types *(*fcst-types *for short)* $\mathbf{L}^{\mathrm{fcst}}$*, ranged over by* $\nu$ *(with superscripts and subscripts if needed) is defined by* $\mathbf{L}^{\mathrm{fcst}} = \bigcup_{\rho \in \mathbf{T}} \mathbf{L}^{\mathrm{fcst}}(\rho)$*, where the sets* $\mathbf{L}^{\mathrm{fcst}}(\rho)$ *are defined by the rules in Fig. 5.11.*

The semantics of fcst-types is specified by the clauses of Definition 5.2 and by the following clauses for the interpretation of the new operators,

$$[\![\downarrow(\nu)]\!]^{\mathcal{M}} = \hat{[\![\nu]\!]}^{\mathcal{M}} \qquad [\![\uparrow(\nu)]\!]^{\mathcal{M}} = \breve{[\![\nu]\!]}^{\mathcal{M}} \ ,$$

where $\hat{\cdot}$ and $\breve{\cdot}$ are respectively the semantic operations of downward and upward closure of Definition 4.49. All the results of this section are valid both for $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef,por}}$.

Let $\mathbf{L}^{\mathrm{fcst}}_\top, \mathbf{L}^{\mathrm{fcst}}_\bot$, and $\mathbf{L}^{\mathrm{fcst}}_{\not\bot}$ be the subsets of $\mathbf{L}^{\mathrm{fcst}}$ corresponding to the subsets $\mathbf{L}'^{\mathrm{st}}_\top, \mathbf{L}'^{\mathrm{st}}_\bot$ and $\mathbf{L}'^{\mathrm{st}}_{\not\bot}$ of $\mathbf{L}'^{\mathrm{st}}$ introduced at the beginning of Section 4.4.2, i.e. for $a \in \{\top, \bot, \not\bot\}$,

$$\mathbf{L}^{\mathrm{fcst}}_a = \{\nu_1 \to \cdots \to \nu_n \to a^\rho \mid n \geq 0, \nu_1, \ldots, \nu_n \in \mathbf{L}^{\mathrm{fcst}} \text{ and } \rho \in \mathbf{T}\}.$$

Note that, owing to the presence of the new operators ($\downarrow$ and $\uparrow$) and to the fact that conjunction is allowed also at the top level and in the right part of the arrow operator, $\mathbf{L}^{\mathrm{fcst}}_\top \cup \mathbf{L}^{\mathrm{fcst}}_\bot \cup \mathbf{L}^{\mathrm{fcst}}_{\not\bot} \neq \mathbf{L}^{\mathrm{fcst}}$. For instance $\downarrow(\nu) \notin \mathbf{L}^{\mathrm{fcst}}_\top \cup \mathbf{L}^{\mathrm{fcst}}_\bot \cup \mathbf{L}^{\mathrm{fcst}}_{\not\bot}$, for every fcst-type $\nu$.

**Lemma 5.55** *1.* $\nu \in \mathbf{L}^{\mathrm{fcst}}_\top$ *implies* $[\![\nu]\!] = [\![\downarrow(\nu)]\!] = [\![\uparrow(\nu)]\!] = [\![\top^{\epsilon(\nu)}]\!] = \mathbf{I}(\epsilon(\nu))$*,*

*2.* $\nu \in \mathbf{L}^{\mathrm{fcst}}_\bot$ *implies* $[\![\nu]\!] = [\![\downarrow(\nu)]\!]$*, and*

*3.* $\nu \in \mathbf{L}^{\mathrm{fcst}}_{\not\bot}$ *implies* $[\![\nu]\!] = [\![\uparrow(\nu)]\!]$*.*

**Proof.**

1. Obvious, since $\nu \in \mathbf{L}^{\mathrm{fcst}}_\top$ implies $[\![\nu]\!] = \mathbf{I}(\epsilon(\nu))$.

2. Let $\nu = \nu_1 \to \cdots \to \nu_n \to \bot^\rho \in \mathbf{L}^{\mathrm{fcst}}_\bot$ $(n \geq 0)$. The case $[\![\nu]\!] = \mathbf{I}(\epsilon(\nu))$ is trivial, so let, for all $i \in \{1, \ldots, n\}$, $[\![\nu_i]\!] \neq \emptyset$.
   Assume that $M' \preceq^{\mathrm{lazy}}_{\mathrm{obs}} M$ for some $[M] \in [\![\nu]\!]$ and $[M'] \notin [\![\nu]\!]$. Then there are $[N_i] \in [\![\nu_i]\!]$, $1 \leq i \leq n$, such that $[M'\, N_1 \cdots N_n] \notin [\![\bot^\rho]\!]$. However this implies that $M'\, N_1 \cdots N_n \Downarrow$ whereas $M\, N_1 \cdots N_n \Uparrow$, contradicting the fact that $M' \preceq^{\mathrm{lazy}}_{\mathrm{obs}} M$.

3. Similar to the proof of 2. □

Let $\mathbf{L}_{\not\perp}^{\mathrm{fcst}^\bullet}$ be the subset of $\mathbf{L}_{\not\perp}^{\mathrm{fcst}}$ defined by $\mathbf{L}_{\not\perp}^{\mathrm{fcst}^\bullet} = \{\not\perp^\rho \mid \rho \in \mathbf{T}\} \cup \mathbf{L}_{\not\perp}^{\mathrm{fcst}^\rightarrow}$, where $\mathbf{L}_{\not\perp}^{\mathrm{fcst}^\rightarrow} = \{\nu_1 \to \cdots \to \nu_n \to \not\perp^{\rho\to\sigma} \mid \ \rho \to \sigma \in \mathbf{T}$, $n \geq 0$, and $\nu_1, \ldots, \nu_n \in \mathbf{L}^{\mathrm{fcst}}\}$.

**Lemma 5.56** *1.* $\nu \in \mathbf{L}_{\not\perp}^{\mathrm{fcst}^\bullet}$ *implies* $[\![\downarrow(\nu)]\!] = [\![\top^\rho]\!]$

*2.* $\nu \in \mathbf{L}_{\perp}^{\mathrm{fcst}}$ *implies* $[\![\uparrow(\nu)]\!] = [\![\top^\rho]\!]$.

**Proof.**

1. By definition, $\mathbf{L}_{\not\perp}^{\mathrm{fcst}^\bullet} = \{\not\perp^\rho \mid \rho \in \mathbf{T}\} \cup \mathbf{L}_{\not\perp}^{\mathrm{fcst}^\rightarrow}$, where $\mathbf{L}_{\not\perp}^{\mathrm{fcst}^\rightarrow} = \{\nu_1 \to \cdots \to \nu_n \to \not\perp^{\rho\to\sigma} \mid \ \rho \to \sigma \in \mathbf{T}$, $n \geq 0$, and $\nu_1, \ldots, \nu_n \in \mathbf{L}^{\mathrm{fcst}}\}$.
   Let $\nu \in \mathbf{L}_{\not\perp}^{\mathrm{fcst}^\rightarrow}$, i.e. $\nu = \nu_1 \to \cdots \to \nu_n \to \not\perp^{\sigma\to\tau}$. For all $[M] \in [\![\top^{\epsilon(\nu)}]\!]$, consider $N = \lambda\, x_1^{\epsilon(\nu_1)}. \cdots \lambda\, x_n^{\epsilon(\nu_n)}. \lambda\, y^\sigma. M\, x_1 \cdots x_n\, y$. It is easy to check that $M \preceq_{\mathrm{obs}}^{\mathrm{lazy}} N$ and $[N] \in [\![\nu]\!]$. Therefore $[\![\downarrow(\nu)]\!] = [\![\hat{\nu}]\!] = [\![\top^{\epsilon(\nu)}]\!]$.
   The case $\nu \in \{\not\perp^\rho \mid \rho \in \mathbf{T}\} - \mathbf{L}_{\not\perp}^{\mathrm{fcst}^\rightarrow}$, i.e. $\nu = \not\perp^{\mathsf{nat}}$ or $\nu = \not\perp^{\mathsf{bool}}$, is trivial.

2. The result follows from the fact that, for $\nu \in \mathbf{L}_{\perp}^{\mathrm{fcst}}$, $[\mathsf{fix}\, x^{\epsilon(\nu)}.x] \in [\![\nu]\!]$ and, for all $[M] \in [\![\top^{\epsilon(\nu)}]\!]$, $\mathsf{fix}\, x^{\epsilon(\nu)}.x \preceq_{\mathrm{obs}}^{\mathrm{lazy}} M$. □

Note that Lemma 5.56.1 cannot be extended to the fcst-types in $\mathbf{L}_{\not\perp}^{\mathrm{fcst}} - \mathbf{L}_{\not\perp}^{\mathrm{fcst}^\bullet}$, i.e. it is not true that, for all $\nu \in \mathbf{L}_{\not\perp}^{\mathrm{fcst}} - \mathbf{L}_{\not\perp}^{\mathrm{fcst}^\bullet}$, $[\![\downarrow(\nu)]\!] = [\![\top^{\epsilon(\nu)}]\!]$. For instance consider the fcst-type $\nu = \top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}$ ($\nu \in \mathbf{L}_{\not\perp}^{\mathrm{fcst}} - \mathbf{L}_{\not\perp}^{\mathrm{fcst}^\bullet}$), which characterizes the constant functions of type $\mathsf{nat} \to \mathsf{nat}$. We have $[\![\top^{\epsilon(\nu)}]\!] = \mathbf{I}(\mathsf{nat} \to \mathsf{nat})$, whereas $[\![\downarrow(\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}})]\!]$ is the set of the $\simeq_{\mathrm{obs}}^{\mathrm{lazy}}$-classes of the closed terms which are $\preceq_{\mathrm{obs}}^{\mathrm{lazy}}$ than a constant function. It is easy to see that, for all the fcst-type $\nu' \in \mathbf{L}_{\top}^{\mathrm{fcst}} \cup \mathbf{L}_{\perp}^{\mathrm{fcst}} \cup \mathbf{L}_{\not\perp}^{\mathrm{fcst}}$, $[\![\downarrow(\nu)]\!] = [\![\hat{\nu}]\!] \neq [\![\nu']\!]$.

**Lemma 5.57** *1.* $[\![\downarrow(\nu_1 \to \nu_2)]\!] \subseteq [\![\nu_1 \to \downarrow(\nu_2)]\!]$

*2.* $[\![\uparrow(\nu_1 \to \nu_2)]\!] \subseteq [\![\nu_1 \to \uparrow(\nu_2)]\!]$ .

**Proof.**

1. The case $[\![\nu_1 \to \nu_2]\!] = \mathbf{I}(\epsilon(\nu_1 \to \nu_2))$ is trivial, so assume $[\![\nu_1]\!] \neq \emptyset$ and $[\![\nu_2]\!] \neq \mathbf{I}(\epsilon(\nu_2))$. For every $[M] \in [\![\downarrow(\nu_1 \to \nu_2)]\!]$ there is $[M'] \in [\![\nu_1 \to \nu_2]\!]$ such that $M \preceq_{\mathrm{obs}}^{\mathrm{lazy}} M'$. So, for every $[N] \in [\![\nu_1]\!]$, $MN \preceq_{\mathrm{obs}}^{\mathrm{lazy}} M'N$. But $[MN] \in [\![\nu_2]\!]$, which implies $[M'N] \in [\![\downarrow(\nu_2)]\!]$.

2. The case $[\![\nu_1 \to \nu_2]\!] = \mathbf{I}(\epsilon(\nu_1 \to \nu_2))$ is trivial, so assume $[\![\nu_1]\!] \neq \emptyset$ and $[\![\nu_2]\!] \neq \mathbf{I}(\epsilon(\nu_2))$. For every $[M] \in [\![\uparrow(\nu_1 \to \nu_2)]\!]$ there is $[M'] \in [\![\nu_1 \to \nu_2]\!]$ such that $M' \preceq_{\mathrm{obs}}^{\mathrm{lazy}} M$. So, for every $[N] \in [\![\nu_1]\!]$, $M'N \preceq_{\mathrm{obs}}^{\mathrm{lazy}} MN$. But $[M'N] \in [\![\nu_2]\!]$, which implies $[MN] \in [\![\uparrow(\nu_2)]\!]$. □

We now introduce an entailment relation between fcst-types, denoted by $\leq_{\mathrm{fcst}}$.

$$(\bot)\ \frac{\nu \in \mathbf{L}_{\bot}^{\mathrm{fcst}} \cup \mathbf{L}_{\top}^{\mathrm{fcst}}}{\bot^{\epsilon(\nu)} \leq \nu} \qquad (\not\downarrow)\ \frac{\nu \in \mathbf{L}_{\not\downarrow\mathrm{safe}}^{\mathrm{fcst}}}{\nu \leq \not\downarrow^{\epsilon(\nu)}} \qquad (\top)\ \frac{\nu_2 \in \mathbf{L}_{\top}^{\mathrm{cst}} \quad \epsilon(\nu_1) = \epsilon(\nu_2)}{\nu_1 \leq \nu_2}$$

$$(\mathrm{Ref})\ \nu \leq \nu \qquad (\mathrm{Trans})\ \frac{\nu_1 \leq \nu_2 \quad \nu_2 \leq \nu_3}{\nu_1 \leq \nu_3}$$

$$(\rightarrow)\ \frac{\nu_1' \leq \nu_1 \quad \nu_2 \leq \nu_2'}{\nu_1 \rightarrow \nu_2 \leq \nu_1' \rightarrow \nu_2'}$$

$$(\wedge_1)\ \nu_1 \wedge \nu_2 \leq \nu_1 \qquad (\wedge_2)\ \nu_1 \wedge \nu_2 \leq \nu_2 \qquad (\wedge_3)\ \frac{\nu \leq \nu_1 \quad \nu \leq \nu_2}{\nu \leq \nu_1 \wedge \nu_2}$$

$$(\rightarrow \wedge)\ (\nu \rightarrow \nu_1) \wedge (\nu \rightarrow \nu_2) \leq \nu \rightarrow (\nu_1 \wedge \nu_2)$$

$$(\mathrm{Mon}_1)\ \nu_1 \rightarrow \nu_2 \leq\downarrow (\nu_1) \rightarrow\downarrow (\nu_2) \qquad (\mathrm{Mon}_2)\ \nu_1 \rightarrow \nu_2 \leq\uparrow (\nu_1) \rightarrow\uparrow (\nu_2)$$

$$(\downarrow_1)\ \nu \leq\downarrow (\nu) \qquad (\downarrow_2)\ \frac{\nu_1 \leq \nu_2}{\downarrow (\nu_1) \leq\downarrow (\nu_2)} \qquad (\downarrow_3)\ \frac{\nu \in \mathbf{L}_{\not\downarrow}^{\mathrm{fcst}^{\bullet}}}{\top^{\epsilon(\nu)} \leq\downarrow (\nu)}$$

$$(\downarrow_4)\ \frac{\nu \in \mathbf{L}_{\bot}^{\mathrm{fcst}}}{\downarrow (\nu) \leq \nu} \qquad (\downarrow_5)\ \downarrow (\downarrow (\nu)) \leq\downarrow (\nu) \qquad (\downarrow_6)\ \downarrow (\nu_1 \wedge \nu_2) \leq\downarrow (\nu_1) \wedge \downarrow (\nu_2)$$

$$(\downarrow_7)\ \downarrow (\nu_1 \rightarrow \nu_2) \leq \nu_1 \rightarrow\downarrow (\nu_2)$$

$$(\uparrow_1)\ \nu \leq\uparrow (\nu) \qquad (\uparrow_2)\ \frac{\nu_1 \leq \nu_2}{\uparrow (\nu_1) \leq\uparrow (\nu_2)} \qquad (\uparrow_3)\ \frac{\nu \in \mathbf{L}_{\not\downarrow}^{\mathrm{fcst}}}{\uparrow (\nu) \leq \nu}$$

$$(\uparrow_4)\ \frac{\nu \in \mathbf{L}_{\bot}^{\mathrm{fcst}}}{\top^{\epsilon(\nu)} \leq\uparrow (\nu)} \qquad (\uparrow_5)\ \uparrow (\uparrow (\nu)) \leq\uparrow (\nu) \qquad (\uparrow_6)\ \uparrow (\nu_1 \wedge \nu_2) \leq\uparrow (\nu_1) \wedge \uparrow (\nu_2)$$

$$(\uparrow_7)\ \uparrow (\nu_1 \rightarrow \nu_2) \leq \nu_1 \rightarrow\uparrow (\nu_2)$$

$$(\downarrow\uparrow_1)\ \downarrow (\uparrow (\nu)) \leq\uparrow (\downarrow (\nu)) \qquad (\downarrow\uparrow_2)\ \uparrow (\downarrow (\nu)) \leq\downarrow (\uparrow (\nu))$$

Figure 5.12: Entailment rules for fcst-types (system $\leq_{\mathrm{fcst}}$)

**Definition 5.58 (Entailment relation for full strictness and totality types)** *Let* $\nu_1, \nu_2 \in \mathbf{L}^{\mathrm{fcst}}$. *We write* $\nu_1 \leq_{\mathrm{fcst}} \nu_2$ *to mean that* $\nu_1 \leq \nu_2$ *is derivable by the rules in Fig. 5.12. By* $\cong_{\mathrm{fcst}}$ *we denote the equivalence relation induced by* $\leq_{\mathrm{fcst}}$.

The set $\mathbf{L}^{\mathrm{fcst}}_{\not\downarrow\mathrm{safe}}$ in the premise of rule $(\not\downarrow)$ corresponds to $\mathbf{L}^{\mathrm{cst}}_{\not\downarrow\mathrm{safe}}$ introduced in Section 5.4.1. $\mathbf{L}^{\mathrm{fcst}}_{\not\downarrow\mathrm{safe}}$ is the subset of $\mathbf{L}^{\mathrm{fcst}}_{\not\downarrow}$ defined by:

- $\nu \in \mathbf{L}^{\mathrm{fcst}}_{\not\downarrow\mathrm{safe}}$, if $\xi \in \mathbf{L}^{\mathrm{fcst}}_{\not\downarrow}$ and does not contain $\bot$ constructors, and
- $\nu_1 \wedge \cdots \wedge \nu_n \to \nu \in \mathbf{L}^{\mathrm{fcst}}_{\not\downarrow\mathrm{safe}}$, if $\nu \in \mathbf{L}^{\mathrm{fcst}}_{\not\downarrow\mathrm{safe}}$, $n \geq 1$, and either $\nu_1, \ldots, \nu_n \in \mathbf{L}^{\mathrm{fcst}}_{\not\downarrow} \cup \mathbf{L}^{\mathrm{fcst}}_{\top}$ or $\nu_1, \ldots, \nu_n \in \mathbf{L}^{\mathrm{fcst}}_{\bot} \cup \mathbf{L}^{\mathrm{fcst}}_{\top}$.

It is easy to see that $\mathbf{L}^{\mathrm{fcst}}_{\not\downarrow\mathrm{safe}}$ contains only fcst-types $\nu$ such that $[\![\nu]\!] \subseteq [\![\not\downarrow^{\epsilon(\nu)}]\!]$.

**Example 5.59** *Using* $\leq_{\mathrm{fcst}}$, *the the weakness of the system* $\leq_{\mathrm{cst}}$ *shown in Example 4.48 can be correctly handled. In fact*

- $\not\downarrow^{\rho} \to \bot^{\sigma} \leq_{\mathrm{fcst}} \downarrow(\not\downarrow^{\rho}) \to \downarrow(\bot^{\sigma})$ *(by* $(\mathrm{Mon}_1)$*), and* $\downarrow(\not\downarrow^{\rho}) \to \downarrow(\bot^{\sigma}) \leq_{\mathrm{fcst}} \top^{\rho} \to \not\downarrow^{\sigma}$ *(by* $(\downarrow_3)$ *and* $(\downarrow_4)$*).*
- *Similarly* $\bot^{\rho} \to \not\downarrow^{\sigma} \leq_{\mathrm{fcst}} \top^{\rho} \to \not\downarrow^{\rho}$.

The soundness of the $\leq_{\mathrm{fcst}}$ relation (w.r.t. both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$) is stated by the following theorem.

**Theorem 5.60 (Soundness of $\leq_{\mathrm{fcst}}$)** *Let* $\nu_1, \nu_2 \in \mathbf{L}^{\mathrm{fcst}}$. $\nu_1 \leq_{\mathrm{fcst}} \nu_2$ *implies* $[\![\nu_1]\!] \subseteq [\![\nu_2]\!]$.

**Proof.** By induction on the derivation of $\nu_1 \leq_{\mathrm{fcst}} \nu_2$. Rules $(\downarrow_3)$ and $(\uparrow_3)$ are by Lemma 5.56. Rules $(\downarrow_4)$ and $(\uparrow_4)$ are by Lemma 5.55. Rules $(\downarrow_7)$ and $(\uparrow_7)$ are by Lemma 5.57. The only interesting remaining rules are the $(\mathrm{Mon}_1)$ and $(\mathrm{Mon}_2)$.

Consider $(\mathrm{Mon}_1)$. We have to show that $[\![\nu_1 \to \nu_2]\!] \subseteq [\![\downarrow(\nu_1) \to \downarrow(\nu_2)]\!]$.
For $[\![\nu_1 \to \nu_2]\!] = \mathbf{I}(\epsilon(\nu_1 \to \nu_2))$ it is immediate, since either $[\![\nu_1]\!] = \emptyset$ or $[\![\nu_2]\!] = [\![\top^{\epsilon(\psi_2)}]\!]$. So assume $[\![\nu_1]\!] \neq \emptyset$ and $[\![\nu_2]\!] \neq [\![\top^{\epsilon(\psi_2)}]\!]$.
Let $[F] \in [\![\nu_1 \to \nu_2]\!]$ and $[Q] \in [\![\nu_1]\!]$, we have $[F\,Q] \in [\![\nu_2]\!]$. But, for each $[N] \in [\![\downarrow(\nu_1)]\!]$ there exists $[N'] \in [\![\nu_1]\!]$ such that $N \preceq^{\mathrm{lazy}}_{\mathrm{obs}} N'$, so $[FN'] \in [\![\nu_2]\!]$. Then, for monotonicity, $FN \preceq^{\mathrm{lazy}}_{\mathrm{obs}} FN'$, i.e. $[FN] \in [\![\downarrow(\nu_2)]\!]$. Therefore $[F] \in [\![\downarrow(\nu_1) \to \downarrow(\nu_2)]\!]$.

For $(\mathrm{Mon}_2)$, we have to show that $[\![\nu_1 \to \nu_2]\!] \subseteq [\![\uparrow(\nu_1) \to \uparrow(\nu_2)]\!]$.
For $[\![\nu_1 \to \nu_2]\!] = \mathbf{I}(\epsilon(\nu_1 \to \nu_2))$ it is immediate, since either $[\![\nu_1]\!] = \emptyset$ or $[\![\nu_2]\!] = [\![\top^{\epsilon(\psi_2)}]\!]$. So assume $[\![\nu_2]\!] \neq \emptyset$ and $[\![\nu_2]\!] \neq [\![\top^{\epsilon(\psi_2)}]\!]$.
Let $[F] \in [\![\nu_1 \to \nu_2]\!]$ and $[Q] \in [\![\nu_1]\!]$, we have $[F\,Q] \in [\![\nu_2]\!]$. But, for each $[N] \in [\![\uparrow(\nu_1)]\!]$ there is $[N'] \in [\![\nu_1]\!]$ such that $N' \preceq^{\mathrm{lazy}}_{\mathrm{obs}} N$, so $[FN'] \in [\![\nu_2]\!]$. For monotonicity $FN' \preceq^{\mathrm{lazy}}_{\mathrm{obs}} FN$, i.e. $[FN] \in [\![\uparrow(\nu_2)]\!]$. Therefore $[F] \in [\![\uparrow(\nu_1) \to \uparrow(\nu_2)]\!]$. □

**Remark 5.61** *The entailment rules of Definition 5.58 are not yet complete w.r.t. the interpretation in* $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$, *in fact the argument of Example 5.53 apply also for the* $\leq_{\mathrm{fcst}}$ *entailment. At the moment we do not know whether a sound (and decidable) extension of* $\leq_{\mathrm{fcst}}$ *which is complete w.r.t.* $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$ *(or* $\mathcal{M}^{\mathrm{lazy}}$*) exists.*

### 5.4.3 An assignment system for full conjunctive strictness and totality

For each PCF′ constant $\mathsf{c}$ a set $\mathbf{L}^{\mathrm{fcst}}(\mathsf{c})$ of minimal fcst-types in $\mathbf{L}'^{\mathrm{s}} \cup \mathbf{L}'^{\mathrm{t}}$ is specified. The idea is that every fcst-type which is entailed ($\leq_{\mathrm{fcst}}$) by an element of $\mathbf{L}^{\mathrm{fcst}}(\mathsf{c})$ can be assigned to $\mathsf{c}$. For example, for all integers $\mathsf{n}$, $\mathbf{L}^{\mathrm{fcst}}(\mathsf{n}) = \{\not\perp^{\mathsf{nat}}\}$ and $\mathbf{L}^{\mathrm{fcst}}(+) = \{\perp^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \perp^{\mathsf{nat}}, \top^{\mathsf{nat}} \to \perp^{\mathsf{nat}} \to \perp^{\mathsf{nat}}, \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}\}$. The assumption that, for all constants $\mathsf{c}$ of the language PCF′, $\mathbf{L}^{\mathrm{fcst}}(\mathsf{c}) \subseteq \mathbf{L}'^{\mathrm{s}} \cup \mathbf{L}'^{\mathrm{t}}$ is not restrictive, since the argument used in Section 4.4.3 for $\vdash'_{\mathrm{st}}$ apply also to $\vdash_{\mathrm{fcst}}$.

Before introducing the fcst-type assignment system we need to introduce a set $\mathbf{L}^{\mathrm{fcst}}_{\mathsf{fix}_\perp\mathrm{safe}}$ of fcst-types such that, for every $\nu \in \mathbf{L}^{\mathrm{fcst}}_{\mathsf{fix}_\perp\mathrm{safe}}$, $[\mathsf{fix}\, x^{\epsilon(\nu)}.x] \in [\![\nu]\!]$. This set is used to formulate a sound assignment rule for fixpoint expressions. In fact the rule obtained from rule ($\mathrm{Fix}_\perp$) in Fig. 5.14 by removing the side condition $\nu \in \mathbf{L}^{\mathrm{fcst}}_{\mathsf{fix}_\perp\mathrm{safe}}$ is not sound, since it allows derivations like

$$(\mathrm{Fix}_\perp)\ \frac{(\mathrm{Var})\ \Sigma, x : \not\perp^\rho \vdash x : \not\perp^\rho}{\Sigma \vdash \mathsf{fix}\, x.M : \not\perp^\rho}\ ,$$

and (more in general)

$$(\mathrm{Fix}_\perp)\ \frac{(\mathrm{Var})\ \Sigma, x : \nu \vdash x : \nu}{\Sigma \vdash \mathsf{fix}\, x.M : \nu}$$

for some $\nu$ such that $[\mathsf{fix}\, x^{\epsilon(\nu)}.M] \notin [\![\nu]\!]$, which are clearly not sound.

Let $\mathbf{L}^{\mathrm{fcst}}_{\mathsf{fix}_\perp\mathrm{safe}}$ be defined by

$$\mathbf{L}^{\mathrm{fcst}}_{\mathsf{fix}_\perp\mathrm{safe}} = \{\nu \mid \nu \in \mathbf{L}^{\mathrm{fcst}} \text{ and } \mathrm{BOT}(\nu)\},$$

where the predicate BOT[3], defined by the clauses in Fig. 5.13, satisfies the following property.

**Proposition 5.62** *For every $\nu \in \mathbf{L}^{\mathrm{fcst}}$, $\mathrm{BOT}(\nu)$ implies $[\mathsf{fix}\, x^{\epsilon(\nu)}.x] \in [\![\nu]\!]$.* □

Note that the reverse implication does not hold.

**Definition 5.63 (Full conjunctive strictness and totality type assignment system)** *We write $\Sigma \vdash_{\mathrm{fcst}} M : \nu$ to mean that $\Sigma \vdash M : \nu$ can be derived by the rules in Fig. 5.14.*

Note that in $\vdash_{\mathrm{fcst}}$ there are no introduction and elimination rules for $\downarrow$ and $\uparrow$ since they are implicit in the $\leq_{\mathrm{fcst}}$ relation. The proof of the following fact is immediate.

**Fact 5.64** *1. $\Sigma \vdash_{\mathrm{fcst}} M : \phi$ implies $\vdash_{\mathbf{T}} M : \epsilon(\phi)$ and $\epsilon(\Sigma) \supseteq \mathrm{FV}(M)$.*

*2. $\vdash_{\mathbf{T}} M : \rho$ implies, for all basis $\Sigma$ such that $\Sigma = \{x^{\epsilon(\nu)} : \nu \mid x^{\epsilon(\nu)} \in \mathrm{FV}(M)$ for some $\nu \in \mathbf{L}^{\mathrm{fcst}}\}$, $\Sigma \vdash_{\mathrm{fcst}} M : \top^\rho$.*

The soundness of the $\vdash_{\mathrm{fcst}}$ assignment system (w.r.t. both $\mathcal{M}^{\mathrm{lazy}}$ and $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$) is stated by the following theorem.

[3]This predicate BOT is the extension to our fcst-types of the predicate BOT introduced in [Solberg, 1995b] page 507 for the system $\vdash_{\mathrm{CST}}$ (see Section 5.4.4 of this thesis).

$$
\begin{array}{lcl}
\mathrm{BOT}(\bot^{\rho}) & = & \textbf{TRUE} \\
\mathrm{BOT}(\not\bot^{\rho}) & = & \textbf{FALSE} \\
\mathrm{BOT}(\top^{\rho}) & = & \textbf{TRUE} \\
\mathrm{BOT}(\downarrow \nu) & = & \mathrm{BOT}(\nu) \\
\mathrm{BOT}(\uparrow \nu) & = & \mathrm{BOT}(\nu) \\
\mathrm{BOT}(\nu_1 \to \nu_2) & = & \mathrm{BOT}(\nu_2) \\
\mathrm{BOT}(\nu_1 \wedge \nu_2) & = & \mathrm{BOT}(\nu_1)\ \textbf{AND}\ \mathrm{BOT}(\nu_2)
\end{array}
$$

Figure 5.13: Predicate BOT

**Theorem 5.65 (Soundness of $\vdash_{\mathrm{fcst}}$)** *Let $\Sigma \vdash_{\mathrm{fcst}} M : \nu$. Then $[\![M]\!]_e \in [\![\nu]\!]$ for all $e$ such that for all $x : \nu' \in \Sigma$, $e(x) \in [\![\nu']\!]$.*

**Proof.** By induction on the derivation of $\Sigma \vdash_{\mathrm{fcst}} M : \nu$. □

### 5.4.4 Comparison with the system in Chapter 3 of Solberg's PhD thesis

In the language of properties introduced in Definition 5.54 (fcst-types), beside the syntactical difference of using $\bot$ instead of **b**, and $\not\bot$ instead of **n**, there is an additional constructor, $\uparrow$, (dual to $\downarrow$), which is not considered in [Solberg, 1995a] Chapter 3 and [Solberg, 1995b].

In *op. cit.* the soundness of the conjunctive strictness and totality analysis is studied w.r.t. a denotational semantics, since (as explained in [Solberg, 1995a] Section 3.3) it is not quite clear how to extend the construction of [Solberg, 1995a] Chapter 2 (see discussion at the beginning of Section 4.4.4 of this thesis) to the language of conjunctive strictness and totality types.

Instead, our operational approach applies quite well.

#### Entailment rules

Comparing our entailment, $\leq_{\mathrm{fcst}}$ (see Fig. 5.12), with the entailment $\leq_{\mathrm{CST}}$ in [Solberg, 1995b] page 506, we can see that our rule ($\top$) correspond to the rules (top1) and (top2) for $\leq_{\mathrm{CST}}$, while rules ($\bot$) and ($\not\bot$) correspond respectively to the rules (bot) and (notbot) for $\leq_{\mathrm{CST}}$. In particular, rule ($\not\bot$) is an extension of rule (notbot), for instance $\bot^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}} \leq_{\mathrm{cst}} \not\bot^{\mathsf{nat}\to\mathsf{nat}}$, but not $\mathbf{b}^{\mathsf{nat}} \to \mathbf{n}^{\mathsf{nat}} \leq_{\mathrm{CST}} \mathbf{n}^{\mathsf{nat}\to\mathsf{nat}}$.

$$(\leq)\ \frac{\Sigma \vdash M : \nu_1 \quad \nu_1 \leq \nu_2}{\Sigma \vdash M : \nu_2}$$

(Var) $\Sigma, x : \nu \vdash x : \nu$ (Con) $\frac{}{\Sigma \vdash \mathsf{c} : \nu}\ \nu \in \mathbf{L}^{\mathrm{fcst}}(\mathsf{c})$

$(\to \mathrm{I}_{\not\downarrow})\ \frac{}{\Sigma \vdash \lambda x.M : \not\downarrow^{\rho}} \vdash_{\mathbf{T}} \lambda x.M : \rho$ $(\to \mathrm{I})\ \frac{\Sigma, x : \nu_1 \vdash M : \nu_2}{\Sigma \vdash \lambda x.M : \nu_1 \to \nu_2}$

$$(\to \mathrm{E})\ \frac{\Sigma \vdash M : \nu_1 \to \nu_2 \quad \Sigma \vdash N : \nu_1}{\Sigma \vdash MN : \nu_2}$$

$$(\mathrm{Fix}_{\perp})\ \frac{\Sigma, x : \nu \vdash M : \nu}{\Sigma \vdash \mathsf{fix}\, x.M : \nu}\ \nu \in \mathbf{L}^{\mathrm{fcst}}_{\mathsf{fix}_{\perp}\mathrm{safe}}$$

$$(\mathrm{Fix}_{\not\downarrow})\ \frac{\Sigma, x : \top^{\rho} \vdash M : \nu_1 \quad \Sigma, x : \nu_1 \vdash M : \nu_2 \quad \cdots \quad \Sigma, x : \nu_{n-1} \vdash M : \nu_n}{\Sigma \vdash \mathsf{fix}\, x.M : \nu_n}\ n \geq 1$$

$$(\mathrm{If}_{\perp})\ \frac{\Sigma \vdash N : \perp^{\mathsf{bool}} \quad \vdash_{\mathbf{T}} M_1 : \rho \quad \vdash_{\mathbf{T}} M_2 : \rho}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \perp^{\rho}}$$

$$(\mathrm{If}_{\not\downarrow})\ \frac{\Sigma \vdash N : \not\downarrow^{\mathsf{bool}} \quad \Sigma \vdash M_1 : \nu \quad \Sigma \vdash M_2 : \nu}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \nu}$$

$$(\mathrm{If}_{\top})\ \frac{\Sigma \vdash N : \top^{\mathsf{bool}} \quad \Sigma \vdash M_1 : \nu \quad \Sigma \vdash M_2 : \nu}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \downarrow(\nu)}$$

$$(\wedge \mathrm{I})\ \frac{\Sigma \vdash M : \nu_1 \quad \Sigma \vdash M : \nu_2}{\Sigma \vdash M : \nu_1 \wedge \nu_2}$$

Figure 5.14: Rules for fcst-type assignment (system $\vdash_{\mathrm{fcst}}$)

Our rules $(\wedge_1)$, $(\wedge_2)$, $(\wedge_3)$ and $(\rightarrow \wedge)$ correspond respectively to the rules $(\wedge 1)$, $(\wedge 2)$, $(\wedge)$ and $(\rightarrow \wedge)$ of $\leq_{\mathrm{CST}}$, while rules $(\downarrow_1)$, $(\downarrow_2)$, $(\downarrow_5)$, and $(\downarrow_6)$ corresponds respectively to the rules $(\downarrow 1)$, $(\downarrow 2)$, $(\downarrow 5)$, and $(\downarrow 6)$ of $\leq_{\mathrm{CST}}$.

Rules $(\downarrow_3)$ and $(\downarrow_4)$ are extensions of the corresponding rules $(\downarrow 3)$ and $(\downarrow 4)$ of $\leq_{\mathrm{CST}}$.

Notice that the rule

$$(\downarrow 7) \quad \downarrow(\phi \rightarrow \psi) \cong_{\mathrm{CST}} \phi \rightarrow \downarrow(\psi)$$

in [Solberg, 1995b] page 506 is not sound. In fact it allows to infer that $\downarrow(\top^{\mathsf{nat}} \rightarrow \not\perp^{\mathsf{nat}})$ is equivalent to $\top^{\mathsf{nat}} \rightarrow \top^{\mathsf{nat}}$, which is clearly false, since the latter characterizes all the programs of type $\mathsf{nat} \rightarrow \mathsf{nat}$ while the former characterizes the class of the programs which are $\preceq^{\mathrm{lazy}}_{\mathrm{obs}}$ than a program which does not use its argument (like $\lambda z^{\mathsf{nat}}.3$).

As suggested by Kirsten Solberg ([Solberg, 1996]) the rule $(\downarrow 7)$ must be weakened to

$$(\downarrow 7') \quad \downarrow(\phi \rightarrow \psi) \leq_{\mathrm{CST}} \phi \rightarrow \downarrow(\psi) \ .$$

Our rule $(\downarrow_7)$ is the rule $(\downarrow 7')$ above. Rules $(\uparrow_1)$, ..., $(\uparrow_7)$, $(\downarrow\uparrow_1)$, and $(\downarrow\uparrow_2)$ are new.

Finally, our rule $(\mathrm{Mon}_1)$ corresponds to the rule (monotone) of $\leq_{\mathrm{CST}}$, while the rule $(\mathrm{Mon}_2)$ is the extension to full conjunctive strictness and totality types of the corresponding rule for st-types without conjunction of the entailment $\leq'_{\mathrm{st}}$ in Section 4.4.2.

So we can conclude that our entailment relation $\leq_{\mathrm{fcst}}$ is an extension of the sound fragment of $\leq_{\mathrm{CST}}$.

### Assignment rules

Comparing our assignment system $\vdash_{\mathrm{fcst}}$ with the assignment system $\vdash_{\mathrm{CST}}$ in [Solberg, 1995b] page 507, we have that the rule $(\mathrm{If}_\top)$ in Fig. 5.14 is more powerful than the corresponding rule of $\vdash_{\mathrm{CST}}$, that follows:

$$(\mathrm{If3}) \ \frac{\Sigma \vdash N : \top^{\mathsf{bool}} \quad \Sigma \vdash M_1 : \nu \quad \Sigma \vdash M_2 : \nu}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \nu} \ \mathrm{BOT}(\nu).$$

For instance, for every closed term $M$,

$$\emptyset \ \vdash_{\mathrm{fcst}} \ \mathsf{if}\ M\ \mathsf{then}\ \lambda x^{\mathsf{nat}}.x\ \mathsf{else}\ \lambda y^{\mathsf{nat}}.\mathsf{succ}(y) \ \ : \downarrow(\not\perp^{\mathsf{nat}} \rightarrow \not\perp^{\mathsf{nat}}) \ ,$$

which cannot be proved in $\vdash_{\mathrm{CST}}$. We can also prove that replacing the two rules $(\mathrm{Fix}_\perp)$ and $(\mathrm{Fix}_{\not\perp})$ of $\vdash_{\mathrm{fcst}}$ by the following rule (fix) [4] of $\vdash_{\mathrm{CST}}$:

$$(\mathrm{fix}) \ \frac{\Sigma, x : \nu_1 \vdash M : \nu_2 \ \cdots \ \Sigma, x : \nu_{n-1} \vdash M : \nu_n}{\Sigma \vdash \mathsf{fix}\, x.M : \nu_n} \left( \begin{array}{l} \mathrm{BOT}(\nu_1) \text{ and} \\ \exists p, q.(p < p \text{ and } \nu_q \leq \nu_p) \end{array} \right)$$

does not change the power of the system.

So we can conclude that our strictness and totality type assignment system $\vdash_{\mathrm{fcst}}$ is an extension of the sound fragment of $\vdash_{\mathrm{CST}}$.

[4] The rule (fix) is indeed the formulation for our syntax of the corresponding rule of $\vdash_{\mathrm{CST}}$.

### 5.4.5 Separability of conjunctive strictness and totality analyses

In this section we prove that a separability result like the one in Section 4.4.5 holds also for $\vdash_{\text{fcst}}$, i.e. we show how the information that can be used to replace safely a lazy application by a strict one, obtained with the system $\vdash_{\text{fcst}}$ (in Definition 5.63), can be obtained from the systems $\vdash_{\text{cs}}$ (in Definition 5.20) and $\vdash_{\text{ct}}$ (in Definition 5.46).

The two systems, separately, have good properties: they both have simple inclusion between types, and have good type checking algorithms (see Remarks 5.32 and 5.51), while the combined system is rather awkward.

The proof is not particularly difficult, but is contains a lot of technical details. For clarity we have spit it in five steps.

1. We prove that $\vdash_{\text{fcst}}$ is as powerful as its two subsystems $\vdash_{\text{fcs}\downarrow\uparrow}$ (obtained by forbidding the use of the constructor $\not\perp$, and hence by removing rules ($\rightarrow \text{I}_{\not\perp}$) and ($\text{If}_{\not\perp}$) in Fig. 5.14) and $\vdash_{\text{fct}\downarrow\uparrow}$ (obtained by forbidding the use of the constructor $\perp$, and hence by removing rule ($\text{If}_{\perp}$) in Fig. 5.14).

2. We prove that $\vdash_{\text{fcs}\downarrow\uparrow}$ is as powerful as its subsystem $\vdash_{\text{fcs}}$, obtained by forbidding the use of the operators $\downarrow$ and $\uparrow$ and by replacing the rule $\text{If}_{\top}$ in Fig. 5.14 by the following rule
$$(\text{If}_{\top}{}^{\text{fcs}})\ \frac{\Sigma \vdash N : \top^{\mathsf{bool}} \qquad \Sigma \vdash M_1 : \nu \qquad \Sigma \vdash M_2 : \nu}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \nu}\ .$$
Similarly for the system $\vdash_{\text{fct}\downarrow\uparrow}$ and its subsystem $\vdash_{\text{fct}}$, obtained by forbidding the use of the operators $\downarrow$ and $\uparrow$ and by replacing the rule $\text{If}_{\top}$ in Fig. 5.14 by the following rule
$$(\text{If}_{\top}{}^{\text{fct}})\ \frac{\Sigma \vdash N : \top^{\mathsf{bool}} \qquad \Sigma \vdash M_1 : \nu \qquad \Sigma \vdash M_2 : \nu}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \top^{\epsilon(\nu)}}\ .$$

3. We prove that $\vdash_{\text{fcs}}$ is as powerful as the (PCF′ version of) system $\vdash_{\text{cs}}$ of Chapter 5.2 (which is obtained from $\vdash_{\text{fcs}}$ by restricting the occurrences of the conjunction operator), and similarly for the system $\vdash_{\text{fct}}$ and the (PCF′version of) system $\vdash_{\text{ct}}$ of Chapter 5.3.

4. We join the results of the previous steps and get that $\vdash_{\text{fcst}}$ is as powerful as the (PCF′ version of) systems $\vdash_{\text{cs}}$ and $\vdash_{\text{ct}}$.

5. Finally we apply to $\vdash_{\text{cs}}$, $\vdash_{\text{ct}}$ and $\vdash_{\text{fcst}}$ the same considerations done at the end of Section 4.4.5 for $\vdash_{\text{s}}$, $\vdash_{\text{t}}$ and $\vdash'_{\text{st}}$.

#### Step I (separating $\perp$ and $\not\perp$)

Let $\mathbf{L}^{\text{fcs}\downarrow\uparrow}$ ($\mathbf{L}^{\text{fct}\downarrow\uparrow}$) be the set of fcst-types that do not contain the constructor $\not\perp$ ($\perp$). Let $\leq_{\text{fcs}\downarrow\uparrow}$ and $\vdash_{\text{fcs}\downarrow\uparrow}$ ($\leq_{\text{fct}\downarrow\uparrow}$ and $\vdash_{\text{fct}\downarrow\uparrow}$) be the restriction of $\leq_{\text{fcst}}$ and $\vdash_{\text{fcst}}$ obtained by forbidding the use of the constructor $\not\perp$ ($\perp$).

As first step we prove a result (Theorem 5.67) which implies that both $\vdash_{\text{fcs}\downarrow\uparrow}$ and $\vdash_{\text{fct}\downarrow\uparrow}$ are conservative over $\vdash_{\text{fcst}}$. I.e. to derive a strictness property in $\in \mathbf{L}^{\text{fcs}\downarrow\uparrow}$ the distinction between the constructor $\not\perp$ and the constructor $\top$ is not relevant (by Theorem 5.67.1), and similarly to derive a totality property in $\in \mathbf{L}^{\text{fct}\downarrow\uparrow}$ the distinction between $\perp$ and $\top$ is not relevant (by Theorem 5.67.2).

**Lemma 5.66** *Let $\nu \leq_{\text{fcst}} \nu'$. Then*

1. $\nu[\not\perp := \top] \leq_{\text{fcs}\downarrow\uparrow} \nu'[\not\perp := \top]$.
2. $\nu[\perp := \top] \leq_{\text{fct}\downarrow\uparrow} \nu'[\perp := \top]$.

**Proof.** The proof of both 1. and 2. are by induction on the derivation of $\phi \leq'_{\text{st}} \psi$.

1. If $\nu \leq_{\text{fcst}} \nu'$ was derived by the rule $(\not\perp)$ or $(\uparrow_3)$, then we have that $\nu'[\not\perp := \top] \in \mathbf{L}_\top{}^{\text{fcs}\downarrow\uparrow}$, so $\nu[\not\perp := \top] \leq_{\text{fcs}\downarrow\uparrow} \nu'[\not\perp := \top]$ is derivable from the rule $(\top)$.
   If $\nu \leq_{\text{fcst}} \nu'$ was derived by the rule $(\downarrow_3)$, then $\nu[\not\perp := \top] \cong_{\text{fcs}\downarrow\uparrow} \nu'[\not\perp := \top]$ is derivable from the rules $(\top)$, $(\downarrow_1)$ and (Trans).
   In all the other cases $\nu[\not\perp := \top] \leq_{\text{fcs}\downarrow\uparrow} \nu'[\not\perp := \top]$ is derivable by the same rule.

2. If $\nu \leq_{\text{fcst}} \nu'$ was derived by the rule $(\perp)$ or $(\downarrow_4)$, then we have that $\nu'[\perp := \top] \in \mathbf{L}_\top{}^{\text{fct}\downarrow\uparrow}$, so $\nu[\perp := \top] \leq_{\text{fct}\downarrow\uparrow} \nu'[\perp := \top]$ is derivable from the rule $(\top)$.
   If $\nu \leq_{\text{fcst}} \nu'$ was derived by the rule $(\uparrow_4)$, then $\nu[\perp := \top] \cong_{\text{fct}\downarrow\uparrow} \nu'[\not\perp := \top]$ is derivable from the rules $(\top)$, $(\uparrow_1)$ and (Trans).
   In all the other cases $\nu[\perp := \top] \leq_{\text{fct}\downarrow\uparrow} \nu'[\perp := \top]$ is derivable by the same rule. □

**Theorem 5.67** *Let $\Sigma \vdash_{\text{fcst}} M : \nu$. Then*

1. $\Sigma[\not\perp := \top] \vdash_{\text{fcs}\downarrow\uparrow} M : \nu[\not\perp := \top]$.
2. $\Sigma[\perp := \top] \vdash_{\text{fct}\downarrow\uparrow} M : \nu[\perp := \top]$.

**Proof.** By induction on the derivation of $\Sigma \vdash_{\text{fcst}} M : \nu$.
If the last rule applied is (Var), then both 1. and 2. are obvious.
Let the last rule applied be (Con). Since constants have st-types that are either in $\mathbf{L}'^{\text{s}}$ or in $\mathbf{L}'^{\text{t}}$, we have that for both 1. and 2. the proof is immediate.
If the last rule applied is $(\rightarrow \text{I})$, or $(\rightarrow \text{E})$ both 1. and 2. are by inductive hypotheses.
If the last rule applied is $(\rightarrow \text{I}_{\not\perp})$, then $M = \lambda x.M'$ for some $M'$, and $\nu = \not\perp^\rho$. So $\Sigma[\perp := \top] \vdash_{\text{fct}\downarrow\uparrow} M : \nu[\perp := \top]$ is derivable by applying the same rule. On the other hand, $\nu[\not\perp := \top] = \top^\rho$, and it is easy to check that $\Sigma[\not\perp := \top] \vdash_{\text{fcs}\downarrow\uparrow} M : \top^\rho$.
If the last rule applied is $(\leq)$ both results derive from induction hypothesis and Lemma 5.66.
If the last rule applied is $(\text{If}_\perp)$. Then $\nu = \perp^\rho$, and $\nu[\perp := \top] = \top^\rho$. It is easy to check that $\Sigma[\perp := \top] \vdash_{\text{fct}\downarrow\uparrow} M : \top^\rho$. On the other hand from the induction hypothesis and applying the rule $(\text{If}_\perp)$, we derive $\Sigma[\not\perp := \top] \vdash_{\text{fcs}\downarrow\uparrow} M : \nu[\not\perp := \top] = \perp^\rho$.
If the last rule applied is $(\text{If}_{\not\perp})$, from the induction hypothesis and applying the rule $(\text{If}_{\not\perp})$,

we derive $\Sigma[\bot := \top] \vdash_{\mathrm{fct}\downarrow\uparrow} M : \nu[\bot := \top]$. On the other side to derive $\Sigma[\not\bot := \top] \vdash_{\mathrm{fcs}\downarrow\uparrow} M : \nu[\not\bot := \top]$ we have to apply the induction hypothesis, the the rule ($\mathrm{If}_\top$), and the fact that $\downarrow(\nu[\not\bot := \top]) \leq_{\mathrm{fcs}\downarrow\uparrow} \nu[\not\bot := \top]$.
If the last rule applied is ($\mathrm{If}_\top$) both 1. and 2. are by induction hypotheses followed by the application of the rule ($\mathrm{If}_\top$).
Finally, if the last rule applied is ($\mathrm{Fix}_\bot$) or ($\mathrm{Fix}_{\not\bot}$) apply induction hypothesis and the corresponding rule. □

### Step II (removing ↓ and ↑)

The non-standard types in $\mathbf{L}^{\mathrm{fcs}\downarrow\uparrow}$ ($\mathbf{L}^{\mathrm{fct}\downarrow\uparrow}$) may contain the ↓ and ↑ operator. Let $\mathbf{L}^{\mathrm{fcs}}$ ($\mathbf{L}^{\mathrm{fct}}$) be the set of fcs-types (fct-types) that do not contain the operators ↓ and ↑. Let $\leq_{\mathrm{fcs}}$ and $\vdash_{\mathrm{fcs}}$ ($\leq_{\mathrm{fct}}$ and $\vdash_{\mathrm{fct}}$) be the restriction of $\leq_{\mathrm{fcs}\downarrow\uparrow}$ and $\vdash_{\mathrm{fcs}\downarrow\uparrow}$ ($\leq_{\mathrm{fct}\downarrow\uparrow}$ and $\vdash_{\mathrm{fct}\downarrow\uparrow}$) obtained by forbidding the use of the operators ↓ and ↑ and by replacing the rule $\mathrm{If}_\top$ in Fig. 5.14 by the rule ${\mathrm{If}_\top}^{\mathrm{fcs}}$ (${\mathrm{If}_\top}^{\mathrm{fct}}$) introduced at the beginning of Section 5.4.5.

As second step we show a result (Lemma 5.68.2) which implies that $\vdash_{\mathrm{fcs}}$ is conservative over $\vdash_{\mathrm{fcs}\downarrow\uparrow}$, and a result (Lemma 5.69.2) which implies that $\vdash_{\mathrm{fct}}$ is conservative over $\vdash_{\mathrm{fct}\downarrow\uparrow}$.

For every type $\nu \in \mathbf{L}^{\mathrm{fcs}\downarrow\uparrow}$ ($\mathbf{L}^{\mathrm{fct}\downarrow\uparrow}$) let $\overline{\nu}^{\mathrm{s}}$ ($\overline{\nu}^{\mathrm{t}}$) be the type obtained in the following way.

- If $\nu \in \mathbf{L}^{\mathrm{fcs}\downarrow\uparrow}$, then (to get $\overline{\nu}^{\mathrm{s}}$) first replace the subexpressions $\uparrow(\nu')$ by $\top^{\epsilon(\nu')}$, and then replace the subexpressions $\downarrow(\nu')$ by $\nu'$.
- If $\nu \in \mathbf{L}^{\mathrm{fct}}$, then (to get $\overline{\nu}^{\mathrm{t}}$) first replace the subexpressions $\uparrow(\nu')$ by $\nu'$, and then replace the subexpressions $\downarrow(\nu')$ by $\top^{\epsilon(\nu')}$.

This notation is extended to $\vdash_{\mathrm{fcs}\downarrow\uparrow}$ (and $\vdash_{\mathrm{fct}\downarrow\uparrow}$) bases by defining, $\overline{\Sigma}^{\mathrm{s}} = \{x : \overline{\nu}^{\mathrm{s}} \mid x : \nu \in \Sigma\}$ (and $\overline{\Sigma}^{\mathrm{t}} = \{x : \overline{\nu}^{\mathrm{t}} \mid x : \nu \in \Sigma\}$).

It is easy to check that, for all $\nu \in \mathbf{L}^{\mathrm{fcs}\downarrow\uparrow}$, $\nu \cong_{\mathrm{fcs}\downarrow\uparrow} \overline{\nu}^{\mathrm{s}}$. Moreover we have that the following lemma holds.

**Lemma 5.68** *1. Let $\nu \leq_{\mathrm{fcs}\downarrow\uparrow} \nu'$. Then $\overline{\nu}^{\mathrm{s}} \leq_{\mathrm{fcs}} \overline{\nu'}^{\mathrm{s}}$.*

*2. Let $\Sigma \vdash_{\mathrm{fcs}\downarrow\uparrow} M : \nu$. Then $\overline{\Sigma}^{\mathrm{s}} \vdash_{\mathrm{fcs}} M : \overline{\nu}^{\mathrm{s}}$.*

**Proof.**

1. By induction of the derivation of $\nu \leq_{\mathrm{fcs}\downarrow\uparrow} \nu'$.
2. By induction of the derivation of $\Sigma \vdash_{\mathrm{fcs}\downarrow\uparrow} M : \nu$. □

For $\nu \in \mathbf{L}^{\mathrm{fct}\downarrow\uparrow}$, instead, it is not true that $\nu \cong_{\mathrm{fct}\downarrow\uparrow} \overline{\nu}^{\mathrm{t}}$, in fact, $\downarrow(\nu) \ncong_{\mathrm{fct}\downarrow\uparrow} \top^{\epsilon(\nu)}$. For instance, for $\xi = \top^{\mathsf{nat}} \to \not\bot^{\mathsf{nat}} \in \mathbf{L}^{\mathrm{fcst}}_{\not\bot} - \mathbf{L}^{\mathrm{fcst}^\bullet}_{\not\bot}$, $[\![\downarrow(\xi)]\!] \neq [\![\top^{\mathsf{nat}\to\mathsf{nat}}]\!]$, since the set of terms that approximate constant functions does not contain all the terms in $\mathsf{nat} \to \mathsf{nat}$. So there cannot be a sound rule that would make $\downarrow(\xi)$ and $\top^{\epsilon(\xi)}$ equivalent. In spite of this we can prove the following lemma.

**Lemma 5.69** *1. Let* $\nu \leq_{\mathrm{fct}\downarrow\uparrow} \nu'$. *Then* $\overline{\nu}^{\mathrm{t}} \leq_{\mathrm{fct}} \overline{\nu'}^{\mathrm{t}}$.

*2. Let* $\Sigma \vdash_{\mathrm{fct}\downarrow\uparrow} M : \nu$. *Then* $\overline{\Sigma}^{\mathrm{t}} \vdash_{\mathrm{fct}} M : \overline{\nu}^{\mathrm{t}}$.

**Proof.**

1. By induction of the derivation of $\nu \leq_{\mathrm{fct}\downarrow\uparrow} \nu'$.
2. By induction of the derivation of $\Sigma \vdash_{\mathrm{fct}\downarrow\uparrow} M : \nu$. □

**Step III (shifting to *simple formulae*)**

As third step we show a result (Theorem 5.70) which implies that $\vdash_{\mathrm{cs}}$ is conservative over $\vdash_{\mathrm{fcs}}$, and that $\vdash_{\mathrm{ct}}$ is conservative over $\vdash_{\mathrm{fct}}$.

The types obtained with the previous transformation still have intersections both at the top level and in the right-hand side of arrows. Applying repeatedly the equivalence: $\nu_1 \to (\nu_2 \wedge \nu_3) \cong_{\mathrm{fcst}} (\nu_1 \to \nu_2) \wedge (\nu_1 \to \nu_3)$ we can obtain from a type $\nu \in \mathbf{L}^{\mathrm{fcs}}$ ($\mathbf{L}^{\mathrm{fct}}$) an equivalent intersection of cs-types (ct-types) $\phi_1 \wedge \cdots \wedge \phi_n$ such that $\nu \cong_{\mathrm{fcs}} \phi_1 \wedge \cdots \wedge \phi_n$ ($\nu \cong_{\mathrm{fct}} \phi_1 \wedge \cdots \wedge \phi_n$).

1. For every $\nu \in \mathbf{L}^{\mathrm{fcs}}$, let $\mathbf{spl_{str}}(\nu)$ denote a cs-conjunction $\phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{C}^{\mathrm{cs}}$ such that $\phi_1 \wedge \cdots \wedge \phi_n \cong_{\mathrm{fcs}} \nu$, and
2. for every $\nu \in \mathbf{L}^{\mathrm{fct}}$, let $\mathbf{spl_{tot}}(\nu)$ denote a ct-conjunction $\phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{C}^{\mathrm{ct}}$ such that $\phi_1 \wedge \cdots \wedge \phi_n \cong_{\mathrm{fct}} \nu$.

This notation is extended to $\vdash_{\mathrm{fcs}}$ (and $\vdash_{\mathrm{fct}}$) bases by defining $\mathbf{spl_{str}}(\Sigma) = \{x : \mathbf{spl_{str}}(\nu') \mid x : \nu' \in \Sigma\}$ (and $\mathbf{spl_{tot}}(\Sigma) = \{x : \mathbf{spl_{tot}}(\nu') \mid x : \nu' \in \Sigma\}$).

**Theorem 5.70** *1. Let* $\Sigma \vdash_{\mathrm{fcs}} M : \nu$, *and let* $\mathbf{spl_{str}}(\nu) = \phi_1 \wedge \cdots \wedge \phi_n$. *Then* $\mathbf{spl_{str}}(\Sigma) \vdash_{\mathrm{cs}} M : \phi_i$, *for all* $i$, $1 \leq i \leq n$.

*2. Let* $\Sigma \vdash_{\mathrm{fct}} M : \nu$, *and let* $\mathbf{spl_{tot}}(\nu) = \phi_1 \wedge \cdots \wedge \phi_n$. *Then* $\mathbf{spl_{tot}}(\Sigma) \vdash_{\mathrm{ct}} M : \phi'_i$, *for all* $i$, $1 \leq i \leq n$. □

**Step IV (the separability result)**

As fourth step we show a result (Theorem 5.71) which implies that both $\vdash_{\mathrm{cs}}$ and $\vdash_{\mathrm{ct}}$ are conservative over $\vdash_{\mathrm{fcst}}$.

For every $\nu \in \mathbf{L}^{\mathrm{fcst}}$,

1. let $\overline{\mathbf{spl_{str}}(\nu)}$ denote a cs-conjunction $\phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{C}^{\mathrm{cs}}$ such that $\phi_1 \wedge \cdots \wedge \phi_n \cong_{\mathrm{fcs}} \overline{(\nu[\not\perp := \top])}^{\mathrm{s}}$, and
2. let $\overline{\mathbf{spl_{tot}}(\nu)}$ denote a ct-conjunction $\phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{C}^{\mathrm{ct}}$ such that $\phi_1 \wedge \cdots \wedge \phi_n \cong_{\mathrm{fct}} \overline{(\xi[\perp := \top])}^{\mathrm{t}}$.

This notations are extended to $\vdash_{\text{fcst}}$ bases by defining $\overline{\mathbf{spl_{str}}}(\Sigma) = \{x : \overline{\mathbf{spl_{str}}}(\nu') \mid x : \nu' \in \Sigma\}$ and $\overline{\mathbf{spl_{tot}}}(\Sigma) = \{x : \overline{\mathbf{spl_{tot}}}(\nu') \mid x : \nu' \in \Sigma\}$. We can now state the main result of this section.

**Theorem 5.71** *Let $\Sigma \vdash_{\text{fcst}} M : \nu$, and let $\overline{\mathbf{spl_{str}}}(\nu) = \phi_1 \wedge \cdots \wedge \phi_n$, and $\overline{\mathbf{spl_{tot}}}(\nu) = \phi'_1 \wedge \cdots \wedge \phi'_m$. Then*

1. *$\overline{\mathbf{spl_{str}}}(\Sigma) \vdash_{\text{cs}} M : \phi_i$, for all $i$, $1 \leq i \leq n$, and*
2. *$\overline{\mathbf{spl_{tot}}}(\Sigma) \vdash_{\text{ct}} M : \phi'_j$, for all $j$, $1 \leq j \leq m$.* □

**Final step (a consequence of the separability result)**

Now we show how, as a consequence of the previous theorem, it is possible to use the systems $\vdash_{\text{cs}}$ and $\vdash_{\text{ct}}$ of Sections 5.2.2 and 5.3.2 to perform the optimization that would be induced by the information gathered with the (significantly more complex) system $\vdash_{\text{fcst}}$.

As explained in the introduction at the beginning of Part II (page 35), given an application

$$FM_1 \cdots M_n,$$

where $n \geq 1$, we are interested in proving, for $k \in \{1, \ldots, n\}$,

1. either
$$\Sigma \vdash_{\text{fcst}} F : \nu_1 \to \cdots \to \nu_{k-1} \to \bot^\rho \to \nu_{k+1} \to \cdots \to \nu_n \to \bot^\sigma \ ,$$
where for all $i \in \{1, \ldots, k-1, k+1, \ldots, n\}$ $M_i$ has the property $\nu_i$,
2. or
$$\Sigma \vdash_{\text{fcst}} M_k : \not\bot^\rho \ ,$$

for some basis $\Sigma$. In fact in both cases the application to the $k$-th argument can be safely done "by value". Assume (without loss of generality) $n = 2$ and $k = 2$. By Theorem 5.71, we have that

1. $\Sigma \vdash_{\text{fcst}} F : \nu \to \bot^\rho \to \bot^\sigma$ implies
   - $\Sigma' \vdash_{\text{cs}} F : \xi \to \bot^\rho \to \bot^\sigma$, where $\Sigma' = \overline{\mathbf{spl_{str}}}(\Sigma)$ and $\xi = \overline{\mathbf{spl_{str}}}(\nu) = \phi_1 \wedge \cdots \wedge \phi_p$, moreover
   - if $\Sigma \vdash_{\text{fcst}} M_1 : \nu$ then (for all $j \in \{1, \ldots, p\}$) $\Sigma' \vdash_{\text{cs}} M_1 : \phi_j$.

   This means that, as far as we interested in properties like $\nu_1 \to \cdots \to \nu_{k-1} \to \bot^\rho \to \nu_{k+1} \to \cdots \to \nu_n \to \bot^\sigma$ above, using $\vdash_{\text{fcst}}$ or $\vdash_{\text{cs}}$ is not relevant.
2. $\Sigma \vdash_{\text{fcst}} M_2 : \not\bot^\rho$ implies
   - $\Sigma' \vdash_{\text{ct}} M_2 : \not\bot^\rho$, where $\Sigma' = \overline{\mathbf{spl_{tot}}}(\Sigma)$ (note that $\not\bot^\rho = \overline{\mathbf{spl_{tot}}}(\not\bot^\rho)$).

This means that, as far as we interested in properties like $\not\perp^{\rho}$, using $\vdash_{\mathrm{fcst}}$ or $\vdash_{\mathrm{ct}}$ is not relevant.

We note, however, that (as for $\vdash'_{\mathrm{st}}$ in Section 4.4.5) the separability of the two analyses is due to the weakness of the $\vdash_{\mathrm{fcst}}$ *type inference rules*. In fact the language of fcst-types allows to express properties which would *not* be expressible in the separate systems. For instance take $(\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \perp^{\mathsf{nat}}$. This is the property of the terms which map (terminating) constant functions in a divergent computation. If we consider the program:

$$M = \lambda\, f^{\mathsf{nat}\to\mathsf{nat}}.\mathsf{if}\ f1 = f2\ \mathsf{then}\ \mathsf{fix}\, x^{\mathsf{nat}}.x\ \mathsf{else}\ 3\ ,$$

then we have that $[M] \in [\![(\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \perp^{\mathsf{nat}}]\!]$, while $[M] \notin [\![(\top^{\mathsf{nat}} \to \top^{\mathsf{nat}}) \to \perp^{\mathsf{nat}}]\!]$. But, by Theorem 5.67, we *cannot* prove $\vdash_{\mathrm{fcst}} M : (\top^{\mathsf{nat}} \to \not\perp^{\mathsf{nat}}) \to \perp^{\mathsf{nat}}$.

## 5.5 Summary

The content of this chapter can be summarized as follows.

- The introduction of the non-standard type assignment for conjunctive strictness analysis $\vdash_{\mathrm{cs}}$ (in Section 5.2.2) and of the non-standard type assignment for conjunctive totality analysis $\vdash_{\mathrm{ct}}$ (in Section 5.3.2), which are sound w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$ of Section 2.4.1.

- The proof of completeness for the entailment relation for cs-types $\leq_{\mathrm{cs}}$ (of $\vdash_{\mathrm{cs}}$) and for the entailment relation for ct-types $\leq_{\mathrm{ct}}$ (of $\vdash_{\mathrm{ct}}$) w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}_{\mathsf{isdef},\mathsf{por}}$ of Section 2.4.2.

- The introduction (in Section 5.2.3) of the non-standard type assignment for conjunctive strictness analysis $\vdash^{\mathrm{g}}_{\mathrm{cs}}$ (which is sound w.r.t. the model $\mathcal{M}^{\mathrm{gnd}}$ of Section 2.4.1) with an entailment relation, $\leq^{\mathrm{g}}_{\mathrm{cs}}$, which is complete w.r.t. the model $\mathcal{M}^{\mathrm{gnd}}_{\mathsf{por}}$ of Section 2.4.2.

- The definition of the combined system for conjunctive strictness and totality $\vdash_{\mathrm{fcst}}$ which is sound w.r.t. the model $\mathcal{M}^{\mathrm{lazy}}$ of Section 2.4.1.

- The proof that (for the system $\vdash_{\mathrm{fcst}}$ introduced in this chapter, and so also for the system $\vdash_{\mathrm{CST}}$ presented in [Solberg, 1995a] Chapter 3 and [Solberg, 1995b]) the inference of cs-types and the inference of ct-types are independent. This implies that using a system only for strictness and one only for totality we can infer properties of terms that (w.r.t. the possibility of replacing safely a lazy application with a strict one) give the same information given by the properties inferred by the systems for combined strictness and totality.

  We have no idea of how to get a strictness and totality analysis where the combination does improve precision and we believe that the separability result is not due to the exact formulation of the analyses presented.

# Chapter 6

# An inference algorithm for strictness

In this chapter we define an inference algorithm for detecting strictness information based on the strictness type inference system $\vdash^{\mathrm{g}}_{\mathrm{s}}$ presented in Section 4.2.3. The algorithm rely on a syntax directed version of the system $\vdash^{\mathrm{g}}_{\mathrm{s}}$ that avoids a free use of the entailment rule.

A preliminary version of the material presented in this chapter appeared in [Damiani and Giannini, 1997], where the programming language considered does not include pairs.

## 6.1 A syntax directed version of $\vdash^{\mathrm{g}}_{\mathrm{s}}$

First of all we introduce a canonical representation for the s-types of Section 4.2.1. These canonical s-types, called s$\star$-types, are the subset of $\mathbf{L}^{\mathrm{s}}$ obtained by forbidding the use of the properties $\bot^{\rho}$ and $\top^{\rho}$ for $\rho \notin \{\mathsf{nat}, \mathsf{bool}\}$, i.e. by choosing as sets of basic properties $\mathbf{B}^{\mathrm{s}\star}(\iota) = \{\bot^{\iota}, \top^{\iota}\}$ for $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$, and $\mathbf{B}^{\mathrm{s}\star}(\rho) = \emptyset$ for $\rho \in \mathbf{T} - \{\mathsf{nat}, \mathsf{bool}\}$.

**Definition 6.1 (s$\star$-types)** *The language* $\mathbf{L}^{\mathrm{s}\star}$ *of the* s$\star$-types *is the subset of* $\mathbf{L}^{\mathrm{s}}$ *defined by the following grammar:* $\phi ::= a^{\iota} \mid \phi \to \phi \mid \phi \times \phi$, *where* $a \in \{\bot, \top\}$ *and* $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$.

**Definition 6.2 (Entailment relation for s$\star$-types)** *Let* $\phi, \psi \in \mathbf{L}^{\mathrm{s}\star}$. *We write* $\phi \leq^{\mathrm{g}}_{\mathrm{s}\star} \psi$ *to mean that* $\phi \leq \psi$ *is derivable by the rules in Fig. 6.1. By* $\cong^{\mathrm{g}}_{\mathrm{s}\star}$ *we denote the equivalence relation induced by* $\leq^{\mathrm{g}}_{\mathrm{s}\star}$.

The following fact guarantees that restricting to s$\star$-types does not reduce the expressive power of the language.

**Fact 6.3** *1. For all* $\phi \in \mathbf{L}^{\mathrm{s}}$ *there is* $\phi' \in \mathbf{L}^{\mathrm{s}\star}$ *such that* $\phi \cong^{\mathrm{g}}_{\mathrm{s}} \phi'$.

*2. For all* $\phi_1, \phi_2 \in \mathbf{L}^{\mathrm{s}\star}$, $\phi_1 \leq^{\mathrm{g}}_{\mathrm{s}} \phi_2$ *if and only if* $\phi_1 \leq^{\mathrm{g}}_{\mathrm{s}\star} \phi_2$. □

**Definition 6.4 (Syntax directed s$\star$-type assignment system)** *We write* $\Sigma \vdash^{\mathrm{g}}_{\mathrm{s}\star} M : \phi$ *to mean that* $\Sigma \vdash M : \phi$ *can be derived by the rules in Fig. 6.2.*

$$(\bot_{\mathrm{g}}{}^{\star})\ \frac{\phi \in \mathbf{L}^{\mathrm{s}\star}_{\bot\mathrm{g}} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi} \qquad (\top^{\star})\ \frac{\psi \in \mathbf{L}^{\mathrm{s}\star}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}$$

$$(\mathrm{Ref}^{\star})\ \phi \leq \phi \qquad (\to^{\star})\ \frac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_2 \to \psi_1 \leq \phi_1 \to \psi_2} \qquad (\times^{\star})\ \frac{\phi_1 \leq \psi_1 \quad \phi_2 \leq \psi_2}{\phi_1 \times \phi_2 \leq \psi_1 \times \psi_2}$$

Figure 6.1: Entailment rules for s⋆-types

$$(\mathrm{Var}^{\star})\ \frac{\phi_1 \leq \phi_2}{\Sigma, x : \phi_1 \vdash x : \phi_2} \qquad (\mathrm{Con}^{\star})\ \frac{\phi_1 \in \mathbf{L}^{\mathrm{s}\star}(\mathsf{c}) \quad \phi_1 \leq \phi_2}{\Sigma \vdash \mathsf{c} : \phi_2}$$

$$(\to \mathrm{I}^{\star})\ \frac{\Sigma, x : \phi \vdash M : \psi}{\Sigma \vdash \lambda x.M : \phi \to \psi} \qquad (\to \mathrm{E}^{\star})\ \frac{\Sigma \vdash M : \phi_1 \to \psi \quad \Sigma \vdash N : \phi_2 \quad \phi_1 \to \psi \leq \phi_2 \to \psi}{\Sigma \vdash MN : \psi}$$

$$(\times \mathrm{I}^{\star})\ \frac{\Sigma \vdash M_1 : \phi_1 \quad \Sigma \vdash M_2 : \phi_2}{\Sigma \vdash \langle M_1, M_2 \rangle : \phi_1 \times \phi_2} \qquad (\times \mathrm{E}^{\star}_i)\ \frac{\Sigma \vdash M : \phi_1 \times \phi_2}{\Sigma \vdash \mathsf{proj}_i M : \phi_i}\ i \in \{1, 2\}$$

$$(\mathrm{Fix}^{\star})\ \frac{\begin{array}{c}\Sigma, x : \phi_1 \vdash M : \psi_1 \ \cdots\ \Sigma, x : \phi_k \vdash M : \psi_k \\ \forall i \in \{1, \ldots, k-1\}.\psi_i \leq \phi_{i+1} \\ \psi_k \leq \phi_1 \quad \psi_k \leq \psi\end{array}}{\Sigma \vdash \mathsf{fix}\, x.M : \psi}\ k \geq 1$$

$$(\mathrm{If}^{\star}_{\bot})\ \frac{\Sigma \vdash N : \bot^{\mathsf{bool}} \quad \vdash_{\mathbf{T}} M_1 : \rho \quad \vdash_{\mathbf{T}} M_2 : \rho \quad \epsilon(\psi) = \rho}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \psi}$$

$$(\mathrm{If}^{\star}_{\top})\ \frac{\Sigma \vdash N : \top^{\mathsf{bool}} \quad \Sigma \vdash M_1 : \phi_1 \quad \Sigma \vdash M_2 : \phi_2 \quad \phi_1 \leq \psi \quad \phi_2 \leq \psi}{\Sigma \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 : \psi}$$

Figure 6.2: Rules for s⋆-type assignment

It is worth mentioning that, in the rule ($\to \mathrm{E}^\star$), the condition $\phi_1 \to \psi \leq \phi_2 \to \psi$ is used instead of $\phi_2 \leq \phi_1$. This is done to take into account the fact that: if $\phi_2 \to \psi$ is a $\top$-s$\star$-type, then $\phi_1$ can be any s$\star$-type such that $\epsilon(\phi_1) = \epsilon(\phi_2)$. It is easy to see that the syntax directed system $\vdash^{\mathrm{g}}_{\mathrm{s}\star}$ is as powerful as the system $\vdash^{\mathrm{g}}_{\mathrm{s}}$, i.e. the following fact holds.

**Fact 6.5** *Let $\Sigma \vdash^{\mathrm{g}}_{\mathrm{s}} M : \phi$, then $\Sigma' \vdash^{\mathrm{g}}_{\mathrm{s}\star} M : \phi'$, for some $\Sigma'$ and $\phi'$ such that*

1. *$\Sigma' \cong^{\mathrm{g}}_{\mathrm{s}} \Sigma$, and*
2. *$\phi' \cong^{\mathrm{g}}_{\mathrm{s}} \phi$.* □

Note that in the rule for fixpoint of Definition 6.4, the number $k$ of premises allowed to assign an s$\star$-type to a fixpoint can be limited without affecting the power of the system. Let $[\phi]_{\cong^{\mathrm{g}}_{\mathrm{s}\star}}$ denote the $\cong^{\mathrm{g}}_{\mathrm{s}\star}$-equivalence class of the s$\star$-type $\phi$, i.e. $[\phi]_{\cong^{\mathrm{g}}_{\mathrm{s}\star}} = \{\phi' \mid \phi' \cong^{\mathrm{g}}_{\mathrm{s}\star} \phi\}$, and let $\mathbf{size}(\phi)$ denote the number of equivalence classes of the s$\star$-types $\chi$ such that $\epsilon(\chi) = \epsilon(\phi)$ and $\chi \not\leq^{\mathrm{g}}_{\mathrm{s}\star} \phi \not\leq^{\mathrm{g}}_{\mathrm{s}\star} \chi$. For every type $\rho \in \mathbf{T}$, $\mathbf{max_{size}}(\rho)$ denotes the maximum element of $\{\mathbf{size}(\phi) \mid \phi \in \mathbf{L}^{\mathrm{s}\star}(\rho)\}$.

**Proposition 6.6** *The system of Definition 6.4 is as powerful as the system obtained by replacing the side condition "$k \geq 1$" in rule* (Fix$^\star$) *with the condition "$1 \leq k \leq \mathbf{max_{size}}(\epsilon(\psi))$".*

**Proof.** Consider an application of the rule (Fix$^\star$):

$$(\mathrm{Fix}^\star)\ \frac{\begin{array}{c}\Sigma, x:\phi_1 \vdash M:\psi_1\ \cdots\ \Sigma, x:\phi_k \vdash M:\psi_k \\ \forall i \in \{1,\ldots,k-1\}.\psi_i \leq \phi_{i+1} \\ \psi_k \leq \phi_1 \qquad \psi_k \leq \psi\end{array}}{\Sigma \vdash \mathsf{fix}\, x.M : \psi}\ k \geq 1.$$

It is easy to see that, for every $\vdash^{\mathrm{g}}_{\mathrm{s}\star}$-typing statement $\Sigma, x : \phi_1 \vdash^{\mathrm{g}}_{\mathrm{s}\star} M : \psi_1$ and for every s$\star$-type $\phi'$ such that and $\phi' \leq^{\mathrm{g}}_{\mathrm{s}\star} \phi_1$, we have that

$$\Sigma, x : \phi' \vdash^{\mathrm{g}}_{\mathrm{s}\star} M : \psi',$$

for some s$\star$-type $\psi'$ such that $\psi' \leq^{\mathrm{g}}_{\mathrm{s}\star} \psi_1$.

So if we allow only applications of the rule (Fix$^\star$) in which, for all $i \in \{1,\ldots,k-1\}$, $\psi_i \not\leq^{\mathrm{g}}_{\mathrm{s}\star} \phi_1 \not\leq^{\mathrm{g}}_{\mathrm{s}\star} \psi_i$, we obtain an equivalent system.

But there are at most $\mathbf{size}(\phi_1) - 1$ s$\star$-type having this property, so if we allow only applications of the rule (Fix$^\star$) in which $1 \leq k \leq \mathbf{max_{size}}(\epsilon(\psi))$ we obtain an equivalent system. □

**Example 6.7** *Let $\rho = \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$. It is easy to check that $\{[\phi]_{\cong^{\mathrm{g}}_{\mathrm{s}\star}} \mid \epsilon(\phi) = \rho\}$ has 9 elements and $\mathbf{max_{size}}(\rho) = 3$. So 3 is an upper bound to the number of premises that are needed in the rule* (Fix$^\star$) *to assign any s$\star$-type $\phi$ such that $\epsilon(\phi) = \rho$.*

## 6.2 Strictness type schemes

Given a PCFP term $M$ the inference algorithm returns a description of all the possible pairs $\langle \Sigma, \phi \rangle$ such that $\Sigma \vdash^{g}_{s\star} M : \phi$. To describe sets of s$\star$-types we introduce the notions of s$\star$-type pattern and s$\star$-type scheme.

**Definition 6.8 (s$\star$-type patterns)** *Let $\mathcal{A}$ be the set of* basic property constructor variables *(*constructor variables *for short), ranged by $\alpha$, $\beta$, $\gamma$, ...*
*The language* $\mathbf{P}$ *of* s$\star$-type patterns *(*s$\star$-patterns *for short), ranged over by $\theta$, $\eta$,..., is defined from the grammar of Definition 6.1 by replacing the basic property constructors $\bot$ and $\top$ by constructor variables, i.e. $\theta ::= \alpha^{\iota} \mid \theta \to \theta \mid \theta \times \theta$, where $\alpha \in \mathcal{A}$ and $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$.*

The constructor variables can be renamed or instantiated to basic property constructors.

**Definition 6.9 (Renaming and instantiation)**
1. *A* renaming *is a one–to–one mapping $\mathbf{r} : \mathcal{A} \to \mathcal{A}$.*
2. *An* instantiation *is a mapping $\mathbf{i} : \mathcal{A} \to \{\bot, \top\}$.*

Both renaming and instantiation can be extended to basic property constructors (by defining $\mathbf{i}(a) = a$ and $\mathbf{r}(a) = a$, for $a \in \{\bot, \top\}$) and to s$\star$-types and s$\star$-patterns (in the obvious way). For example: $\mathbf{i}(\alpha^{\mathsf{nat}} \to \beta^{\mathsf{nat}}) = \mathbf{i}(\alpha)^{\mathsf{nat}} \to \mathbf{i}(\beta)^{\mathsf{nat}}$. Of course, for any s$\star$-type $\phi \in \mathbf{L}^{s\star}$, $\mathbf{i}(\phi) = \phi$ and $\mathbf{r}(\phi) = \phi$.

**Definition 6.10 (S$\star$-constraints)** *Let $\zeta$ range over $\{\bot, \top\} \cup \mathcal{A}$. An* s$\star$-constraint *is a formula of one of the following shapes:*

- $\zeta_1 \sqsubseteq \zeta_2$
- $(\bot \ \mathsf{in}\ \mathcal{G}) \Rightarrow \mathcal{E}$, *where $\mathcal{G}$ is a finite not empty subset of $\{\bot\} \cup \mathcal{A}$ and $\mathcal{E}$ is a finite set of s$\star$-constraints*
- $\mathcal{E}_1 \bigvee \mathcal{E}_2$, *where $\mathcal{E}_1$ and $\mathcal{E}_2$ are finite set of s$\star$-constraints.*

The symbol $\sqsubseteq$ denotes the order relation (between the constructors $\bot$ and $\top$) defined by: $\bot \sqsubseteq \bot$, $\bot \sqsubseteq \top$ and $\top \sqsubseteq \top$. The meaning of the s$\star$-constraints is given by the satisfaction relation that follows.

**Definition 6.11** *An instantiation* $\mathbf{i}$ *satisfies a set of s$\star$-constraints $\mathcal{E}$ if*

- $\zeta_1 \sqsubseteq \zeta_2 \in \mathcal{E}$ *implies* $\mathbf{i}(\zeta_1) \sqsubseteq \mathbf{i}(\zeta_2)$ *(that is either $\mathbf{i}(\zeta_1) = \delta$ or $\mathbf{i}(\zeta_2) = \top$)*
- $(\bot \,\mathsf{in}\, \mathcal{G}) \Rightarrow \mathcal{E}' \in \mathcal{E}$ *implies that, if $\bot \in \mathcal{G}$, then* $\mathbf{i}$ *satisfies $\mathcal{E}'$*
- $\mathcal{E}_1 \bigvee \mathcal{E}_2 \in \mathcal{E}$ *implies that* $\mathbf{i}$ *satisfies $\mathcal{E}_1$ or* $\mathbf{i}$ *satisfies $\mathcal{E}_2$.*

The set of all the instantiations that satisfy $\mathcal{E}$ is denoted by $\mathbf{sat}(\mathcal{E})$.

**Definition 6.12 (s⋆-type schemes)** *An* s⋆-type scheme *is a pair* $\langle \theta, \mathcal{E} \rangle$ *where* $\theta$ *is a* s⋆*-pattern and* $\mathcal{E}$ *is a finite set of* s⋆*-constraints.*

An s⋆-type scheme $\langle \theta, \mathcal{E} \rangle$ represents the set of s⋆-types that can be obtained from the s⋆-pattern $\theta$ by replacing constructor variables with basic property constructors in such a way that the s⋆-constraints in $\mathcal{E}$ are satisfied, i.e. $\langle \theta, \mathcal{E} \rangle$ represents all the s⋆-types $\mathbf{i}(\theta)$, for any $\mathbf{i} \in \mathbf{sat}(\mathcal{E})$.

**Definition 6.13** *Let* $\mathbf{i}_1$, $\mathbf{i}_2$ *be instantiation. We write* $\mathbf{i}_1 \sqsubseteq \mathbf{i}_2$ *if, for all* $\alpha \in \mathcal{A}$, $\mathbf{i}_1(\alpha) \sqsubseteq \mathbf{i}_2(\alpha)$.

A set of s⋆-constraints is said *choice-free* if it does not contain s⋆-constraints of the shape $\mathcal{E}_1 \bigvee \mathcal{E}_2$.

**Fact 6.14** *1. Let* $\mathcal{E}$ *be a choice-free finite set of* s⋆*-constraints. Then the set* $\mathbf{sat}(\mathcal{E})$ *is either empty or it has a maximum element.*

*2. Let* $\mathcal{E}$ *be a finite set of* s⋆*-constraints. Then the set* $\mathbf{sat}(\mathcal{E})$ *has a finite number of maximal elements.* □

**Example 6.15** *Consider the sets of* s⋆*-constraints:*

$$\begin{array}{rcl}
\mathcal{E} & = & \{\ (\bot \,\mathsf{in}\, \{\alpha_2'\}) \Rightarrow \{\ (\bot \,\mathsf{in}\, \{\alpha_2'\}) \Rightarrow \{\ \alpha_2 \sqsubseteq \alpha_2',\ \alpha_1' \sqsubseteq \alpha_1\ \},\ \alpha_2'' \sqsubseteq \alpha_1',\ (\bot \,\mathsf{in}\, \{\alpha_2''\}) \Rightarrow \{\ (\bot \,\mathsf{in}\, \{\alpha_2''\}) \Rightarrow \{\alpha_2 \sqsubseteq \alpha_2'', \alpha_1'' \sqsubseteq \alpha_1\},\ \beta' \sqsubseteq \alpha_1'',\ \beta \sqsubseteq \beta'\ \}\ \}\ \},\\
\mathcal{E}_1 & = & \{\ \alpha_2' \sqsubseteq \bot,\ \beta \sqsubseteq \bot,\ \alpha_2 \sqsubseteq \bot,\ \alpha_1 \sqsubseteq \bot\ \},\\
\mathcal{E}_2 & = & \{\ \alpha_2' \sqsubseteq \bot,\ \top \sqsubseteq \beta,\ \alpha_2 \sqsubseteq \bot,\ \top \sqsubseteq \alpha_1\ \},\\
\mathcal{E}_3 & = & \{\ \alpha_2' \sqsubseteq \bot,\ \beta \sqsubseteq \bot,\ \top \sqsubseteq \alpha_2\ \}.
\end{array}$$

*To find the maximum element* $\mathbf{i}_1$ *of* $\mathbf{sat}(\mathcal{E} \cup \mathcal{E}_1)$ *observe that from the four* s⋆*-constraints in* $\mathcal{E}_1$ *we get* $\mathbf{i}_1(\alpha_2') = \mathbf{i}_1(\beta) = \mathbf{i}_1(\alpha_2) = \mathbf{i}_1(\alpha_1) = \bot$. *Then from* $\mathcal{E}$ *we get* $\mathbf{i}_1(\alpha_1') = \mathbf{i}_1(\alpha_2'') = \mathbf{i}_1(\alpha_1'') = \mathbf{i}(\beta') = \bot$.

*Let* $\mathcal{I}_1 = \{\alpha_1, \alpha_1', \alpha_1'', \alpha_2, \alpha_2', \alpha_2'', \beta, \beta'\}$, *then* $\mathbf{i}_1$ *defined by:* $\mathbf{i}_1(\alpha) = \bot$ *if* $\alpha \in \mathcal{I}_1$ *and* $\mathbf{i}_1(\alpha) = \top$ *otherwise, is the maximum element of* $\mathbf{sat}(\mathcal{E} \cup \mathcal{E}_1)$.

*Similarly observe that from* $\mathcal{E}_2$ *we get* $\mathbf{i}_2(\alpha_2') = \mathbf{i}_2(\alpha_2) = \bot$ *and* $\mathbf{i}_2(\beta) = \mathbf{i}_2(\alpha_1) = \top$. *Let* $\mathcal{I}_2 = \{\alpha_2, \alpha_2'\}$, *we have that the maximum element* $\mathbf{i}_2$ *of* $\mathbf{sat}(\mathcal{E} \cup \mathcal{E}_2)$ *is defined by:* $\mathbf{i}_2(\alpha) = \bot$ *if* $\alpha \in \mathcal{I}_2$ *and* $\mathbf{i}_2(\alpha) = \top$ *otherwise. Notice that* $\mathbf{i}_2$ *is also the maximum element of* $\mathbf{sat}(\mathcal{E} \cup \{\alpha_2' \sqsubseteq \bot\})$.

*Looking for the maximum element* $\mathbf{i}_3$ *of* $\mathbf{sat}(\mathcal{E} \cup \mathcal{E}_3)$, *we get from* $\mathcal{E}_3$ *that* $\mathbf{i}_3(\alpha_2') = \mathbf{i}_1(\beta) = \bot$ *and* $\mathbf{i}_3(\alpha_2) = \top$. *Then from* $\mathcal{E}$ *we get* $\top \sqsubseteq \bot$. *This means that* $\mathbf{sat}(\mathcal{E} \cup \mathcal{E}_3)$ *is empty.*

(SUCCESS) $\dfrac{}{\mathcal{E} \rightsquigarrow \{\emptyset\}}$ if no other rule can be applied

(FAILURE) $\dfrac{}{\mathcal{E},\, \top \sqsubseteq \bot \rightsquigarrow \emptyset}$

(BOTTOM) $\dfrac{\mathcal{E}[\alpha := \bot] \rightsquigarrow \mathcal{S}}{\mathcal{E},\, \{\alpha \sqsubseteq \bot\} \rightsquigarrow \{\mathcal{I} \cup \{\alpha\} \mid \mathcal{I} \in \mathcal{S}\}}$

(TOP) $\dfrac{\mathcal{E}[\alpha := \bot] \rightsquigarrow \mathcal{S}}{\mathcal{E},\, \top \sqsubseteq \alpha \rightsquigarrow \mathcal{S}}$

(GUARD) $\dfrac{\mathcal{E} \cup \mathcal{E}'' \rightsquigarrow \mathcal{S} \quad \bot \in \mathcal{G}}{\mathcal{E},\, (\bot \,\mathsf{in}\, \mathcal{G}) \Rightarrow \mathcal{E}'' \rightsquigarrow \mathcal{S}}$

(CHOICE) $\dfrac{\mathcal{E} \cup \mathcal{E}_1 \rightsquigarrow \mathcal{S}_1 \quad \mathcal{E} \cup \mathcal{E}_2 \rightsquigarrow \mathcal{S}_2}{\mathcal{E},\, \mathcal{E}_1 \bigvee \mathcal{E}_2 \rightsquigarrow \mathbf{comp}(\mathcal{S}_1, \mathcal{S}_2)}$

Figure 6.3: "Natural semantics" rules for s⋆-constraints solution

## 6.3 Solving the s⋆-constraints

### 6.3.1 Finding the set of the maximal solutions

In this section we define an algorithm that, given a finite set of s⋆-constraints $\mathcal{E}$, finds the maximal elements $\mathbf{i}_1, \ldots, \mathbf{i}_n$ ($n \geq 0$) of $\mathbf{sat}(\mathcal{E})$. The algorithm is presented in natural semantics style using judgements $\mathcal{E} \rightsquigarrow \mathcal{S}$, where $\mathcal{S} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ and each $\mathcal{I}_i$, $1 \leq i \leq n$, is the set of constructor variables that *represents* $\mathbf{i}_i$, i.e. such that if $\alpha \in \mathcal{I}_i$ then $\mathbf{i}_i(\alpha) = \bot$ else $\mathbf{i}_i(\alpha) = \top$. The idea is simply that of recognizing, following the inequalities, all the variables that are forced to represent $\bot$. All other atomic variables are then replaced by $\top$ in a maximal solution.

To describe the algorithm we need a preliminary notation. Let $\mathcal{S}_1 = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ and $\mathcal{S}_2 = \{\mathcal{I}'_1, \ldots, \mathcal{I}'_m\}$. Then $\mathbf{comp}(\mathcal{S}_1, \mathcal{S}_2)$ denotes the set of the maximal (w.r.t. the preorder $\sqsubseteq$ of Definition 6.13) elements in $\mathcal{S}_1 \cup \mathcal{S}_2$, i.e. $\mathbf{comp}(\mathcal{S}_1, \mathcal{S}_2) =$

$$\{\mathcal{I} \mid \mathcal{I} \in \mathcal{S}_1 \cup \mathcal{S}_2 \text{ and for no } \mathcal{I}' \in \mathcal{S}_1 \cup \mathcal{S}_2,\, \mathcal{I}' \subset \mathcal{I}\} \ .$$

**Definition 6.16 ("Natural semantics" rules for s⋆-constraints solution)** *Let $\mathcal{E}$ be a finite set of s⋆-constraints, and let* cns *be an s⋆-constraint. The expression $\mathcal{E}$,* cns *denotes the set of constraints $\mathcal{E} \cup \{\text{cns}\}$ where it is assumed that* cns $\notin \mathcal{E}$*. We write $\mathcal{E} \rightsquigarrow \mathcal{S}$ to mean that this judgement is derivable by the rules in Fig. 6.3.*

**Proposition 6.17** *Let $\mathcal{E}$ be a finite set of s⋆-constraints and $n \geq 0$. Then $\mathcal{E} \rightsquigarrow \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ if and only if $\mathcal{I}_1, \ldots, \mathcal{I}_n$ represent the maximal elements of $\mathbf{sat}(\mathcal{E})$.* □

(FAILURE′) $\dfrac{}{\mathcal{E},\ \top \sqsubseteq \bot \rightsquigarrow' \text{FALSE}}$

(BOTTOM′) $\dfrac{\mathcal{E}[\bot/\alpha] \rightsquigarrow' b}{\mathcal{E},\ \alpha \sqsubseteq \bot \ \rightsquigarrow' b}$

(TOP′) $\dfrac{\mathcal{E}[\top/\alpha] \rightsquigarrow' b}{\mathcal{E},\ \top \sqsubseteq \alpha \rightsquigarrow' b}$

(GUARD′) $\dfrac{\mathcal{E} \cup \mathcal{E}'' \rightsquigarrow' b \quad \bot \in \mathcal{G}}{\mathcal{E},\ (\bot \,\mathsf{in}\, \mathcal{G}) \Rightarrow \mathcal{E}'' \rightsquigarrow' b}$ if no previous rule can be applied

(CHOICE$_1$′) $\dfrac{\mathcal{E} \cup \mathcal{E}_1 \rightsquigarrow' \text{TRUE}}{\mathcal{E},\ \mathcal{E}_1 \bigvee \mathcal{E}_2 \rightsquigarrow' \text{TRUE}}$ if no previous rule can be applied

(CHOICE$_2$′) $\dfrac{\mathcal{E} \cup \mathcal{E}_2 \rightsquigarrow' b}{\mathcal{E},\ \mathcal{E}_1 \bigvee \mathcal{E}_2 \rightsquigarrow' b}$ if no previous rule can be applied

(SUCCESS′) $\dfrac{}{\mathcal{E} \rightsquigarrow' \text{TRUE}}$ if no previous rule can be applied .

Figure 6.4: "Natural semantics" rules for s⋆-constraints checking

**Remark 6.18** *An efficient implementation of the algorithm specified by the rules in Fig. 6.3 would try to delay the application of the rules* (GUARD) *and* (CHOICE) *as much as possible to first detect possible failures.*

### 6.3.2 Checking for consistency

In this section we present an algorithm for checking whether a given set of s⋆-constraints $\mathcal{E}$ is satisfiable. The algorithm is presented in natural semantics style using judgements $\mathcal{E} \rightsquigarrow' b$, where $b \in \{\text{TRUE}, \text{FALSE}\}$, with the property that $\mathcal{E} \rightsquigarrow' \text{FALSE}$ if and only if $\mathbf{sat}(\mathcal{E}) = \emptyset$.

The algorithm $\rightsquigarrow'$ could be immediately defined on top of the algorithm $\rightsquigarrow$ of Definition 6.16 by observing that

$$\mathcal{E} \rightsquigarrow' \begin{cases} \text{FALSE} & \text{if } \mathcal{E} \rightsquigarrow \emptyset \\ \text{TRUE} & \text{otherwise.} \end{cases}$$

However we prefer to axiomatize directly the judgement $\rightsquigarrow'$.

**Definition 6.19 ("Natural semantics" rules for s⋆-constraints checking)** *We write $\mathcal{E} \rightsquigarrow' b$ to mean that this judgement is derivable by the rules in Fig. 6.4.*

**Proposition 6.20** *Let $\mathcal{E}$ be a finite set of s⋆-constraints. Then if $\mathbf{sat}(\mathcal{E}) = \emptyset$ then $\mathcal{E} \rightsquigarrow'$* FALSE *else $\mathcal{E} \rightsquigarrow'$* TRUE*.* □

## 6.4 The algorithm $\mathcal{W}^{s\star}$

To define the inference algorithm $\mathcal{W}^{s\star}$ we need some preliminary notations. Let $\rho$ be a type.

- By $\mathbf{fresh}(\rho)$ we denote the s$\star$-pattern obtained from $\rho$ by associating a fresh constructor variable to each occurrence of any ground type in $\rho$. For example: $\mathbf{fresh}(\mathsf{nat} \to \mathsf{nat}) = \alpha^{\mathsf{nat}} \to \beta^{\mathsf{nat}}$. For a set of term variables, $\Gamma$, $\mathbf{fresh}(\Gamma) = \{x : \mathbf{fresh}(\rho) \mid x^{\rho} \in \Gamma\}$.

- The function **vars** maps a s$\star$-pattern to its finite set of constructor variables. For example: $\mathbf{vars}(\alpha^{\mathsf{nat}} \to \beta^{\mathsf{nat}}) = \{\alpha, \beta\}$.

- The function **tail**, that maps s$\star$-patterns and s$\star$-types to finite subsets of $\{\bot, \top\} \cup \mathcal{A}$, is inductively defined by: $\mathbf{tail}(\zeta^{\iota}) = \{\zeta\}$, $\mathbf{tail}(\theta \to \eta) = \mathbf{tail}(\eta)$, and $\mathbf{tail}(\theta \times \eta) = \mathbf{tail}(\theta) \cup \mathbf{tail}(\eta)$. For example: $\mathbf{tail}(\alpha^{\mathsf{nat}} \to (\beta^{\mathsf{nat}} \times \gamma^{\mathsf{bool}})) = \{\beta, \gamma\}$.

- The function $\epsilon : \mathbf{L}^{s\star} \to \mathbf{T}$ is extended in the obvious way to s$\star$-patterns. For example: $\epsilon(\alpha^{\mathsf{nat}} \to \beta^{\mathsf{nat}}) = \mathsf{nat} \to \mathsf{nat}$.

- Let $\theta_1$, $\theta_2$ be s$\star$-patterns or s$\star$-types such that $\epsilon(\theta_1) = \epsilon(\theta_2)$, then $\mathbf{cs}_{\leq}(\theta_1, \theta_2)$ and $\mathbf{ucs}_{\leq}(\theta_1, \theta_2)$ denote the set of s$\star$-constraints inductively defined by the clauses in Fig. 6.5 and 6.6. Notice that for all instantiations $\mathbf{i}$:

  - $\mathbf{i}(\theta_1) \leq^{g}_{s\star} \mathbf{i}(\theta_2)$ if and only if $\mathbf{i} \in \mathbf{sat}(\mathbf{cs}_{\leq}(\theta_1, \theta_2))$, and
  - $\mathbf{i}(\theta_2) \notin \mathbf{L}^{s\star}_{\top}$ implies: $\mathbf{i}(\theta_1) \leq^{g}_{s\star} \mathbf{i}(\theta_2)$ if and only if $\mathbf{i} \in \mathbf{sat}(\mathbf{ucs}_{\leq}(\theta_1, \theta_2))$.

  In particular $\mathbf{cs}_{\leq}(\theta_1, \theta_2)$ consists of the single s$\star$-constraint $(\bot \,\mathsf{in}\, \mathbf{tail}(\theta_2)) \Rightarrow \mathbf{ucs}_{\leq}(\theta_1, \theta_2)$, which is trivially satisfied by all the instantiations $\mathbf{i}$ such that $\mathbf{i}(\mathbf{tail}(\theta_2)) = \{\top\}$, i.e. such that $\mathbf{i}(\theta_2) \in \mathbf{L}^{s\star}_{\top}$.

- For every constant $\mathsf{c}$ an s-type scheme $\mathbf{sts}(\mathsf{c})$ is specified. For example, for any integer $\mathsf{n}$, $\mathbf{sts}(\mathsf{n}) = \langle \alpha^{\mathsf{nat}}, \{\top \sqsubseteq \alpha\} \rangle$ and $\mathbf{sts}(+) = \langle {\alpha_1}^{\mathsf{nat}} \times {\alpha_2}^{\mathsf{nat}} \to \beta^{\mathsf{nat}}, \{(\bot \,\mathsf{in}\, \{\beta\}) \Rightarrow \{\{\alpha_1 \sqsubseteq \bot\} \bigvee \{\alpha_2 \sqsubseteq \bot\}\}\} \rangle$.

We can now proceed to define the s$\star$-type inference algorithm $\mathcal{W}^{s\star}$. This algorithm is presented in Fig. 6.7 and in Fig. 6.8. Let $\vdash_{\mathbf{T}} M : \rho$ and $\mathcal{W}^{s\star}(M) = \langle \Theta, \theta, \mathcal{E} \rangle$, then $\Theta$ is a basis that associates a s$\star$-pattern to each term variable in $\mathrm{FV}(M)$, $\theta$ is a s$\star$-pattern, and $\mathcal{E}$ is a finite set of s$\star$-constraints.

Proposition 6.6 implies that to get a complete inference algorithm it is sufficient to build a set of s$\star$-constraints that codifies the applications of the rule (Fix$^{\star}$) for all the values of $k$ less or equal than $\mathbf{max}_{\mathbf{size}}(\rho)$, where $\rho$ is the type of the fixpoint expression. The set $\mathcal{S}(1)$ (where $\mathcal{S}$ is the function defined in the last branch of the "case" in the algorithm $\mathcal{W}^{s\star}_0$ in Fig. 6.8) codifies such applications. For example let $\rho = \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$. The set $\{[\phi]_{\cong^{g}_{s\star}} \mid \epsilon(\phi) = \rho\}$ has 9 elements while $\mathbf{max}_{\mathbf{size}}(\rho) = 3$. So 3 is an upper bound to

$$\mathbf{cs}_{\leq}({\zeta_1}^{\iota}, {\zeta_2}^{\iota}) = \{\zeta_1 \sqsubseteq \zeta_2\}, \text{ where } \zeta_1, \zeta_2 \in \{\bot, \top\} \cup \mathcal{A}$$

$$\mathbf{cs}_{\leq}(\theta_1 \to \cdots \to \theta_n \to \eta, \theta'_1 \to \cdots \to \theta'_n \to \eta') = \{(\bot \text{ in } \mathbf{tail}(\eta')) \Rightarrow (\mathbf{cs}_{\leq}(\eta, \eta') \cup \textstyle\bigcup_{1 \leq i \leq n} \mathbf{cs}_{\leq}(\theta'_i, \theta_i))\},$$
*where $n \geq 1$ and $\eta, \eta'$ are not arrow s$\star$-patterns or arrow s$\star$-types*

$$\mathbf{cs}_{\leq}(\theta_1 \times \eta_1, \theta_2 \times \eta_2) = \mathbf{cs}_{\leq}(\theta_1, \theta_2) \cup \mathbf{cs}_{\leq}(\eta_1, \eta_2)$$

Figure 6.5: Function $\mathbf{cs}_{\leq}$

$$\mathbf{ucs}_{\leq}(\theta'_1 \to \cdots \to \theta'_n \to \eta', \theta''_1 \to \cdots \to \theta''_n \to \eta'') = \mathbf{ucs}_{\leq}(\eta', \eta'') \cup \textstyle\bigcup_{1 \leq i \leq n} \mathbf{cs}_{\leq}(\theta''_i, \theta'_i),$$
*where $n \geq 0$ and $\eta', \eta''$ are not arrow s$\star$-patterns or arrow s$\star$-types*

$$\mathbf{ucs}_{\leq}(\theta_1 \times \eta_1, \theta_2 \times \eta_2) = \mathbf{cs}_{\leq}(\theta_1, \theta_2) \cup \mathbf{cs}_{\leq}(\eta_1, \eta_2)$$

Figure 6.6: Function $\mathbf{ucs}_{\leq}$

$$\begin{array}{lll} \mathcal{W}^{\mathrm{s}\star}(P) = & \text{let} & \Theta = \mathbf{fresh}(\mathrm{FV}(P)) \\ & & \text{and } \langle \theta, \mathcal{E} \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta, P) \\ & \text{in} & \langle \Theta, \eta, \mathcal{E} \rangle \text{ end} \end{array}$$

Figure 6.7: Algorithm $\mathcal{W}^{\mathrm{s}\star}$

$$
\begin{array}{lll}
\mathcal{W}_0^{\mathrm{s}\star}(\Theta, P) = \text{case } P \text{ of} & & \\
\quad \mathsf{c}: & & \mathbf{sts}(\mathsf{c}) \\
\quad x: & \text{let} & \theta_1 = \Theta(x) \\
 & & \text{and } \theta_2 = \mathbf{fresh}(\epsilon(\theta_1)) \\
 & \text{in} & \langle \theta_2, \mathbf{ucs}_{\leq}(\theta_1, \theta_2) \rangle \ \text{ end} \\
\quad \lambda x^{\sigma}.M: & \text{let} & \theta = \mathbf{fresh}(\sigma) \\
 & & \text{and } \langle \eta, \mathcal{E} \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta \cup \{x : \theta\}, M) \\
 & \text{in} & \langle \theta \to \eta, \mathcal{E} \rangle \ \text{ end} \\
\quad MN: & \text{let} & \langle \theta_1 \to \eta, \mathcal{E}_1 \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta, M) \\
 & & \text{and } \langle \theta_2, \mathcal{E}_2 \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta, N) \\
 & \text{in} & \langle \eta, \{(\bot \,\mathsf{in}\, \mathbf{tail}(\eta)) \Rightarrow (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathbf{ucs}_{\leq}(\theta_2, \theta_1)\} \rangle \ \text{ end} \\
\quad \langle M_1, M_2 \rangle: & & \\
 & \text{let} & \langle \theta_1, \mathcal{E}_1 \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta, M_1) \\
 & & \text{and } \langle \theta_2, \mathcal{E}_2 \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta, M_2) \\
 & \text{in} & \langle \theta_1 \times \theta_2, \mathcal{E}_1 \cup \mathcal{E}_2 \rangle \ \text{ end} \\
\quad \mathsf{proj}_i M: & \text{let} & \langle \theta_1 \times \theta_2, \mathcal{E} \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta, M) \\
 & \text{in} & \langle \theta_i, \mathcal{E} \rangle \ \text{ end} \\
\quad \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2: & & \\
 & \text{let} & \langle \alpha, \mathcal{E}_0 \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta, N) \\
 & & \text{and } \langle \theta_1, \mathcal{E}_1 \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta, M_1) \\
 & & \text{and } \langle \theta_2, \mathcal{E}_2 \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta, M_2) \\
 & & \text{and } \eta = \mathbf{fresh}(\epsilon(\theta_1)) \quad (* \text{ Notice that } \epsilon(\theta_1) = \epsilon(\theta_2) *) \\
 & \text{in} & \langle \eta, \{(\mathcal{E}_0 \cup \{\alpha \sqsubseteq \bot\}) \bigvee (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathbf{ucs}_{\leq}(\theta_1, \eta) \cup \mathbf{ucs}_{\leq}(\theta_2, \eta))\} \rangle \ \text{ end} \\
\quad \mathsf{fix}\ x^{\rho}.M: & & \\
 & \text{let} & \theta = \mathbf{fresh}(\rho) \\
 & & \text{and } \langle \eta, \mathcal{E} \rangle = \mathcal{W}_0^{\mathrm{s}\star}(\Theta \cup \{x : \theta\}, M) \\
 & & \text{and } h = \mathbf{max}_{\mathbf{size}}(\rho) \\
 & & \text{and } \mathcal{S}(l) = \text{let} \\
 & & \qquad \mathbf{r}_l = \text{“a fresh renaming of the constructor variables not in } \Theta\text{”} \\
 & & \qquad \text{and } \theta_l = \mathbf{r}_l(\theta) \quad \text{and } \eta_l = \mathbf{r}_l(\eta) \quad \text{and } \mathcal{E}_l = \mathbf{r}_l(\mathcal{E}) \\
 & & \quad \text{in if } l = h \\
 & & \qquad \text{then } \mathcal{E}_h \cup \mathbf{ucs}_{\leq}(\eta_h, \theta_1) \cup \mathbf{ucs}_{\leq}(\eta_h, \eta) \\
 & & \qquad \text{else } \mathcal{E}_l \cup \{(\mathbf{ucs}_{\leq}(\eta_l, \theta_1) \cup \mathbf{ucs}_{\leq}(\eta_l, \eta)) \\
 & & \qquad\qquad \bigvee (\mathbf{ucs}_{\leq}(\theta_l, \eta_{l+1}) \cup \mathcal{S}(l+1))\} \\
 & & \quad \text{end} \\
 & \text{in} & \langle \eta, \mathcal{S}(1) \rangle \ \text{ end}
\end{array}
$$

Figure 6.8: Algorithm $\mathcal{W}_0^{\mathrm{s}\star}$

$$\mathbf{cs}_{=}(\zeta^{\iota}, \bot^{\iota}) = \{\zeta \sqsubseteq \bot\}$$
$$\mathbf{cs}_{=}(\zeta^{\iota}, \top^{\iota}) = \{\top \sqsubseteq \zeta\}$$

$$\mathbf{cs}_{=}(\theta \to \eta, \phi \to \psi) = \mathbf{cs}_{=}(\eta, \psi) \cup \mathbf{cs}_{=}(\theta, \phi)$$

$$\mathbf{cs}_{=}(\theta_1 \times \theta_2, \phi_1 \times \phi_2) = \mathbf{cs}_{=}(\theta_1, \phi_1) \cup \mathbf{cs}_{=}(\theta_2, \phi_2)$$

Figure 6.9: Function $\mathbf{cs}_{=}$

the number of premises that an algorithm has to consider while looking for a property of a fixpoint expression of type $\rho$.

Correctness and completeness of the inference are expressed by Theorem 6.21. As a consequence of completeness we have that, for every PCFP term $M$, $\mathcal{W}^{\mathrm{s}\star}(M)$ is a kind of principal s$\star$-type scheme for $M$, i.e. it can be used for generating all the pairs $\langle \Sigma, \phi \rangle$ such that $\Sigma \vdash^{\mathrm{g}}_{\mathrm{s}\star} M : \phi$.

**Theorem 6.21** *Let $\vdash_{\mathbf{T}} M : \rho$ and $\mathcal{W}^{\mathrm{s}\star}(M) = \langle \Theta, \theta, \mathcal{E} \rangle$.*

1. *For all instantiations $\mathbf{i}$, if $\mathbf{i} \in \mathbf{sat}(\mathcal{E})$, then $\mathbf{i}(\Theta) \vdash^{\mathrm{g}}_{\mathrm{s}\star} M : \mathbf{i}(\theta)$.*

2. *For all s$\star$-type assignment statements $\Sigma \vdash^{\mathrm{g}}_{\mathrm{s}\star} M : \phi$ such that $\epsilon(\Sigma) = \mathrm{FV}(M)$, there exists $\mathbf{i} \in \mathbf{sat}(\mathcal{E})$ such that $\mathbf{i}(\Theta) = \Sigma$ and $\mathbf{i}(\theta) = \phi$.* □

Let $M$ be a PCFP term of functional type $\rho$, the output of the s$\star$-type inference algorithm, $\mathcal{W}^{\mathrm{s}\star}(M) = \langle \Theta, \theta, \mathcal{E} \rangle$, can be used either to check if a function has a given s$\star$-type, or to detect the set of arguments in which the function can be proved to be strict. Such a set can be found by examining the s$\star$-types of the shape $\phi = \phi_1 \to \cdots \to \phi_n \to \psi_1 \to \psi_2$, where $\epsilon(\phi) = \rho$, $\psi_1, \psi_2 \in \mathbf{L}^{\mathrm{s}\star}_{\bot\mathrm{g}}$, $n \geq 0$ and, for all $i \in \{1, \ldots, n\}$, $\phi_i \in \mathbf{L}^{\mathrm{s}\star}_{\top}$. In the following we will refer to such s$\star$-types as *first order strictness properties*.

Given a s$\star$-pattern $\theta$ and an s$\star$-type $\phi$ such that $\epsilon(\theta) = \epsilon(\phi)$, $\mathbf{cs}_{=}(\theta, \phi)$ denotes the set of s$\star$-constraints defined according to the clauses in Fig. 6.9. It is easy to check that, for all instantiation $\mathbf{i}$, $\mathbf{i} \in \mathbf{cs}_{=}(\theta, \phi)$ if and only if $\mathbf{i}(\theta) = \phi$.

**Corollary 6.22** *Let $\vdash_{\mathbf{T}} M : \rho$ and $\mathcal{W}^{\mathrm{s}\star}(M) = \langle \Theta, \theta, \mathcal{E} \rangle$.*

1. *An s$\star$-type $\phi$ can be assigned to $M$ if and only if $\mathbf{sat}(\mathcal{E} \cup \mathbf{cs}_{=}(\theta, \phi)) \neq \emptyset$.*

2. *Let $\phi$ and $\theta$ be a first order strictness property and an s$\star$-pattern such that $\epsilon(\phi) = \epsilon(\theta) = \rho$. Then $\phi$ can be assigned to $M$ if and only if, for some maximal instantiation $\mathbf{i} \in \mathbf{sat}(\mathcal{E} \cup \{\alpha \sqsubseteq \bot \mid \alpha \in \mathbf{tail}(\theta)\})$, either $\mathbf{i}(\theta) \in \mathbf{L}^{\mathrm{s}\star}_{\bot\mathrm{g}}$, i.e. $M$ is an always divergent program, or $\mathbf{i}(\theta) \cong^{\mathrm{g}}_{\mathrm{s}\star} \phi$.* □

**Example 6.23** *Let $\vdash_{\mathbf{T}}$ twice : $\rho$ where $\rho = (\sigma \to \sigma) \to \sigma \to \sigma$ and* twice $= \lambda f^{\sigma \to \sigma}.\ \lambda x^{\sigma}.\ f(f\,x)$.

*We have that* $\mathcal{W}^{\mathrm{s}\star}(\text{twice}) = \langle \Theta, \theta, \mathcal{E} \rangle$*, where* $\theta = (\alpha_1{}^{\mathsf{nat}} \to \alpha_2{}^{\mathsf{nat}}) \to \beta^{\mathsf{nat}} \to \alpha_2'{}^{\mathsf{nat}}$ *and* $\mathcal{E}$ *is the first set of s⋆-constraints in Example 6.15.*

*Then* $\mathcal{E} \cup \mathbf{cs}_{=}(\alpha_2', \bot) \rightsquigarrow \{\mathcal{I}_2\}$*, where* $\mathcal{I}_2$ *is defined in Example 6.15 and represents the maximum instantiation,* $\mathbf{i}_2$*, in* $\mathbf{sat}(\mathcal{E} \cup \{\alpha_2' \sqsubseteq \bot\})$*. Since* $\mathbf{i}_2(\theta) = (\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$ *we have that* twice *is strict in its first argument. Moreover, since* $\mathbf{i}_2$ *is the maximum of* $\mathbf{sat}(\mathcal{E} \cup \{\alpha_2' \sqsubseteq \bot\})$ *and* $\mathbf{i}_2(\theta) \notin \mathbf{L}^{\mathrm{s}\star}_{\bot\mathrm{g}}$*, we have that (by Corollary 6.22) it is not possible to assign the property* $(\top^{\mathsf{nat}} \to \top^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$ *to* twice*, i.e.* twice *cannot be proved to be strict in the second argument.*

*The sets* $\mathbf{cs}_{=}(\theta, (\bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}})$ *and* $\mathbf{cs}_{=}(\theta, (\top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}})$ *are the sets* $\mathcal{E}_1$ *and* $\mathcal{E}_2$ *of Example 6.15. So* $\mathcal{E} \cup \mathcal{E}_1 \rightsquigarrow' \text{TRUE}$ *and* $\mathcal{E} \cup \mathcal{E}_2 \rightsquigarrow' \text{TRUE}$*.*

*Finally observe that the set* $\mathcal{E}_3$ *of Example 6.15 is* $\mathbf{cs}_{=}(\theta, (\top^{\mathsf{nat}} \to \top^{\mathsf{nat}}) \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}})$*. Obviously* $\mathcal{E} \cup \mathcal{E}_3 \rightsquigarrow' \text{FALSE}$*, since* twice *cannot be proved to be strict in the second argument.*

We conclude the section with an example of application of the algorithm involving a fixpoint.

**Example 6.24** *Let* $F = \mathsf{fix}\, f^{\rho}.M$*, where* $\rho = \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ *and*

$$M = \lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\lambda z^{\mathsf{nat}}.\mathsf{if}\ z = 0\ \mathsf{then}\ x + y\ \mathsf{else}\ fyx(z-1) \quad .$$

*It is easy to check that* $F$ *is strict in each of its 3 arguments, i.e.* $\emptyset \vdash^{\mathrm{g}}_{\mathrm{s}\star} F : \phi_1$*,* $\emptyset \vdash^{\mathrm{g}}_{\mathrm{s}\star} F : \phi_2$*, and* $\emptyset \vdash^{\mathrm{g}}_{\mathrm{s}\star} F : \phi_3$*, where* $\phi_1 = \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$*,* $\phi_2 = \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$*, and* $\phi_3 = \top^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \bot^{\mathsf{nat}} \to \bot^{\mathsf{nat}}$*. In particular as for* $\vdash_{\mathrm{s}}$ *(see Example 4.20) to derive* $\emptyset \vdash^{\mathrm{g}}_{\mathrm{s}\star} F : \phi_1$ *and* $\emptyset \vdash^{\mathrm{g}}_{\mathrm{s}\star} F : \phi_2$ *are needed two premises in the application of the rule* (Fix$^{\star}$)*, while to derive* $\emptyset \vdash^{\mathrm{g}}_{\mathrm{s}\star} F : \phi_3$ *one premise is sufficient.*

*It is easy to check that, for some* $\theta = \alpha_{1,0} \to \alpha_{2,0} \to \alpha_{3,0} \to \alpha_{4,0} = \mathbf{fresh}(\rho)$ *,* $\beta_{1,0} = \mathbf{fresh}(\mathsf{nat})$*,* $\beta_{2,0} = \mathbf{fresh}(\mathsf{nat})$*,* $\beta_{3,0} = \mathbf{fresh}(\mathsf{nat})$*, and* $\mathbf{sts}(=) = \langle \gamma_{1,0} \times \gamma_{2,0} \to \gamma_{3,0}, \{(\bot\, \mathsf{in}\, \{\gamma_{3,0}\}) \Rightarrow \{\{\gamma_{1,0} \sqsubseteq \bot\} \bigvee \{\gamma_{2,0} \sqsubseteq \bot\}\}\}\rangle$*, we have*

$$\mathcal{W}_0^{\mathrm{s}\star}(\{f : \theta,\ x : \beta_{1,0},\ y : \beta_{2,0},\ z : \beta_{3,0}\}, z = 0) = \langle\ \gamma_{3,0},\ \mathcal{E}_{3,0}\rangle\ ,$$

*where* $\mathcal{E}_{3,0} =$

$$\left\{\ \ (\bot\, \mathsf{in}\, \{\gamma_{3,0}\}) \Rightarrow\ \ \left\{\ \ \begin{array}{l} (\bot\, \mathsf{in}\, \{\gamma_{3,0}\}) \Rightarrow \{\{\gamma_{1,0} \sqsubseteq \bot\} \bigvee \{\gamma_{2,0} \sqsubseteq \bot\}\}, \\ \beta_{3,0} \sqsubseteq \beta'_{3,0}, \\ \beta'_{3,0} \sqsubseteq \gamma_{1,0}, \\ \top \sqsubseteq \gamma_{2,0} \end{array} \right\}\ \right\}\ .$$

*Then, if* $\mathbf{sts}(+) = \langle \beta_{4,0} \times \beta_{5,0} \to \beta_{6,0}, \{(\bot\, \mathsf{in}\, \{\beta_{6,0}\}) \Rightarrow \{\{\beta_{4,0} \sqsubseteq \bot\} \bigvee \{\beta_{5,0} \sqsubseteq \bot\}\}\}\rangle$*, we have*

$$\mathcal{W}_0^{\mathrm{s}\star}(\{f : \theta,\ x : \beta_{1,0},\ y : \beta_{2,0},\ z : \beta_{3,0}\}, x + y) = \langle\ \beta_{6,0},\ \mathcal{E}_{1,0}\rangle\ ,$$

*where* $\mathcal{E}_{1,0} =$

$$\{\ \ (\bot \,\mathsf{in}\, \{\beta_{6,0}\}) \Rightarrow\ \ \{\ \ (\bot \,\mathsf{in}\, \{\beta_{6,0}\}) \Rightarrow \{\{\beta_{4,0} \sqsubseteq \bot\} \bigvee \{\beta_{5,0} \sqsubseteq \bot\}\},\ \beta_{1,0} \sqsubseteq \beta'_{1,0},\ \beta'_{1,0} \sqsubseteq \beta_{4,0},\ \beta_{2,0} \sqsubseteq \beta'_{2,0},\ \beta'_{2,0} \sqsubseteq \beta_{5,0}\ \ \}\ \ \}\ .$$

*Moreover, if* $\mathbf{sts}(-) = \langle \beta_{7,0} \times \beta_{8,0} \rightarrow \beta_{9,0}, \{(\bot \,\mathsf{in}\, \{\beta_{9,0}\}) \Rightarrow \{\{\beta_{7,0} \sqsubseteq \bot\} \bigvee \{\beta_{8,0} \sqsubseteq \bot\}\}\}\rangle$*, we have*

$$\mathcal{W}_0^{\mathrm{s}\star}(\{f : \theta,\ x : \beta_{1,0},\ y : \beta_{2,0},\ z : \beta_{3,0}\}, fyx(z-1)) = \langle\ \alpha'_{4,0},\ \mathcal{E}_{2,0}\rangle\ ,$$

*where* $\mathcal{E}_{2,0} = \{(\bot \,\mathsf{in}\, \{\alpha'_{4,0}\}) \Rightarrow$

$$\{\ \ (\bot \,\mathsf{in}\, \{\alpha'_{4,0}\}) \Rightarrow \{\ \ (\bot \,\mathsf{in}\, \{\alpha'_{4,0}\}) \Rightarrow \{\ \ (\bot \,\mathsf{in}\, \{\alpha'_{4,0}\}) \Rightarrow \{\ \alpha'_{3,0} \sqsubseteq \alpha_{3,0},\ \alpha'_{2,0} \sqsubseteq \alpha_{2,0},\ \alpha'_{1,0} \sqsubseteq \alpha_{1,0}\ \ \},\ \beta_{2,0} \sqsubseteq \beta''_{2,0},\ \beta''_{2,0} \sqsubseteq \alpha'_{1,0}\ \ \},\ \beta_{1,0} \sqsubseteq \beta''_{1,0},\ \beta''_{1,0} \sqsubseteq \alpha'_{2,0}\ \ \},$$

$$(\bot \,\mathsf{in}\, \{\beta_{9,0}\}) \Rightarrow \{\ \ (\bot \,\mathsf{in}\, \{\beta_{9,0}\}) \Rightarrow \{\{\beta_{7,0} \sqsubseteq \bot\} \bigvee \{\beta_{8,0} \sqsubseteq \bot\}\},\ \beta_{3,0} \sqsubseteq \beta''_{3,0},\ \beta''_{3,0} \sqsubseteq \beta_{7,0},\ \top \sqsubseteq \beta_{8,0}\ \ \},$$

$$\beta_{9,0} \sqsubseteq \alpha'_{3,0}\ \ \}\}\ .$$

*Lastly we get*

$$\mathcal{W}_0^{\mathrm{s}\star}(\{f : \theta\}, M) = \langle \eta, \mathcal{E}\rangle$$

*where*

$$\begin{array}{rcl} \eta & = & \beta_{1,0} \rightarrow \beta_{2,0} \rightarrow \beta_{3,0} \rightarrow \gamma_0, \text{ and} \\ \mathcal{E} & = & \{\ (\mathcal{E}_{3,0} \cup \{\beta'_{3,0} \sqsubseteq \bot\}) \bigvee (\mathcal{E}_{1,0} \cup \mathcal{E}_{2,0} \cup \{\beta_{6,0} \sqsubseteq \gamma_0, \alpha'_{4,0} \sqsubseteq \gamma_0\})\ \}. \end{array}$$

*So, if we apply the s⋆-type inference algorithm* $\mathcal{W}^{\mathrm{s}\star}$ *(in Fig. 6.7), we get*

$$\mathcal{W}^{\mathrm{s}\star}(\emptyset, F) = \langle \emptyset, \eta, \mathcal{S}(1)\rangle$$

*where* $\mathcal{S}(1)$ *is as in the last branch of the "case" in Fig. 6.8 (*$h = \mathbf{max_{size}}(\rho) = 3$*) and, for* $l \in \{1, \ldots, 4\}$*,* $\mathcal{E}_l$*,* $\theta_l$*,* $\eta_l$ *are obtained from* $\mathcal{E}$*,* $\theta$*,* $\eta$ *by replacing all the occurrences of the index "0" by the index "l".*

*Since* $\phi_1$*,* $\phi_2$ *and* $\phi_3$ *are first order strictness properties and* $\top^{\mathsf{nat}} \rightarrow \top^{\mathsf{nat}} \rightarrow \top^{\mathsf{nat}} \rightarrow \bot^{\mathsf{nat}}$ *is not a property of* $F$*, we have that* $\mathbf{sat}(\mathcal{S}(1) \cup \mathbf{cs}_{=}(\gamma, \bot))$ *has three maximal elements,* $\mathbf{i}_1$*,* $\mathbf{i}_2$*, and* $\mathbf{i}_3$*, corresponding to the first order strictness property of* $F$*, i.e.* $\mathbf{i}_1(\eta) = \phi_1$*,* $\mathbf{i}_2(\eta) = \phi_2$*, and* $\mathbf{i}_3(\eta) = \phi_3$*. So, if* $\mathcal{I}_1$*,* $\mathcal{I}_2$*,* $\mathcal{I}_3$ *are the representations of* $\mathbf{i}_1$*,* $\mathbf{i}_2$*,* $\mathbf{i}_3$*, then* $\mathcal{S}(1) \cup \mathbf{cs}_{=}(\gamma, \bot) \rightsquigarrow \{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$*.*

*Obviously we also have* $\mathcal{S}(1) \cup \mathbf{cs}_{=}(\eta, \phi_1) \rightsquigarrow'$ TRUE*,* $\mathcal{S}(1) \cup \mathbf{cs}_{=}(\eta, \phi_2) \rightsquigarrow'$ TRUE*, and* $\mathcal{S}(1) \cup \mathbf{cs}_{=}(\eta, \phi_3) \rightsquigarrow'$ TRUE*.*

## 6.5 Final remarks

In this chapter we have presented an *inference* algorithm, i.e. an algorithm that given a PCFP term $M$ returns a kind of principal scheme representing (by means of a set of constraints) all the pairs (basis, s$\star$-type) for $M$. The extraction of the s$\star$-constraints from a PCFP program is done once. Then the set of s$\star$-constraints can be used to find all the s$\star$-types that can be assigned to $M$ by $\vdash^{\mathrm{g}}_{\mathrm{s}\star}$.

The system $\vdash^{\mathrm{g}}_{\mathrm{s}\star}$ and the algorithm $\mathcal{W}^{\mathrm{s}\star}$ are more powerful than the corresponding system and algorithm in the pioneering work [Kuo and Mishra, 1989]. This is due to the different rule for handling recursive terms (see Examples 4.23 and 6.24).

Generating guarded s$\star$-constraints for codifying uses of the rule ($\rightarrow \mathrm{E}^{\star}$) of Definition 6.4 (see the treatment of applications in Fig. 6.7) results in an effect similar to the lazy types optimization used for the checking algorithm in [Hankin and Le Métayer, 1994b]: if the properties of some subterms of the PCFP term in input are not relevant w.r.t. the s$\star$-types under examination, then the constraint associated to these not relevant subterms are not examined.

Indeed the algorithm $\mathcal{W}^{\mathrm{s}\star}$ generates sets of s$\star$-constraint containing a greater number of useless occurrences of the "guard" construct, but the generation of these useless "guards" can be easily avoided in a real implementation.

The s$\star$-constraints generated for the rule ($\mathrm{Fix}^{\star}$) codify all the different alternatives of number of premises needed to analyze a fixpoint. Each alternative is in turn codified by a set of s$\star$-constraints whose solutions represent all the possible uses of the rule with the corresponding number of premises.

Given a set of constraints generated by the algorithm $\mathcal{W}^{\mathrm{s}\star}$, the algorithm for checking whether the set of s$\star$-constraints is satisfiable (in Section 6.3.2) explores the various alternatives codifying the application of the rule ($\mathrm{Fix}^{\star}$) from the cheapest (i.e. 1 premise) to the most expensive (i.e. $\mathbf{max}_{\mathbf{size}}(\rho)$ premises, where $\rho$ is the type of the recursive term examined) stopping as soon as a success is reached.

At the moment we have no idea of how the inference algorithm for $\vdash^{\mathrm{g}}_{\mathrm{s}\star}$ could be extended to the conjunctive strictness analysis of Section 5.2.3.

# Part III

# Dead-code analysis

# An introduction to dead-code analysis

In this part of the thesis we present various non-standard type assignment systems and simplifications mappings for detecting and removing dead-code in PCFP$^{\mathcal{T}}$ programs. PCFP$^{\mathcal{T}}$ (the extension of PCFP presented in Section 2.6) is a lazy language, however (as we will explain in Section 7.8), all the dead-code detection and elimination techniques developed in the thesis can be straightforwardly applied to eager versions of PCFP$^{\mathcal{T}}$ (see Section 2.7).

A first motivation for dead-code analysis (also called liveness analysis) is the efficient implementation of functional programs (see for instance [Nielson, 1989], where a simple dead-code analysis is specified by abstract interpretation). In particular dead-code can arise in programs in which general-purpose functions, taken from standard libraries, are used in particular contexts. However dead-code analysis has been mainly studied in the context of *logical frameworks*, like Coq [Barras *et al.*, 1996], to remove redundant code from functional programs extracted from formal proofs (see [Paulin-Mohring, 1989b; Paulin-Mohring, 1989a] for an introduction to the subject). In fact, programs extracted from proofs usually contain large parts that are useless for the computation of the final result (i.e. dead-code) and some sort of simplification is mandatory. Various simplification techniques have been developed in the last few years, see for instance [Takayama, 1991; Berardi, 1996; Boerio, 1995; Prost, 1995].

The basic idea (inspired by the technique of [Berardi, 1993b; Berardi, 1996]) of the dead-code detection and elimination techniques presented in this part of the thesis is to start from a typed term and to decorate it by non-standard types (called *dead-code types*) that show the dead-code. To define dead-code types (see Section 3.2.3), for every ground type $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$, two basic properties are introduced:

- $\delta^\iota$, which is the property of terms of type $\iota$ whose values have a precise identity ("useful" terms), i.e. they could only be replaced by terms with the same behaviour, and

- $\top^\iota$, which is the property of terms of type $\iota$ whose values do not have a precise identity ("useless" terms), i.e. they could be replaced by any term of the same type.

The symbols $\delta$ and $\top$ are seen as unary operators, called *basic property constructors*. Dead-code types are constructed from these basic properties using the standard type constructors (see Fig. 3.1 in Section 3.1.1). For instance if a term of type $\mathsf{nat} \to \mathsf{nat}$ has the dead-code types $\delta^{\mathsf{nat}} \to \top^{\mathsf{nat}}$ or $\top^{\mathsf{nat}} \to \top^{\mathsf{nat}}$ then the whole term is assumed to be useless. The dead-code type $\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$, instead, is the property of all the terms of type $\mathsf{nat} \to \mathsf{nat}$ which yield a useful output whenever applied to an argument which is not used for the computation of this output (like $\lambda x^{\mathsf{nat}}.Q$ where $x$ does not occur in $Q$). In other words, $\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$ characterizes all the (necessary constant) functions of type $\mathsf{nat} \to \mathsf{nat}$ that do not use their argument. Finally, the dead-code type $\delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$ does not contain any information about dead-code.

To detect the dead-code in a given PCFP$^{\mathcal{T}}$ term $M$ we will assume that $M$ is a useful term (i.e. we force it to have a dead-code type containing only $\delta$'s) and try to assign the maximum amount of $\top$'s to its subterms (i.e. to detect as many useless subterms as possible). Let us look at a couple of simple examples, to give an idea of what we mean by dead-code detection and elimination.

**Example III.1** *Let $M = (\lambda x^{\mathsf{nat}}.3)P$ where $P$ is a term of type* $\mathsf{nat}$*. Since $x$ is never used in the body of the $\lambda$-abstraction $F = \lambda x^{\mathsf{nat}}.3$, we have that $F$ has the property $\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$. This implies that $P$ is not used in the computation of the value of $M$ and could be replaced by any term of the same type. In fact, in a call-by-name language, $M$ behaves like the term $M' = (\lambda x^{\mathsf{nat}}.3)\mathsf{d}$, where* $\mathsf{d}$ *is a placeholder for the dead-code removed.*

*Note that, in this case, it is indeed possible to remove the dead-code in a clever way, i.e., replacing $M$ with the body of the $\lambda$-abstraction (the constant 3).*

**Example III.2** *Take*

$$N = (\lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, Q_2\rangle)P$$

*for some term $Q_1$, $Q_2$, and $P$ of type* $\mathsf{nat}$ *such that $Q_1$ does not contain any occurrence of $x$. Since the projection* $\mathsf{proj}_1$ *returns the first component of the pair, the term $Q_2$ is dead-code. Moreover, since $Q_1$ does not contain any occurrence of $x$, the $\lambda$-abstraction $G = \lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, Q_2\rangle$ has the property $\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$. This implies that also $P$ is dead-code. So, in a call by name language, $M$ behaves like the term*

$$N' = (\lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, \mathsf{d}\rangle)\mathsf{d}.$$

*Also in this case it is possible to perform a better simplification, i.e., $N'$ can be replaced by the simpler term $Q_1$.*

# Chapter 7

# Automatic dead-code detection and elimination

The first section of this chapter introduces a language of dead-code types and its semantics. Section 7.2 presents a dead-code type assignment system and a code simplification based on dead-code type information, in particular it is showed that a term and its simplified version are ground observationally equivalent. In Section 7.3 it is presented a weaker dead-code type assignment system (which characterizes a proper subset of the dead-code characterized by the system in Section 7.2) with a code simplification that induces a deeper transformation on programs. In Section 7.4 we show how the best way of using the systems and the program transformations in Sections 7.2 and 7.3 is to compose them in a sequential way: first detecting and removing dead-code using the former, and then applying the latter to the result.

Section 7.5 presents some examples of dead-code elimination.

In Section 7.6 it is presented an algorithm for inferring a representation of all the dead-code typings of a term w.r.t. both the systems of Sections 7.2 and 7.3. In Section 7.7 we show how the algorithm of Section 7.6 can be used to build an incremental dead-code detection and elimination procedure.

Related work is considered in Section 7.9.

A preliminary version of the material presented in this chapter appeared in [Coppo *et al.*, 1996], where

- the language considered does not include the fix constructor (is terminating),
- the completeness result for the dead-code type entailment relation (Theorem 7.11) is not given, and
- the techniques described in Sections 7.11 and 7.11 are not presented.

## 7.1 Dead-code types

In this section we consider the language of dead-code types introduced in Section 3.2.3, i.e., we choose as sets of basic properties $\mathbf{B}^{\mathrm{d}}(\iota) = \{\delta^\iota, \top^\iota\}$ for $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$, and $\mathbf{B}^{\mathrm{d}}(\rho) = \emptyset$ for $\rho \in \mathbf{T} - \{\mathsf{nat}, \mathsf{bool}\}$.

**Definition 7.1 (Dead-code types)** *The language* $\mathbf{L}^{\mathrm{d}}$ *of* dead-code types *(*d-types *for short) is defined by the following grammar:* $\phi ::= a^\iota \mid \phi \to \phi \mid \phi \times \phi$, *where* $a \in \{\delta, \top\}$ *and* $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$.

Since d-types are properties of terms, in the following we will use the words d-type and property interchangeably.

The meaning of d-types is formalized by interpreting each d-type, $\phi$, as a p.e.r. over the interpretation of the type $\epsilon(\phi)$ in the model $\mathcal{M}^{\mathrm{gnd}}$ of Section 2.4.1, i.e., over the set of the ground observational equivalence classes of closed terms of type $\epsilon(\phi)$. The intuition behind the following p.e.r. semantics of d-types is that, for every ground type $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$,

- the interpretation of $\delta^\iota$, which is the property of being a term of type $\iota$ whose value has a precise identity, brings the information that an occurrence a term having the property $\delta^\iota$ could be only replaced with an observational equivalent term (so it can be regarded as expressing the fact that the value of the term is used in the computation),
- while the interpretation of $\top^\iota$, which is the property of being a term of type $\iota$ whose value has not a precise identity, brings the information that an occurrence of a term having the property $\top^\iota$ could be replaced by any term of type $\iota$ (so it can be regarded as expressing the fact that the value of the term is not used in the computation).

I.e., the p.e.r. semantics captures the informal meaning of $\delta^\iota$ by interpreting it as the diagonal p.e.r. over $\mathbf{I}(\iota)$ (which equates each element in $\mathbf{I}(\iota)$ only with itself), and captures the informal meaning of $\top^\iota$ by interpreting it as the trivial p.e.r. over $\mathbf{I}(\iota)$ (the one that equates all the elements in $\mathbf{I}(\iota)$).

**Definition 7.2 (Semantics of d-types)** *1. The interpretation* $[\![\phi]\!]$ *of a d-type* $\phi$ *is defined by:*

$$
\begin{aligned}
[\![\delta^\iota]\!] &= \{\langle [N], [N]\rangle \mid [N] \in \mathbf{I}(\iota)\} \\
[\![\top^\iota]\!] &= \mathbf{I}(\iota) \times \mathbf{I}(\iota) \\
[\![\phi \to \psi]\!] &= \{\langle [M], [N]\rangle \mid \forall \langle [P], [Q]\rangle \in [\![\phi]\!].\langle [MP], [NQ]\rangle \in [\![\psi]\!]\} \\
[\![\phi_1 \times \phi_2]\!] &= \{\langle [M], [N]\rangle \mid \forall i \in \{1, 2\}.\ \langle [\mathsf{proj}_i M], [\mathsf{proj}_i N]\rangle \in [\![\phi_i]\!]\}.
\end{aligned}
$$

*2. By* $\sim_\phi$ *we denote the relation between closed terms defined by:* $M \sim_\phi N$ *if and only if* $\langle [M], [N]\rangle \in [\![\phi]\!]$.

**Notation 7.3** *For every type $\rho$, let $\delta(\rho)$ be the d-type obtained by replacing each occurrence of any ground type $\iota$ by $\delta^\iota$, and similarly let $\top(\rho)$ be the d-type obtained by replacing each occurrence of any ground type $\iota$ by $\top^\iota$.*

**Proposition 7.4** *For all $\rho \in \mathbf{T}$, let $\mathcal{R}_\rho$ be the relation*

$$\{\langle M, N\rangle \mid \vdash M : \rho, \vdash N : \rho, \mathrm{FV}(M) = \mathrm{FV}(N) = \emptyset \text{ and } M \simeq^{\mathrm{gnd}}_{\mathrm{obs}} N\},$$

*then $\mathcal{R}_\rho = \sim_{\delta(\rho)}$.*

**Proof.** By induction on $\rho \in \mathbf{T}$. The case $\rho \in \{\mathsf{nat}, \mathsf{bool}\}$ is trivial.

Let $\rho = \rho_1 \to \rho_2$. By induction hypothesis both $\mathcal{R}_{\rho_1} = \sim_{\delta(\rho_1)}$ and $\mathcal{R}_{\rho_2} = \sim_{\delta(\rho_2)}$. By definition of $\sim_{\delta(\rho_1 \to \rho_2)}$ we have that

$(*)$ $P \sim_{\delta(\rho_1\to\rho_2)} Q$ if and only if, for all $M$ and $N$ such that $M \sim_{\delta(\rho_1)} N$, $P\,M \sim_{\delta(\rho_2)} Q\,N$.

We prove the result by showing that

1. $P \sim_{\delta(\rho_1\to\rho_2)} Q$ implies $P \simeq^{\mathrm{gnd}}_{\mathrm{obs}} Q$, and
2. $P \simeq^{\mathrm{gnd}}_{\mathrm{obs}} Q$ implies $P \sim_{\delta(\rho_1\to\rho_2)} Q$.

The first implication can be easily proved by contraposition. Let $P \not\simeq^{\mathrm{gnd}}_{\mathrm{obs}} Q$ then (by Theorem 2.8) $P \not\simeq Q$, so there would be an $M$ such that $P\,M \not\simeq Q\,M$ and therefore $P\,M \not\simeq^{\mathrm{gnd}}_{\mathrm{obs}} Q\,M$. So (by induction hypothesis) $P\,M \not\sim_{\delta(\rho_2)} Q\,M$, and therefore (from $(*)$) $P \not\sim_{\delta(\rho_1\to\rho_2)} Q$. For the second one, let $P \simeq^{\mathrm{gnd}}_{\mathrm{obs}} Q$. This implies that, for all $R$ and $S$ such that $R \simeq^{\mathrm{gnd}}_{\mathrm{obs}} S$, $P\,R \simeq^{\mathrm{gnd}}_{\mathrm{obs}} Q\,S$. So (by induction hypothesis and $(*)$) $P \sim_{\delta(\rho_1\to\rho_2)} Q$.

The case $\rho = \rho_1 \times \rho_2$ is analogous. □

**Corollary 7.5** *The indexed set family of relations $\sim_{\delta(\rho)}$ ($\rho \in \mathbf{T}$) is the PCFP$^{\mathcal{T}}$ bisimilarity.*

**Proof.** The results follows immediately from Proposition 7.4 since, by Theorem 2.8, the class of type indexed relations $\mathcal{R}_\rho$ ($\rho \in \mathbf{T}$) is the the PCFP$^{\mathcal{T}}$ bisimilarity. □

We now introduce two subclasses of d-types, the *$\delta$-dead-code types* (*$\delta$-d-types* for short) and the *$\top$-dead-code types* (*$\top$-d-types* ), which collect all the d-types characterizing the terms having values with and without a precise identity, respectively. The informal meaning of such d-types is reflected by their interpretation in the p.e.r. model. I.e., (see Proposition 7.7) if $\phi$ is a $\top$-d-type, then $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi)) \times \mathbf{I}(\epsilon(\phi))$ ($[\![\phi]\!]$ is the p.e.r. which relates all the elements of $\mathbf{I}(\epsilon(\phi))$) and, if $\phi$ is an $\delta$-d-type, then $[\![\phi]\!] = \{\langle [M], [M]\rangle \mid [M] \in \mathbf{I}(\epsilon(\phi))\}$ ($[\![\phi]\!]$ is the p.e.r. which relates each element of $\mathbf{I}(\epsilon(\phi))$ with itself).

**Definition 7.6 ($\delta$-d-types and $\top$-d-types)** *1. The set $\mathbf{L}^{\mathrm{d}}_\delta$ of $\delta$-d-types is the subset of $\mathbf{L}^{\mathrm{d}}$ containing only the basic property constructor $\delta$, i.e., $\mathbf{L}^{\mathrm{d}}_\delta = \{\delta(\rho) \mid \rho \in \mathbf{T}\}$.*

2. *The set* $\mathbf{L}^{\mathrm{d}}_{\top}$ *of* $\top$*-d-types is inductively defined by:*

   - $\top^{\iota} \in \mathbf{L}^{\mathrm{d}}_{\top}$*, for* $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$
   - $\phi \to \psi \in \mathbf{L}^{\mathrm{d}}_{\top}$*, for for* $\phi \in \mathbf{L}^{\mathrm{d}}$ *and* $\psi \in \mathbf{L}^{\mathrm{d}}_{\top}$
   - $\phi \times \psi \in \mathbf{L}^{\mathrm{d}}_{\top}$*, for* $\phi, \psi \in \mathbf{L}^{\mathrm{d}}_{\top}$*.*

For instance $\top^{\mathsf{bool}}$, $\delta^{\mathsf{nat}} \to \top^{\mathsf{nat}}$, $\top^{\mathsf{nat}} \to \top^{\mathsf{nat}}$, and $(\delta^{\mathsf{nat}} \times \delta^{\mathsf{nat}}) \to (\top^{\mathsf{nat}} \times (\delta^{\mathsf{nat}} \to \top^{\mathsf{nat}}))$, are $\top$-d-types. While $\top^{\mathsf{nat}} \times \delta^{\mathsf{nat}}$ and $(\top^{\mathsf{nat}} \times \top^{\mathsf{nat}}) \to (\top^{\mathsf{nat}} \times (\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}))$ are not $\top$-d-types. Of course, for any type $\rho$, $\top(\rho)$ is a $\top$-d-type.

**Proposition 7.7** *1.* $\phi \in \mathbf{L}^{\mathrm{d}}_{\top}$ *if and only if* $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi)) \times \mathbf{I}(\epsilon(\phi))$*, and*

*2.* $\phi \in \mathbf{L}^{\mathrm{d}}_{\delta}$ *imply* $[\![\phi]\!] = \{\langle [M], [M] \rangle \mid [M] \in \mathbf{I}(\epsilon(\phi))\}$*.*

**Proof.** Point 1 follows by induction on d-types from Definition 7.2. Point 2 derives directly from Proposition 7.4. □

**Remark 7.8** *Also the reverse implication of Proposition 7.7.2 above holds, i.e. we have that*

$$\phi \in \mathbf{L}^{\mathrm{d}}_{\delta} \text{ if and only if } [\![\phi]\!] = \{\langle [M], [M] \rangle \mid [M] \in \mathbf{I}(\epsilon(\phi))\}.$$

*This can be easily be proved using Theorem 7.11.*

## 7.2 Detecting and removing dead-code I

In this section we carry on the dead-code analysis sketched in Section 3.2.3. In particular, we not only introduce an entailment relation (in Section 7.2.1) and an assignment system (in Section 7.2.2), but we also consider the problem of simplifying a program by removing the dead-code detected by the assignment system (Section 7.2.3).

### 7.2.1 An entailment relation for dead-code types

**Definition 7.9 (Entailment relation $\leq_1$)** *Let* $\phi, \psi \in \mathbf{L}^{\mathrm{d}}$*. We write* $\phi \leq_1 \psi$ *to mean that* $\phi \leq \psi$ *is derivable by the rules in Fig. 7.1. By* $\cong_1$ *we denote the equivalence relation induced by* $\leq_1$*.*

The intuition behind the entailment relation $\leq_1$ is that it corresponds to loss of information, i.e., it represents a logical implication between properties: $\phi_1 \leq_1 \phi_2$ means that $\phi_2$ is implied by $\phi_1$, so it cannot contain more information than $\phi_1$. For instance, consider the entailment $\delta^{\mathsf{nat}} \leq_1 \top^{\mathsf{nat}}$, which is straightforwardly derivable from rule $(\top)$. The intuitive meaning of this entailment is the following: we have that (as explained after Definition 7.2) $\delta^{\mathsf{nat}}$ express the fact of being a term of type $\mathsf{nat}$ with a precise value, while $\top^{\mathsf{nat}}$ express just the fact of being a term of type $\mathsf{nat}$ (its precise value does not matter), so $\top^{\mathsf{nat}}$ brings less information than $\delta^{\mathsf{nat}}$. Let us consider a couple of examples involving the arrow constructor.

$$\text{(Ref)}\ \phi \leq \phi \qquad (\top)\ \frac{\psi \in \mathbf{L}^{\mathrm{d}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}$$

$$(\rightarrow)\ \frac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_2 \rightarrow \psi_1 \leq \phi_1 \rightarrow \psi_2} \qquad (\times)\ \frac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_1 \times \psi_1 \leq \phi_2 \times \psi_2}$$

Figure 7.1: Entailment rules for d-types (system $\leq_1$)

- If a function has d-type $\delta^{\mathsf{nat}} \rightarrow \delta^{\mathsf{nat}}$, then it maps a precise value of type $\mathsf{nat}$ to a precise value of type $\mathsf{nat}$. From rule $(\top)$ we can derive $\delta^{\mathsf{nat}} \rightarrow \delta^{\mathsf{nat}} \leq_1 \delta^{\mathsf{nat}} \rightarrow \top^{\mathsf{nat}}$ and $\delta^{\mathsf{nat}} \rightarrow \delta^{\mathsf{nat}} \leq_1 \top^{\mathsf{nat}} \rightarrow \top^{\mathsf{nat}}$. Both these entailments fits well with the intuition given above, since being a function which gives back a precise value of type $\mathsf{nat}$ is more informative that being a function returning any value of type $\mathsf{nat}$.

- If a function has d-type $\top^{\mathsf{nat}} \rightarrow \delta^{\mathsf{nat}}$, then it maps any value of type $\mathsf{nat}$ to a precise value of type $\mathsf{nat}$. From rule $(\rightarrow)$ we can derive $\top^{\mathsf{nat}} \rightarrow \delta^{\mathsf{nat}} \leq_1 \delta^{\mathsf{nat}} \rightarrow \delta^{\mathsf{nat}}$. Also this entailment fits with the given intuition, since it is clear that a function mapping any value to a precise value also maps a precise value to a precise value.

It is immediate to show that $\leq_1$ is reflexive and transitive, since the rule (Trans) of Fig. 3.4 is admissible. Notice that, if $\phi$ and $\psi$ are $\top$-d-types such that $\epsilon(\phi) = \epsilon(\psi)$, then $\phi \cong_1 \psi$.

The $\leq_1$ entailment relation between d-types is sound and complete w.r.t. the p.e.r. interpretation of Definition 7.2. To our knowledge the completeness result is new.

**Theorem 7.10 (Soundness of $\leq_1$)** *$\phi \leq_1 \psi$ implies $[\![\phi]\!] \subseteq [\![\psi]\!]$.*

**Proof.** By induction on the derivation of the judgement $\phi \leq_1 \psi$. □

**Theorem 7.11 (Completeness of $\leq_1$)** *$[\![\phi]\!] \subseteq [\![\psi]\!]$ implies $\phi \leq_1 \psi$.* □

To prove completeness of the entailment we define, for each d-type $\phi$, two finite sets: $\mathbf{Extr}(\phi)$ and $U^{\phi}$, that in some sense characterize the set $[\![\phi]\!]$. The underlying idea is that

- $U^{\phi}$ is a set of pairs of closed PCFP$^{\mathcal{T}}$ terms such that: $\langle P, Q\rangle \in U^{\phi}$ implies $\langle [P], [Q]\rangle \in [\![\phi]\!]$, and

- $\mathbf{Extr}(\phi)$ is a set of pairs $\langle E, \overrightarrow{\psi}\rangle$ where $E$ is a closed PCFP$^{\mathcal{T}}$ term, and $\overrightarrow{\psi}$ is a sequence of $n \geq 0$ d-types $\phi_1 \cdots \phi_n$. Each pair $\langle E, \phi_1 \cdots \phi_n\rangle \in \mathbf{Extr}(\phi)$ identifies an occurrence of $\delta^{\iota}$ ($\iota \in \{\mathsf{nat}, \mathsf{bool}\}$) in $\phi$ that does *not* occur in the left-hand side of any arrow operator, and therefore associated to a "component" of ground type of the final result produced by the evaluation of a term of type $\epsilon(\phi)$ (in a suitable context of ground type). In particular $E$ is such that, whenever applied to a term

- If $\phi \in \mathbf{L}^{\mathrm{d}}_{\top}$, then $\mathbf{Extr}(\phi) = \emptyset$
- If $\phi = \phi_1 \to \cdots \to \phi_n \to \delta^\iota$, $n \geq 0$, then $\mathbf{Extr}(\phi) = \{\langle \lambda x^{\epsilon(\phi)}.x, \phi_1 \cdots \phi_n \rangle\}$
- If $\phi = \phi_1 \to \cdots \to \phi_n \to \psi_1 \times \psi_2$, $n \geq 0$, then $\mathbf{Extr}(\phi) =$

$$\bigcup_{i\in\{1,2\}}\{\ \langle \lambda x^{\epsilon(\phi)}.\lambda y_1^{\epsilon(\phi_1)}.\cdots\lambda y_n^{\epsilon(\phi_n)}.E(\mathsf{proj}_i(xy_1\cdots y_n)), \phi_1\cdots\phi_n\,\overrightarrow{\psi}\rangle \mid \langle E, \overrightarrow{\psi}\rangle \in \mathbf{Extr}(\psi_i)\ \}$$

Figure 7.2: Set $\mathbf{Extr}(\phi)$

$M$ of type $\epsilon(\phi)$ and to $n$ terms $N_i$ of type $\epsilon(\phi_i)$ $(1 \leq i \leq n)$, returns a value of type $\iota$ that corresponds to the "component" of the result of the evaluation of $MN_1 \cdots N_n$ (in a suitable context of ground type) determined by the occurrence of $\delta^\iota$ associated to the pair.

Both $\mathbf{Extr}(\phi)$ and $U^\phi$ are defined by structural induction on $\phi$.

**Definition 7.12** *Let $\phi$ be a d-type. The set of pairs $\mathbf{Extr}(\phi)$ is defined according to the clauses in Fig. 7.2.*

**Example 7.13** *Let $\phi = (\delta^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \delta^{\mathsf{bool}}) \times (\top^{\mathsf{nat}} \to (\delta^{\mathsf{nat}} \times \top^{\mathsf{nat}}))$. $\mathbf{Extr}(\phi)$ is the set*

$$\{\langle \lambda x^{\epsilon(\phi)}.(\lambda x'^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.x')(\mathsf{proj}_1 x), \delta^{\mathsf{nat}}\top^{\mathsf{nat}}\rangle\}$$
$$\cup\{\langle \lambda x^{\epsilon(\phi)}.(\lambda x'^{\mathsf{nat}\to(\mathsf{nat}\times\mathsf{nat})}.\lambda y_1^{\mathsf{nat}}.(\lambda x''^{\mathsf{nat}}.x'')(\mathsf{proj}_1(x'y_1)))(\mathsf{proj}_2 x), \top^{\mathsf{nat}}\rangle\}.$$

The set $\mathbf{Extr}(\phi)$ characterizes $\phi$ in the following sense.

**Proposition 7.14** *Let $\phi \in \mathbf{L}^{\mathrm{d}}$. $\langle [P], [Q]\rangle \in [\![\phi]\!]$ if and only if, for all $\langle E, \psi_1 \cdots \psi_n\rangle \in \mathbf{Extr}(\phi)$ and for all $\langle [M_i], [N_i]\rangle \in [\![\psi_i]\!]$ $(1 \leq i \leq n)$,*

$$EPM_1 \cdots M_n \simeq^{\mathrm{gnd}}_{\mathrm{obs}} EQN_1 \cdots N_n.$$

**Proof.** The case $\phi \in \mathbf{L}^{\mathrm{d}}_{\top}$ is immediate. If $\phi \notin \mathbf{L}^{\mathrm{d}}_{\top}$ the proof is by structural induction on $\phi$.

Let $\phi = \phi_1 \to \cdots \phi_n \to \delta^\iota$, $n \geq 0$. We have (by Definition 7.12) $\mathbf{Extr}(\phi) = \{\langle \lambda x^{\epsilon(\phi)}.x, \phi_1 \cdots \phi_n\rangle\}$, and (by Definition 7.2) $\langle [P], [Q]\rangle \in [\![\phi]\!]$ if and only if, for all $\langle [M_i], [N_i]\rangle \in [\![\phi_i]\!]$ $(1 \leq i \leq n)$, $\langle [P\,M_1 \cdots M_n], [Q\,N_1 \cdots N_n]\rangle \in [\![\delta^\iota]\!]$. But this is if and only if

$$(\lambda x^{\epsilon(\phi)}.x)P\,M_1 \cdots M_n \simeq^{\mathrm{gnd}}_{\mathrm{obs}} (\lambda x^{\epsilon(\phi)}.x)Q\,N_1 \cdots N_n.$$

Let $\phi = \phi_1 \to \cdots \phi_n \to \psi_1 \times \psi_2$, $n \geq 0$. We have (by Definition 7.2) $\langle [P], [Q]\rangle \in [\![\phi]\!]$ if and only if, for all for all $\langle [M_i], [N_i]\rangle \in [\![\phi_i]\!]$ $(1 \leq i \leq n)$, $\langle [P\,M_1 \cdots M_n], [Q\,N_1 \cdots N_n]\rangle \in$

- If $\phi = \phi_1 \to \cdots \to \phi_n \to \top^\iota$, $n \geq 0$, then
  $U^\phi = \{\langle \lambda x_1^{\epsilon(\phi_1)} \cdots x_n^{\epsilon(\phi_n)}.\iota(0), \lambda x_1^{\epsilon(\phi_1)} \cdots x_n^{\epsilon(\phi_n)}.\iota(1)\rangle\}$
- If $\phi = \delta^\iota$, then
  $U^\phi = \{\langle \iota(0), \iota(0)\rangle\}$
- If $\phi = \phi_1 \to \cdots \to \phi_n \to \delta^\iota$, $n \geq 1$, then $U^\phi =$

$$\cup_{i\in\{1,\ldots,n\}}\{\langle \lambda x_1^{\epsilon(\phi_1)} \cdots x_n^{\epsilon(\phi_n)}.\iota(Ex_i M_1 \cdots M_k), \lambda x_1^{\epsilon(\phi_1)} \cdots x_n^{\epsilon(\phi_n)}.\iota(Ex_i N_1 \cdots N_k)\rangle \mid \langle E, \phi'_1 \cdots \phi'_k\rangle \in \mathbf{Extr}(\phi_i) \text{ and, for all } j \in \{1,\ldots,k\}, \langle M_j, N_j\rangle \in U^{\phi'_j}\}$$

- If $\phi = \phi_1 \to \cdots \to \phi_n \to \psi \times \psi'$, $n \geq 0$, then $U^\phi =$

$$\{\ \langle \lambda x_1 \cdots x_n.\langle P\, x_1 \cdots x_n, P'\, x_1 \cdots x_n\rangle, \lambda x_1 \cdots x_n.\langle Q\, x_1 \cdots x_n, Q'\, x_1 \cdots x_n\rangle\rangle \mid \langle P, Q\rangle \in U^{\phi_1\to\cdots\to\phi_n\to\psi} \text{ and } \langle P', Q'\rangle \in U^{\phi_1\to\cdots\to\phi_n\to\psi'}\ \}$$

Figure 7.3: Set $U^\phi$

$[\![\psi_1 \times \psi_2]\!]$. Moreover (by Definition 7.2 again) $\langle [PM_1 \cdots M_n], [QN_1 \cdots N_n]\rangle \in [\![\psi_1 \times \psi_2]\!]$ if and only if both $[\mathsf{proj}_1(P\, M_1 \cdots M_n)] \in [\![\psi_1]\!]$ and $[\mathsf{proj}_2(Q\, N_1 \cdots N_n)] \in [\![\psi_2]\!]$. So the the result follows by induction hypotheses and Definition 7.12. □

**Notation 7.15** *For all $\iota_1, \iota_2 \in \{\mathsf{nat}, \mathsf{bool}\}$ and for all closed PCFP$^{\mathcal{T}}$ term $M$ of type $\iota_1$ let $\iota_2(M)$ denote the closed term (representing a coercion of $M$ to the type $\iota_2$) defined as follows*

$$\iota_2(M) = \begin{cases} M & \text{if } \iota_1 = \iota_2 \\ \mathsf{if}\ M = 0\ \mathsf{then\ false\ else\ true} & \text{if } \iota_1 = \mathsf{nat} \text{ and } \iota_2 = \mathsf{bool} \\ \mathsf{if}\ M\ \mathsf{then}\ 1\ \mathsf{else}\ 0 & \text{if } \iota_1 = \mathsf{bool} \text{ and } \iota_2 = \mathsf{nat}. \end{cases}$$

The notation $\iota(M)$ is used, in the following Definition 7.16, to allow to formulate the first three clauses in Fig. 7.3 independently from the fact that $\iota$ be nat or bool.

The set of characteristic pairs $U^\phi$ is defined by induction on $\phi$.

**Definition 7.16** *Let $\phi$ be a d-type. The set of pairs of closed terms $U^\phi$ is defined according to the clauses in Fig. 7.3.*

The following proposition can be easily proved by induction on types.

**Proposition 7.17** $\{\langle [P], [Q]\rangle \mid \langle P, Q\rangle \in U^\phi\} \subseteq [\![\phi]\!]$. □

The proof of completeness rely on the following lemma.

**Lemma 7.18** *$\phi \not\leq_1 \psi$ implies that there are $\langle P, Q\rangle \in U^\phi$ and $\langle E, \psi_1 \cdots \psi_n\rangle \in \mathbf{Extr}(\psi)$, such that for some $\langle M_1, N_1\rangle \in U^{\psi_1}$, ..., $\langle M_n, N_n\rangle \in U^{\psi_n}$,*

$$EPM_1 \cdots M_n \not\approx^{\mathrm{gnd}}_{\mathrm{obs}} EQN_1 \cdots N_n.$$

**Proof.** By structural induction on $\phi$.

Let $\underline{\phi = \phi_1 \to \cdots \to \phi_n \to \top^\iota}$. Then (since $\phi \not\leq_1 \psi$) it must be that $\psi = \phi_1 \to \cdots \to \phi_n \to \delta^\iota$. We have

- $U^\phi = \{\langle P, Q\rangle\}$, where $P = \lambda x_1^{\epsilon(\phi_1)} \cdots x_n^{\epsilon(\phi_n)}.\iota(0)$ and $Q = \lambda x_1^{\epsilon(\phi_1)} \cdots x_n^{\epsilon(\phi_n)}.\iota(1)$, and
- $\mathbf{Extr}(\psi) = \{\langle E, \psi_1 \cdots \psi_n\rangle\}$, where $E = \lambda x^{\epsilon(\phi)}.x$.

So, for all $\langle M_i, N_i\rangle \in U^{\psi_i}$ $(1 \leq i \leq n)$, $EPM_1 \cdots M_n \not\approx^{\mathrm{gnd}}_{\mathrm{obs}} EQN_1 \cdots N_n$.

Let $\underline{\phi = \phi_1 \to \cdots \to \phi_n \to \delta^\iota}$, $n \geq 0$. Then (since $\phi \not\leq_1 \psi$) it must be that $n \geq 1$, $\psi = \psi_1 \to \cdots \to \psi_n \to \delta^\iota$, and, for some $j \in \{1, \ldots, n\}$, $\psi_j \not\leq_1 \phi_j$. By induction hypothesis there are $\langle P', Q'\rangle \in U^{\psi_j}$ and $\langle E', \phi'_1 \cdots \phi'_k\rangle \in \mathbf{Extr}(\phi_i)$ such that, for some $\langle M'_1, N'_1\rangle \in U^{\phi'_1}$, ..., $\langle M'_k, N'_k\rangle \in U^{\phi'_k}$,

$$E'P'M'_1 \cdots M'_k \not\approx^{\mathrm{gnd}}_{\mathrm{obs}} E'Q'N'_1 \cdots N'_k.$$

Let $P = \lambda x_1^{\epsilon(\phi_1)} \cdots x_n^{\epsilon(\phi_n)}.E'x_iM'_1 \cdots M'_k$ and $Q = \lambda x_1^{\epsilon(\phi_1)} \cdots x_n^{\epsilon(\phi_n)}.E'x_iN'_1 \cdots N'_k$. It is immediate to see that $\langle P, Q\rangle \in U^\phi$. But $\mathbf{Extr}(\psi) = \langle \lambda x^{\epsilon(\psi)}.x, \psi_1 \cdots \psi_n\rangle$, so, for any $\langle M_i, N_i\rangle \in U^{\psi_i}$ $(1 \leq i \neq j \leq n)$, we have that

$$(\lambda x^{\epsilon(\psi).x)} PM_1 \cdots M_{j-1}P'M_{j+1} \cdots M_n \not\approx^{\mathrm{gnd}}_{\mathrm{obs}} (\lambda x^{\epsilon(\psi).x)} QN_1 \cdots N_{j-1}Q'N_{j+1} \cdots N_n,$$

which proves the result.

Let $\underline{\phi = \phi_1 \to \cdots \to \phi_n \to \phi' \times \phi''}$, $n \geq 0$. Then (since $\phi \not\leq_1 \psi$) it must be that $\psi = \psi_1 \to \cdots \to \psi_n \to \psi' \times \psi''$, and

1. either $\phi_1 \to \cdots \to \phi_n \to \phi' \not\leq_1 \psi_1 \to \cdots \to \psi_n \to \psi'$,
2. or $\phi_1 \to \cdots \to \phi_n \to \phi'' \not\leq_1 \psi_1 \to \cdots \to \psi_n \to \psi''$.

Assume that condition 1 holds (a similar proof can be given if 2 holds). By inductive hypothesis there are $\langle P', Q'\rangle \in U^{\phi_1 \to \cdots \to \phi_n \to \phi'}$ and $\langle E', \psi_1 \cdots \psi_n \psi'_1 \cdots \psi'_k\rangle \in \mathbf{Extr}(\psi_1 \to \cdots \to \psi_n \to \psi')$ $(n, k \geq 0)$, such that for some $\langle M_i, N_i\rangle \in U^{\psi_i}$ $(1 \leq i \leq n)$ and $\langle M'_j, N'_j\rangle \in U^{\psi'_j}$ $(1 \leq j \leq k)$

$$E'P'M_1 \cdots M_nM'_1 \cdots M'_k \not\approx^{\mathrm{gnd}}_{\mathrm{obs}} E'Q'N_1 \cdots N_nN'_1 \cdots N'_k.$$

Let

- $P = \lambda x_1^{\epsilon(\psi_1)} \cdots x_n^{\epsilon(\psi_n)}.\langle (P'x_1 \cdots x_n), (P''x_1 \cdots x_n)\rangle$, and
- $Q = \lambda x_1^{\epsilon(\psi_1)} \cdots x_n^{\epsilon(\psi_n)}.\langle (Q'x_1 \cdots x_n), (Q''x_1 \cdots x_n)\rangle$,

for some $\langle P'', Q'' \rangle \in U^{\phi_1 \to \cdots \to \phi_n \to \phi''}$. It is easy to check that $\langle P, Q \rangle \in U^{\phi_1 \to \cdots \to \phi_n \to \phi'}$. Let

$$\langle E, \overrightarrow{\psi} \rangle = \langle \lambda x^{\epsilon(\psi)}.\lambda y_1^{\epsilon(\psi_1)}.\cdots\lambda y_n^{\epsilon(\psi_n)}.E'(\mathsf{proj}_1(xy_1\cdots y_n)), \psi_1 \cdots \psi_n \psi'_1 \cdots \psi'_k \rangle,$$

it is easy to check that $\langle E, \overrightarrow{\psi} \rangle \in \mathbf{Extr}(\psi)$. But

$$\begin{array}{lll} EPM_1\cdots M_nM'_1\cdots M'_k & \simeq^{\mathrm{gnd}}_{\mathrm{obs}} & E'P'M_1\cdots M_nM'_1\cdots M'_k, \text{ and} \\ EQN_1\cdots N_nN'_1\cdots N'_k & \simeq^{\mathrm{gnd}}_{\mathrm{obs}} & E'Q'N_1\cdots N_nN'_1\cdots N'_k, \end{array}$$

so $EPM_1\cdots M_nM'_1\cdots M'_k \not\simeq^{\mathrm{gnd}}_{\mathrm{obs}} EQN_1\cdots N_nN'_1\cdots N'_k$, and the result is proved. □

From the previous proposition and lemma we derive the completeness result.

**Proof of Theorem 7.11.** By contraposition. Assume $\phi \not\leq_1 \psi$. By Lemma 7.18 there are $\langle P, Q \rangle \in U^\phi$ and $\langle E, \psi_1 \cdots \psi_n \rangle \in \mathbf{Extr}(\psi)$, such that for some $\langle M_1, N_1 \rangle \in U^{\psi_1}$, ..., $\langle M_n, N_n \rangle \in U^{\psi_n}$,

$$EPM_1\cdots M_n \not\simeq^{\mathrm{gnd}}_{\mathrm{obs}} EQN_1\cdots N_n.$$

But (from Proposition 7.17) $\langle [M_i], [N_i] \rangle \in [\![\psi_i]\!]$ $(1 \leq i \leq n)$, which implies (by Proposition 7.14) $\langle P, Q \rangle \notin [\![\psi]\!]$. Since from Proposition 7.17 $\langle P, Q \rangle \in [\![\phi]\!]$, this implies that $[\![\phi]\!] \not\subseteq [\![\psi]\!]$. □

**Remark 7.19** *It is easy to see that, for every d-type $\phi$, the set $U^\phi$ characterizes $\phi$, in the sense that:*

$$\phi \leq_1 \psi \text{ if and only if } \{\langle [P], [Q] \rangle \mid \langle P, Q \rangle \in U^\phi\} \subseteq [\![\psi]\!].$$

*The* (only if) *part is immediate from Proposition 7.17 by Theorem 7.10 (soundness of $\leq_1$). For the* (if) *one, use the argument in the proof of Theorem 7.11 above.*

### 7.2.2 An assignment system for detecting dead-code

D-types are assigned to PCFP$^{\mathcal{T}}$ terms by a set of inference rules. The functions $\delta(\cdot)$ and $\top(\cdot)$ (defined in Section 7.1) are extended to basis. More precisely: for any finite set $\Gamma$ of term variables, $\delta(\Gamma)$ and $\top(\Gamma)$ denote respectively the basis $\{x^\sigma : \delta(\sigma) \mid x^\sigma \in \Gamma\}$ and $\{x^\sigma : \top(\sigma) \mid x^\sigma \in \Gamma\}$.

The d-type assignment system of Definition 7.20 is not the one obtained by a straightforward instantiation the framework of Section 3.1, but it has been obtained from it after some "tuning". The differences can be summarized as following.

1. We have considered judgements of the form $\Sigma \vdash_1 M^\phi$ where $M^\phi$ is a *decorated* term, i.e., it has written in it the d-types assigned to its subterms. As explained at the end of Section 3.2.3, this notation allows to find the subterms to which are assigned $\top$-d-types (i.e., proved to be dead-code) just by inspecting the final judgement of a derivation, while finding the dead-code with the original formulation of the judgements, $\Sigma \vdash M : \phi$, requires the inspection of the derivation tree.

2. We have given a syntax-directed assignment system, which is immediately suitable for the definition of a d-type inference algorithm (in Section 7.6). The syntax-directed system (in Fig. 7.4) has been obtained from the original one by removing explicit rule for d-type entailment (the rule ($\leq$) of Fig. 3.4) and by embedding the use of entailment in rules (Var), (Con), ($\to$ E), (If), (Case), (It) and (Rec). This transformation has not reduced the power of the system (i.e., both the original system and the syntax-directed one allow to find the same dead-code).

3. For each constant $\mathsf{c}$ it is specified a minimal d-type (*not a set*) $\mathbf{L}^{\mathrm{d}}(\mathsf{c})$ such that $\epsilon(\mathbf{L}^{\mathrm{d}}(\mathsf{c})) = \mathbf{T}(\mathsf{c})$. The idea is that every d-type which is entailed ($\leq_1$) by $\mathbf{L}^{\mathrm{d}}(\mathsf{c})$ can be assigned to $\mathsf{c}$. For example, for all integers $\mathsf{n}$, $\mathbf{L}^{\mathrm{d}}(\mathsf{n}) = \delta^{\mathsf{nat}}$ and $\mathbf{L}^{\mathrm{d}}(+) = \delta^{\mathsf{nat}} \times \delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$.

**Definition 7.20 (D-type assignment system $\vdash_1$)** *A $\vdash_1$-typing statement is an expression $\Sigma \vdash_1 M^\phi$ where $\Sigma$ is a basis containing an assumption for each free variable of $M$. We write $\Sigma \vdash_1 M^\phi$ to mean that $\Sigma \vdash M^\phi$ can be derived by the rules in Fig. 7.4 where $\leq$ stands for $\leq_1$.*

**Remark 7.21** *1. Due to PCFP$^{\mathcal{T}}$ evaluation rules, for any PCFP$^{\mathcal{T}}$ constant $\mathsf{c}$ of type $\rho$, the $\delta$-d-type $\delta(\rho)$ is indeed the smallest ($\leq_1$) d-type of $\mathsf{c}$. So we can choose, for every PCFP$^{\mathcal{T}}$ constant $\mathsf{c}$, $\mathbf{L}^{\mathrm{d}}(\mathsf{c}) = \delta(\mathbf{T}(\mathsf{c}))$. Moreover since, for every PCFP$^{\mathcal{T}}$ constant $\mathsf{c}$,*

$$\mathbf{T}(\mathsf{c}) \in \{\iota_1, \iota_1 \to \iota_2, \iota_1 \times \iota_2 \to \iota_3 \mid \iota_1, \iota_2, \iota_3 \in \{\mathsf{nat}, \mathsf{bool}\}\},$$

*we have that, for every d-type $\phi$, $\mathbf{L}^{\mathrm{d}}(\mathsf{c}) \leq_1 \phi$ implies either $\phi \cong_1 \mathbf{L}^{\mathrm{d}}(\mathsf{c})$ or $\phi \in \mathbf{L}^{\mathrm{d}}_{\top}$.*

*2. It is worth mentioning that, in the rule ($\to$ E), the condition $\phi_1 \to \psi \leq \phi_2 \to \psi$ is used instead of $\phi_2 \leq \phi_1$. This is done to take into account the fact that: if $\phi_2 \to \psi$ is an $\top$-d-type then $\phi_1$ can be any d-type such that $\epsilon(\phi_1) = \epsilon(\phi_2)$.*

Note that, being $\vdash_1$ an inference system, the same terms can have different decorations. Let $\Lambda^{\vdash_1}_{\mathbf{T}}$ be the set of $\vdash_1$-decorated PCFP$^{\mathcal{T}}$ terms, i.e.,

$$\Lambda^{\vdash_1}_{\mathbf{T}} = \{M^\phi \mid \Sigma \vdash_1 M^\phi \text{ for some d-type } \phi \text{ and basis } \Sigma\}.$$

**Notation 7.22** *The notations $\epsilon(\cdot)$, $\delta(\cdot)$, and $\top(\cdot)$ are naturally extended to decorated terms. In particular $\epsilon(M^\phi)$ is simply the term obtained from $M^\phi$ by erasing all the d-type decorations, while, for each $a \in \{\delta, \top\}$ and $M \in \Lambda_{\mathbf{T}}$, $a(M)$ is the decorated term obtained from $M$ by decorating each subterm with the corresponding $a$-d-type.*

The proof of the following fact is immediate.

**Fact 7.23** *1. $\Sigma \vdash_1 M^\phi$ implies $\vdash_{\mathbf{T}} \epsilon(M^\phi) : \epsilon(\phi)$ and $\epsilon(\Sigma) \supseteq \mathrm{FV}(M)$.*

*2. $\vdash_{\mathbf{T}} M : \rho$ implies*

$$\text{(Var)}\ \frac{\phi_1 \leq \phi_2}{\Sigma, x : \phi_1 \vdash x^{\phi_2}} \qquad \text{(Con)}\ \frac{\mathbf{L}^{\mathrm{d}}(\mathsf{c}) \leq \phi}{\Sigma \vdash \mathsf{c}^{\phi}}$$

$$(\to \text{I})\ \frac{\Sigma, x : \phi \vdash M^{\psi}}{\Sigma \vdash (\lambda x^{\phi}.M^{\psi})^{\phi \to \psi}} \qquad (\to \text{E})\ \frac{\Sigma \vdash M^{\phi_1 \to \psi} \quad \Sigma \vdash N^{\phi_2} \quad \phi_1 \to \psi \leq \phi_2 \to \psi}{\Sigma \vdash (M^{\phi_1 \to \psi} N^{\phi_2})^{\psi}}$$

$$(\times \text{I})\ \frac{\Sigma \vdash {M_1}^{\phi_1} \quad \Sigma \vdash {M_2}^{\phi_2}}{\Sigma \vdash \langle {M_1}^{\phi_1}, {M_2}^{\phi_2} \rangle^{\phi_1 \times \phi_2}} \qquad (\times \text{E}_i)\ \frac{\Sigma \vdash M^{\phi_1 \times \phi_2}}{\Sigma \vdash (\mathsf{proj}_i M^{\phi_1 \times \phi_2})^{\phi_i}}\ i \in \{1, 2\}$$

$$\text{(Fix)}\ \frac{\Sigma, x : \phi \vdash M^{\phi}}{\Sigma \vdash (\mathsf{fix}\, x^{\phi}.M^{\phi})^{\phi}}$$

$$\text{(If)}\ \frac{\Sigma \vdash N^{a_0\,\mathsf{bool}} \quad \Sigma \vdash {M_1}^{\phi_1} \quad \Sigma \vdash {M_2}^{\phi_2} \quad \phi_1 \leq \phi \quad \phi_2 \leq \phi \quad a_0 = \top \text{ implies } \phi \in \mathbf{L}^{\mathrm{d}}_{\top}}{\Sigma \vdash (\mathsf{if}\, N^{a_0\,\mathsf{bool}}\, \mathsf{then}\, {M_1}^{\phi_1}\, \mathsf{else}\, {M_2}^{\phi_2})^{\phi}}$$

$$\text{(Case)}\ \frac{\Sigma \vdash N^{a_0\,\mathsf{nat}} \quad \Sigma \vdash M^{\phi_1} \quad \Sigma \vdash F^{a^{\mathsf{nat}} \to \phi_2} \quad \phi_1 \leq \phi \quad a^{\mathsf{nat}} \to \phi_2 \leq \delta^{\mathsf{nat}} \to \phi \quad a_0 = \top \text{ implies } \phi \in \mathbf{L}^{\mathrm{d}}_{\top}}{\Sigma \vdash \mathsf{case}(N^{a_0\,\mathsf{nat}}, M^{\phi_1}, F^{a^{\mathsf{nat}} \to \phi_2})^{\phi}}$$

$$\text{(It)}\ \frac{\Sigma \vdash N^{a_0\,\mathsf{nat}} \quad \Sigma \vdash M^{\phi_1} \quad \Sigma \vdash F^{\phi_2 \to \phi_3} \quad \phi_1 \leq \phi \quad \phi_2 \to \phi_3 \leq \phi \to \phi \quad a_0 = \top \text{ implies } \phi \in \mathbf{L}^{\mathrm{d}}_{\top}}{\Sigma \vdash \mathsf{it}(N^{a_0\,\mathsf{nat}}, M^{\phi_1}, F^{\phi_2 \to \phi_3})^{\phi}}$$

$$\text{(Rec)}\ \frac{\Sigma \vdash N^{a_0\,\mathsf{nat}} \quad \Sigma \vdash M^{\phi_1} \quad \Sigma \vdash F^{a^{\mathsf{nat}} \to \phi_2 \to \phi_3} \quad \phi_1 \leq \phi \quad a^{\mathsf{nat}} \to \phi_2 \to \phi_3 \leq \delta^{\mathsf{nat}} \to \phi \to \phi \quad a_0 = \top \text{ implies } \phi \in \mathbf{L}^{\mathrm{d}}_{\top}}{\Sigma \vdash \mathsf{rec}(N^{a_0\,\mathsf{nat}}, M^{\phi_1}, F^{a^{\mathsf{nat}} \to \phi_2 \to \phi_3})^{\phi}}$$

Figure 7.4: Rules for d-type assignment (system $\vdash_1$)

*(a)* $\delta(\mathrm{FV}(M)) \vdash_1 \delta(M)$, *and*

*(b) for all basis* $\Sigma$ *such that* $\Sigma = \{x^{\epsilon(\psi)} : \psi \mid x^{\epsilon(\psi)} \in \mathrm{FV}(M)$ *for some* $\psi \in \mathbf{L}^{\mathrm{d}}\}$, $\Sigma \vdash_1 \top(M)$. □

**Example 7.24** *Take the terms* $M$ *and* $N$ *of Examples III.1 and III.2 of page 140:*

- $M = (\lambda x^{\mathsf{nat}}.3)P$, *and*
- $N = (\lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, Q_2\rangle)P$, *where* $Q_1$ *does not contain any occurrence of* $x$.

*Assume for simplicity that* $P$, $Q_1$ *and* $Q_2$ *are closed. It it easy to check that the following* $\vdash_1$*-typings hold:*

- $\emptyset \vdash_1 ((\lambda x^{\top^{\mathsf{nat}}}.3^{\delta^{\mathsf{nat}}})^{\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}} P^{\top^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$, *and*
- $\emptyset \vdash_1 ((\lambda x^{\top^{\mathsf{nat}}}.(\mathsf{proj}_1\langle Q_1^{\delta^{\mathsf{nat}}}, Q_2^{\top^{\mathsf{nat}}}\rangle^{\delta^{\mathsf{nat}} \times \top^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}})^{\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}} P^{\top^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$.

To state the soundness of the d-type assignment system w.r.t. the semantics we introduce the following definition.

**Definition 7.25** *1. Two environments* $e_1$, $e_2$ *are* $\Sigma$*-related if and only if, for all* $x^{\psi} \in \Sigma$, $\langle e_1(x), e_2(x)\rangle \in [\![\psi]\!]$.

*2. Let* $\Sigma \vdash_1 M^{\phi}$ *and* $\Sigma \vdash_1 N^{\phi}$. *We write* $\epsilon(M^{\phi}) \sim_{\phi}^{\Sigma} \epsilon(N^{\phi})$ *to mean that for all* $e_1$, $e_2$, *if* $e_1$ *and* $e_2$ *are* $\Sigma$*-related, then* $\langle [\![\epsilon(M^{\phi})]\!]_{e_1}, [\![\epsilon(N^{\phi})]\!]_{e_2}\rangle \in [\![\phi]\!]$.

Now we can state the soundness of the system $\vdash_1$.

**Theorem 7.26 (Soundness of $\vdash_1$)** *Let* $\Sigma \vdash_1 M^{\phi}$. *Then* $\epsilon(M^{\phi}) \sim_{\phi}^{\Sigma} \epsilon(M^{\phi})$.

**Proof.** By induction on the derivation of $\Sigma \vdash_1 M^{\phi}$, using Theorem 7.10 (soundness of $\leq_1$) for the rules involving entailment. □

### 7.2.3 A dead-code elimination

In this section we introduce a dead-code elimination mapping $\mathbf{O}_1$ that, given an $\vdash_1$-decorated term $M^{\phi}$, defines an optimized version of it.

#### Dummy variables

For each type $\rho$, define $\mathsf{d}^{\rho}$, $\mathsf{d}_1^{\rho}$, $\mathsf{d}_2^{\rho}$, ..., be *dummy variables* of type $\rho$. Dummy variables are not present in the original programs: they are introduced by the dead-code elimination mapping $\mathbf{O}_1$ as placeholders for the dead-code removed. Therefore each occurrence of a dummy variable in a program is dead-code. So they can be handled in a special by the dead-code type assignment system $\vdash_1$. In particular, given a term $M$ obtained by a correct application of the mapping $\mathbf{O}_1$ (see Theorem 7.33), it is safe to allow that any d-type $\phi \in \mathbf{L}^{\mathrm{d}}(\rho)$ can be assigned to a dummy variable $\mathsf{d}^{\rho}$ occurring in $M$. This can be ensured by:

- forbidding dummy variables to occur in bases, and
- adding to the rules in Fig. 7.4 the following rule

$$(\mathsf{d})\ \frac{\vdash_{\mathbf{T}} \mathsf{d}^{\rho} : \rho \qquad \phi \in \mathbf{L}^{\mathrm{d}}(\rho)}{\vdash \mathsf{d}^{\phi}} \quad .$$

For every term $M$, define $\mathrm{FV}'(M)$ be the set of the free non dummy variables in $M$.

### The simplification mapping

Let $\Lambda^{\vdash_1}_{\mathbf{T}}$ be the set of all decorated terms which are defined according to Definition 7.20, i.e., $\Lambda^{\vdash_1}_{\mathbf{T}} = \{M^{\psi} \mid \Sigma \vdash_1 M : \psi \text{ for some d-type } \psi \text{ and basis } \Sigma\}$.

**Definition 7.27 (Simplification mapping $\mathbf{O}_1$)** *1. The function*

$$\mathbf{O}_1 : \Lambda^{\vdash_1}_{\mathbf{T}} \to \Lambda^{\vdash_1}_{\mathbf{T}}$$

*is defined by the clauses in Fig. 7.5.*

*2. If $\Sigma$ is a basis then*

$$\mathbf{O}_1(\Sigma) = \{x : \chi \mid x : \chi \in \Sigma \text{ and } \chi \notin \mathbf{L}^{\mathrm{d}}_{\top}\}.$$

We have immediately that if $\Sigma \vdash_1 M^{\phi}$ then $\mathbf{O}_1(\Sigma) \vdash_1 \mathbf{O}_1(M^{\phi})$ and $\mathbf{O}_1(\Sigma) \subseteq \Sigma$.

To prove the correctness of the simplification mapping $\mathbf{O}_1$ we identify a subset of $\vdash_1$-typings for which the $\sim^{\Sigma}_{\phi}$ relations implies the $\simeq^{\mathrm{gnd}}_{\mathrm{obs}}$ relation.

**Definition 7.28 (Faithful $\vdash_1$-typing)** *$\Sigma \vdash_1 M^{\phi}$ is a* faithful $\vdash_1$*-type assignment statement if $\phi \in \mathbf{L}^{\mathrm{d}}_{\delta}$, and for all $x : \psi \in \Sigma$, $\psi \in \mathbf{L}^{\mathrm{d}}_{\top} \cup \mathbf{L}^{\mathrm{d}}_{\delta}$.*

**Example 7.29** *Both the two $\vdash_1$-typings in Example 7.24 are faithful. By applying the simplification mapping $\mathbf{O}_1$ we get the following simplifications of the decorated version of $M$ and $N$:*

- $M'^{\delta} = ((\lambda x^{\top^{\mathsf{nat}}}.3^{\delta^{\mathsf{nat}}})^{\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}} \mathsf{d}^{\top^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$, *and*
- $N'^{\delta} = ((\lambda x^{\top^{\mathsf{nat}}}.(\mathsf{proj}_1\langle Q_1^{\delta^{\mathsf{nat}}}, \mathsf{d}_1^{\top^{\mathsf{nat}}}\rangle^{\delta^{\mathsf{nat}} \times \top^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}})^{\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}} \mathsf{d}_2^{\top^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$.

*Then, by erasing all the d-type decorations, we get:*

- $M' = (\lambda x^{\mathsf{nat}}.3)\mathsf{d}^{\mathsf{nat}}$, *and*
- $N' = (\lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, \mathsf{d}_1^{\mathsf{nat}}\rangle)\mathsf{d}_2^{\mathsf{nat}}$.

The correctness proof of the simplification mapping $\mathbf{O}_1$ rely on the following theorem.

$\mathbf{O}_1(M^{\psi}) = \mathsf{d}^{\psi}, \quad \text{if } \psi \in \mathbf{L}^2_{\top}$

otherwise:

$\mathbf{O}_1(\mathsf{c}^{\psi}) = \mathsf{c}^{\psi}$

$\mathbf{O}_1(x^{\psi}) = x^{\psi}$

$\mathbf{O}_1(\langle {M_1}^{\psi_1}, {M_2}^{\psi_2} \rangle^{\psi_1 \times \psi_2}) = \langle \mathbf{O}_1({M_1}^{\psi_1}), \mathbf{O}_1({M_2}^{\psi_2}) \rangle^{\psi_1 \times \psi_2}$

$\mathbf{O}_1((\mathsf{proj}_i M^{\psi_1 \times \psi_2})^{\psi_i}) = (\mathsf{proj}_i \mathbf{O}_1(M^{\psi_1 \times \psi_2}))^{\psi_i}, \quad \text{where } i \in \{1, 2\}$

$\mathbf{O}_1((M^{\psi_1 \to \psi_2} N^{\psi'_1})^{\psi}) = (\mathbf{O}_1(M^{\psi \to \psi_2}) \mathbf{O}_1(N^{\psi'_1}))^{\psi_2}$

$\mathbf{O}_1((\lambda x^{\psi_1}.M^{\psi_2})^{\psi_1 \to \psi_2}) = (\lambda x^{\psi_1}.\mathbf{O}_1(M^{\psi_2}))^{\psi_1 \to \psi_2}$

$\mathbf{O}_1((\mathsf{fix}\, x^{\phi}.M^{\phi})^{\phi}) = \mathsf{fix}\, x^{\phi}.\mathbf{O}_1(M^{\phi})^{\phi}$

$\mathbf{O}_1((\mathsf{if}\ N^{\delta^{\mathsf{bool}}}\ \mathsf{then}\ {M_1}^{\psi_1}\ \mathsf{else}\ {M_2}^{\psi_2})^{\psi}) = (\mathsf{if}\ \mathbf{O}_1(N^{\delta^{\mathsf{bool}}})\ \mathsf{then}\ \mathbf{O}_1({M_1}^{\psi_1})\ \mathsf{else}\ \mathbf{O}_1({M_2}^{\psi_2}))^{\psi}$

$\mathbf{O}_1(\mathsf{case}(N^{\delta^{\mathsf{nat}}}, M^{\psi_1}, {F^{a^{\mathsf{nat}} \to \psi_2}})^{\psi}) = \mathsf{case}(\mathbf{O}_1(N^{\delta^{\mathsf{nat}}}), \mathbf{O}_1(M^{\psi_1}), \mathbf{O}_1(F^{a^{\mathsf{nat}} \to \psi_2}))^{\psi}$

$\mathbf{O}_1(\mathsf{it}(N^{\delta^{\mathsf{nat}}}, M^{\phi_1}, F^{\phi_2 \to \phi_3})^{\phi}) = \mathsf{it}(\mathbf{O}_1(N^{\delta^{\mathsf{nat}}}), \mathbf{O}_1(M^{\phi_1}), \mathbf{O}_1(F^{\phi_2 \to \phi_3}))^{\phi}$

$\mathbf{O}_1(\mathsf{rec}(N^{\delta^{\mathsf{nat}}}, M^{\phi_1}, {F^{a^{\mathsf{nat}} \to \phi_2 \to \phi_3}})^{\phi}) = \mathsf{rec}(\mathbf{O}_1(N^{\delta^{\mathsf{nat}}}), \mathbf{O}_1(M^{\phi_1}), \mathbf{O}_1(F^{a^{\mathsf{nat}} \to \phi_2 \to \phi_3}))^{\phi}$

Figure 7.5: Mapping $\mathbf{O}_1$ on terms

**Theorem 7.30** *Let $\Sigma \vdash_1 M^\phi$ and $\Sigma \vdash_1 N^\phi$ be faithful $\vdash_1$-typings. Then $\epsilon(M^\phi) \sim_\phi^\Sigma \epsilon(N^\phi)$ implies $\epsilon(M^\phi) \simeq_{\text{obs}}^{\text{gnd}} \epsilon(N^\phi)$.*

**Proof.** Since $\phi \in \mathbf{L}_\delta^{\text{d}}$, from Theorem 2.8 and from Corollary 7.5, we have that: if, for all environments $e$, $\langle [\![M]\!]_e, [\![N]\!]_e \rangle \in [\![\phi]\!]$ then $\epsilon(M^\phi) \simeq_{\text{obs}}^{\text{gnd}} \epsilon(N^\phi)$. So we can prove the result by showing that whenever $\Sigma \vdash_1 M : \phi$ and $\Sigma \vdash_1 N : \phi$ are faithful, i.e., such that

- $\phi \in \mathbf{L}_\delta^{\text{d}}$, and
- for all $x : \psi \in \Sigma$, $\psi \in \mathbf{L}_\top^{\text{d}} \cup \mathbf{L}_\delta^{\text{d}}$,

$\epsilon(M^\phi) \sim_\phi^\Sigma \epsilon(N^\phi)$ implies that, for all environments $e$, $\langle [\![M]\!]_e, [\![N]\!]_e \rangle \in [\![\phi]\!]$.

Let $\Sigma'$ is the restriction of $\Sigma$ to the variables $\{x_1, \ldots, x_n\} = \text{FV}'(M) \cup \text{FV}'(N)$ (observe that $\Sigma$ contains all the free (non dummy) variables of $M$ and $N$). It is easy to see that, for any environment $e$, the fact that $\Sigma \vdash_1 M : \phi$ and $\Sigma \vdash_1 N : \phi$ are faithful, implies that $e$ is $\Sigma'$ related with $e$. Let now $\epsilon(M^\phi) \sim_\phi^\Sigma \epsilon(N^\phi)$. Then (by Definition 7.25.2) $\langle [\![\epsilon(M^\phi)]\!]_e, [\![\epsilon(N^\phi)]\!]_e \rangle \in [\![\phi]\!]$, and this proves the result. □

**Remark 7.31** *We should notice that Theorem 7.30 can be proved with an hypothesis weaker than faithfulness of $\Sigma \vdash_1 M^\phi$ and $\Sigma \vdash_1 N^\phi$. Indeed it is sufficient to require that $\phi \leq_1 \delta(\epsilon(\phi))$ and, for all $x : \psi \in \Sigma$, $\delta(\epsilon(\psi)) \leq_1 \psi$.*

The following result follows from the d-type semantics.

**Theorem 7.32** *If $\Sigma \vdash_1 M^\phi$ then $\epsilon(M^\phi) \sim_\phi^\Sigma \epsilon(\mathbf{O}_1(M^\phi))$.*

**Proof.** Just notice that $\sim_\phi^\Sigma$ is a congruence. □

This result is especially interesting when the typing of $M$ is faithful since, using the theorem above and Theorem 7.30, we can prove that that if $\Sigma \vdash_1 M^\phi$ is a faithful typing statement then $\epsilon(M^\phi)$ and $\epsilon(\mathbf{O}_1(M^\phi))$ are observationally equivalent.

**Theorem 7.33 (Correctness of the mapping $\mathbf{O}_1$)** *Let $\Sigma \vdash_1 M^\phi$ be a faithful $\vdash_1$-typing. Then $\epsilon(M^\phi) \simeq_{\text{obs}}^{\text{gnd}} \epsilon(\mathbf{O}_1(M^\phi))$.*

**Proof.** It easy to see that if $\Sigma \vdash_1 M^\phi$ is a faithful $\vdash_1$-typing then also $\Sigma \vdash_1 \mathbf{O}_1(M^\phi)$ is a faithful $\vdash_1$-typing. So the result follows immediately from Theorems 7.32 and 7.30. □

**Example 7.34** *Let $\vdash_{\mathbf{T}} M : \mathsf{nat}$ where $\text{FV}(M) = \{f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}}, u^{\mathsf{nat}}, v^{\mathsf{nat}}\}$ and*

$$M = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.(fg + gx) + (\lambda y^{\mathsf{nat}}.1)P)(\lambda z^{\mathsf{nat}}.3)Q,$$

*with $P$, $Q$ terms of type $\mathsf{nat}$ such that $\text{FV}(P) = \{u^{\mathsf{nat}}\}$ and $\text{FV}(Q) = \{v^{\mathsf{nat}}\}$.*

*It is easy to check that* $\Sigma' \vdash_1 M'^{\delta^{\mathsf{nat}}}$ *is a faithful* $\vdash_1$*-typing, where (writing, for short,* $\delta$ *and* $\top$ *instead of* $\delta^{\mathsf{nat}}$ *and* $\top^{\mathsf{nat}}$*):* $\Sigma' = \{f^{\,(\delta\to\delta)\to\delta}, u^{\,\top}, v^{\,\top}\}$ *and* $M'^{\delta} =$

$$\begin{array}{lll}(((\lambda g^{\top\to\delta}. & (\lambda x^{\top}. & \\ & (((f^{(\top\to\delta)\to\delta} g^{\top\to\delta})^{\delta} +^{\delta\times\delta\to\delta} (g^{\top\to\delta} x^{\top})^{\delta})^{\delta} & \\ & \quad +^{\delta\times\delta\to\delta}((\lambda y^{\top}.1^{\delta})^{\top\to\delta} P^{\top})^{\delta})^{\delta})^{\top\to\delta})^{(\top\to\delta)\to\top\to\delta} & \\ & (\lambda z^{\top}.3^{\delta})^{\top\to\delta})^{\top\to\delta} Q^{\top})^{\delta} & .\end{array}$$

*Applying the* $\mathbf{O}_1$ *simplification mapping we get* $\mathbf{O}_1(\Sigma') \vdash_1 \mathbf{O}_1(M'^{\delta})$*, where* $\mathbf{O}_1(\Sigma') = \{f^{\,(\delta\to\delta)\to\delta}\}$ *and* $\mathbf{O}_1(M'^{\delta}) =$

$$\begin{array}{lll}(((\lambda g^{\top\to\delta}. & (\lambda x^{\top}. & \\ & (((f^{(\top\to\delta)\to\delta} g^{\top\to\delta})^{\delta} +^{\delta\times\delta\to\delta} (g^{\top\to\delta} \mathsf{d}_1^{\top})^{\delta})^{\delta} & \\ & \quad +^{\delta\times\delta\to\delta}((\lambda y^{\top}.1^{\delta})^{\top\to\delta} \mathsf{d}_2^{\top})^{\delta})^{\delta})^{\top\to\delta})^{(\top\to\delta)\to\top\to\delta} & \\ & (\lambda z^{\top}.3^{\delta})^{\top\to\delta})^{\top\to\delta} \mathsf{d}_3^{\top})^{\delta}. & \end{array}$$

*Let* $N_1$ *be the term obtained from* $\mathbf{O}_1(M'^{\delta})$ *by erasing the d-type decorations, i.e.,* $N_1 = \epsilon(\mathbf{O}_1(M'^{\delta}))$*. We have* $\vdash_{\mathbf{T}} N_1 : \mathsf{nat}$*,* $\mathrm{FV}(N_1) = \{f^{\,(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}}, \mathsf{d}_1^{\,\mathsf{nat}}, \mathsf{d}_2^{\,\mathsf{nat}}, \mathsf{d}_3^{\,\mathsf{nat}}\}$ *and*

$$N_1 = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.fg + g\mathsf{d}_1^{\mathsf{nat}} + (\lambda y^{\mathsf{nat}}.1)\mathsf{d}_2^{\mathsf{nat}})(\lambda z^{\mathsf{nat}}.3)\mathsf{d}_3^{\mathsf{nat}} \quad .$$

## 7.3 Detecting and removing dead-code II

In Section 7.3.1 we introduce a weaker entailment relation between d-types, $\leq_2$, which induces a weaker dead-code type assignment, $\vdash_2$ (in Section 7.3.2). As we will see in Section 7.3.3, the system $\vdash_2$ enjoins a very effective simplification mapping, which transforms not only the code of a program but also its type.

### 7.3.1 A weaker entailment relation for dead-code types

**Definition 7.35 (Entailment relation $\leq_2$)** *Let* $\phi, \psi \in \mathbf{L}^{\mathrm{d}}$*. We write* $\phi \leq_2 \psi$ *to mean that* $\phi \leq \psi$ *is derivable by using the rules* (Ref) *and* $(\top)$ *in Fig. 7.1. By* $\cong_2$ *we denote the equivalence relation induced by* $\leq_2$*.*

It is immediate to see that $\leq_2$ is reflexive and transitive. Moreover, as for $\leq_1$ we have that, for all $\phi, \psi \in \mathbf{L}^{\mathrm{d}}$:

- $\phi \leq_2 \psi$ implies $\epsilon(\phi) = \epsilon(\psi)$, and
- if $\phi, \psi \in \mathbf{L}^{\mathrm{d}}_{\top}$ and $\epsilon(\phi) = \epsilon(\psi)$, then $\phi \cong_2 \psi$.

Since the entailment rules for $\leq_2$ are a subset of the ones for $\leq_1$, we have that, for all $\phi, \psi \in \mathbf{L}^{\mathrm{d}}$, $\phi \leq_2 \psi$ imply $\phi \leq_1 \psi$. Note that the reverse implication does not hold. For instance $\top^{\iota} \to \delta^{\iota} \leq_1 \delta^{\iota} \to \delta^{\iota}$ and $\delta^{\iota} \times \delta^{\iota} \leq_1 \top^{\iota} \times \delta^{\iota}$, however $\top^{\iota} \to \delta^{\iota} \not\leq_2 \delta^{\iota} \to \delta^{\iota}$ and

$\delta^\iota \times \delta^\iota \not\leq_2 \top^\iota \times \delta^\iota$. For every type $\rho$, the $\top$-d-types in $\mathbf{L}^{\mathrm{d}}_{\top}(\rho)$ are the maximal elements of $\mathbf{L}^{\mathrm{d}}(\rho)$ w.r.t. the relation $\leq_2$, and any two different d-types $\phi, \psi \notin \mathbf{L}^{\mathrm{d}}_{\top}$ are incomparable. For instance $(\top^{\mathsf{nat}} \to \top^{\mathsf{nat}}) \to \delta^{\mathsf{nat}}$ and $(\delta^{\mathsf{nat}} \to \top^{\mathsf{nat}}) \to \delta^{\mathsf{nat}}$, are equivalent w.r.t. $\cong_1$ but incomparable w.r.t. $\leq_2$.

The $\leq_2$ relation between d-types is sound w.r.t. the p.e.r. semantics of Definition 7.2.

**Theorem 7.36 (Soundness of $\leq_2$)** *$\phi \leq_2 \psi$ implies $[\![\phi]\!] \subseteq [\![\psi]\!]$.*

**Proof.** By Theorem 7.10, since $\phi \leq_2 \psi$ implies $\phi \leq_1 \psi$. □

Obviously $\leq_2$ is not complete. For instance $[\![\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}]\!] \subseteq [\![\delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}]\!]$, but $\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}} \not\leq_2 \delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$.

### 7.3.2 A weaker assigment system for detecting dead-code

**Definition 7.37 (D-type assignment system $\vdash_2$)** *We write $\Sigma \vdash_2 M^\phi$ to mean that $\Sigma \vdash M^\phi$ can be derived by the rules in Fig. 7.4 where $\leq$ denotes the $\leq_2$ entailment relation.*

It is immediate to prove the analogous of Fact 7.23:

**Fact 7.38** *1. $\Sigma \vdash_2 M^\phi$ implies $\vdash_{\mathbf{T}} \epsilon(M^\phi) : \epsilon(\phi)$ and $\epsilon(\Sigma) \supseteq \mathrm{FV}(M)$.*

*2. $\vdash_{\mathbf{T}} M : \rho$ implies*

*(a) $\delta(\mathrm{FV}(M)) \vdash_2 \delta(M)$, and*

*(b) for all basis $\Sigma$ such that $\Sigma = \{x^{\epsilon(\psi)} : \psi \mid x^{\epsilon(\psi)} \in \mathrm{FV}(M)$ for some $\psi \in \mathbf{L}^{\mathrm{d}}\}$, $\Sigma \vdash_2 \top(M)$.* □

**Example 7.39** *Take the terms $M$ and $N$ of Examples III.1 and III.2 of page 140:*

- *$M = (\lambda x^{\mathsf{nat}}.3)P$, and*
- *$N = (\lambda x^{\mathsf{nat}}.\mathsf{proj}_1\langle Q_1, Q_2\rangle)P$, where $Q_1$ does not contain any occurrence of $x$.*

*Assume for simplicity that $P$, $Q_1$ and $Q_2$ are closed. It it easy to check that the following $\vdash_1$-typings hold:*

- $\emptyset \vdash_2 ((\lambda x^{\top^{\mathsf{nat}}}.3^{\delta^{\mathsf{nat}}})^{\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}} P^{\top^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$, *and*
- $\emptyset \vdash_2 ((\lambda x^{\top^{\mathsf{nat}}}.(\mathsf{proj}_1\langle Q_1^{\delta^{\mathsf{nat}}}, Q_2^{\top^{\mathsf{nat}}}\rangle^{\delta^{\mathsf{nat}} \times \top^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}})^{\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}} P^{\top^{\mathsf{nat}}})^{\delta^{\mathsf{nat}}}$.

We state now the soundness of the system $\vdash_2$.

**Theorem 7.40 (Soundness of $\vdash_2$)** *Let $\Sigma \vdash_2 M^\phi$. Then $\epsilon(M^\phi) \sim_\phi^\Sigma \epsilon(M^\phi)$.*

**Proof.** By Theorem 7.26, since $\Sigma \vdash_2 M^\phi$ implies $\Sigma \vdash_1 M^\phi$. □

$$
\begin{array}{lcl}
\mathbf{O}_2(\delta^\iota) & = & \delta^\iota \\
\mathbf{O}_2(\phi \to \psi) & = & \begin{cases} \mathbf{O}_2(\phi) \to \mathbf{O}_2(\psi) & \text{if } \phi \notin \mathbf{L}^{\mathrm{d}}_\top \\ \mathbf{O}_2(\psi) & \text{if } \phi \in \mathbf{L}^{\mathrm{d}}_\top \end{cases} \\
\mathbf{O}_2(\phi \times \psi) & = & \begin{cases} \mathbf{O}_2(\phi) \times \mathbf{O}_2(\psi) & \text{if } \phi, \psi \notin \mathbf{L}^{\mathrm{d}}_\top \\ \mathbf{O}_2(\phi) & \text{if } \psi \in \mathbf{L}^{\mathrm{d}}_\top \\ \mathbf{O}_2(\psi) & \text{if } \phi \in \mathbf{L}^{\mathrm{d}}_\top \end{cases}
\end{array}
$$

Figure 7.6: Mapping $\mathbf{O}_2$ on types

### 7.3.3 A clever dead-code elimination

In this section, following [Berardi, 1993b], we introduce a simplification mapping $\mathbf{O}_2$ that, given a $\vdash_2$-decorated term $M^\phi$, defines an optimized version of it. The simplification performed by $\mathbf{O}_2$ can be much stronger than those performed by $\mathbf{O}_1$. In certain cases $\mathbf{O}_2$ can also replace program constructors transforming, for instance, a recursor $\mathsf{rec}(N^{\delta^{\mathsf{nat}}}, M^{\phi_1}, F^{\top^{\mathsf{nat}} \to \phi_2 \to \phi_3})^\psi$ in an iterator $\mathsf{it}(N^{\delta^{\mathsf{nat}}}, M^{\phi_1}, G^{\phi_2 \to \phi_3})^\psi$.

Since $\mathbf{O}_2$ introduces a deep change in the structure of the term we need to define it also on types and not simply on terms.

#### The mapping on types

**Definition 7.41 (Simplification mapping $\mathbf{O}_2$ on types)** *The function* $\mathbf{O}_2 : (\mathbf{L}^{\mathrm{d}} - \mathbf{L}^{\mathrm{d}}_\top) \to (\mathbf{L}^{\mathrm{d}} - \mathbf{L}^{\mathrm{d}}_\top)$ *is defined by the clauses in Fig. 7.6.*

Note that, in general, $\epsilon(\mathbf{O}_2(\phi)) \neq \epsilon(\phi)$. There is anyway a (PCFP-definable) isomorphism between the interpretation $\mathbf{I}(\epsilon(\phi))$ of $\epsilon(\phi)$ for any d-type $\phi \notin \mathbf{L}^{\mathrm{d}}_\top$ and $\mathbf{I}(\epsilon(\mathbf{O}_2(\phi)))$ which is defined, for each d-type $\phi \notin \mathbf{L}^{\mathrm{d}}_\top$, by a pair of PCFP terms. The basic idea is to map a constant function of d-type $\top^{\iota_1} \to \delta^{\iota_2}$ in a term of d-type $\delta^{\iota_2}$ (and vice-versa). This idea is lifted to higher types in a standard way.

**Definition 7.42** *For all* $\chi \in \mathbf{L}^{\mathrm{d}} - \mathbf{L}^{\mathrm{d}}_\top$, *the closed terms* $\mathbf{I}_{[\chi]}$ *and* $\mathbf{J}_{[\chi]}$ *are defined by induction on* $\chi$ *according to the clauses in Fig. 7.7.*

**Fact 7.43** *For all* $\chi \in \mathbf{L}^{\mathrm{d}} - \mathbf{L}^{\mathrm{d}}_\top$, $\emptyset \vdash_{\mathbf{T}} \mathbf{I}_{[\chi]} : \epsilon(\chi) \to \epsilon(\mathbf{O}_2(\chi))$ *and* $\emptyset \vdash_{\mathbf{T}} \mathbf{J}_{[\chi]} : \epsilon(\mathbf{O}_2(\chi)) \to \epsilon(\chi)$. □

The following technical lemmas can be proved by a standard induction on types.

**Lemma 7.44** *For all* $\chi \in \mathbf{L}^{\mathrm{d}} - \mathbf{L}^{\mathrm{d}}_\top$,

1. $[\lambda\, x^{\epsilon(\mathbf{O}_2(\chi))}.\mathbf{I}_{[\chi]}(\mathbf{J}_{[\chi]}x)] = [\lambda\, x^{\epsilon(\mathbf{O}_2(\chi))}.x]$, *and*

$$\mathbf{I}_{[\delta^\iota]} = \lambda\, x^\iota.x$$

$$\mathbf{J}_{[\delta^\iota]} = \lambda\, x^\iota.x$$

$$\mathbf{I}_{[\phi\to\psi]} = \begin{cases} \lambda f^{\epsilon(\phi\to\psi)}.\lambda z^{\epsilon(\mathbf{O}_2(\phi))}.\mathbf{I}_{[\psi]}(f(\mathbf{J}_{[\phi]}z)) & \text{if } \phi \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ \lambda f^{\epsilon(\phi\to\psi)}.\mathbf{I}_{[\psi]}(f\,\mathsf{d}^{\epsilon(\phi)}) & \text{if } \phi \in \mathbf{L}^{\mathrm{d}}_{\top} \end{cases}$$

$$\mathbf{J}_{[\phi\to\psi]} = \begin{cases} \lambda f^{\epsilon(\mathbf{O}_2(\phi)\to\mathbf{O}_2(\psi))}.\lambda z^{\epsilon(\phi)}.\mathbf{J}_{[\psi]}(f(\mathbf{I}_{[\phi]}z)) & \text{if } \phi \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ \lambda y^{\epsilon(\mathbf{O}_2(\psi))}.\lambda z^{\epsilon(\phi)}.\mathbf{J}_{[\psi]}y & \text{if } \phi \in \mathbf{L}^{\mathrm{d}}_{\top} \end{cases}$$

$$\mathbf{I}_{[\phi_1\times\phi_2]} = \begin{cases} \lambda z^{\epsilon(\phi_1\times\phi_2)}.\langle \mathbf{I}_{[\phi_1]}(\mathsf{proj}_1 z), \mathbf{I}_{[\phi_2]}(\mathsf{proj}_2 z)\rangle & \text{if } \phi_1, \phi_2 \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ \lambda z^{\epsilon(\phi_1\times\phi_2)}.\mathbf{I}_{[\phi_1]}(\mathsf{proj}_1 z) & \text{if } \phi_2 \in \mathbf{L}^{\mathrm{d}}_{\top} \\ \lambda z^{\epsilon(\phi_1\times\phi_2)}.\mathbf{I}_{[\phi_2]}(\mathsf{proj}_2 z) & \text{if } \phi_1 \in \mathbf{L}^{\mathrm{d}}_{\top} \end{cases}$$

$$\mathbf{J}_{[\phi_1\times\phi_2]} = \begin{cases} \lambda z^{\epsilon(\mathbf{O}_2(\phi_1)\times\mathbf{O}_2(\phi_2))}.\langle \mathbf{J}_{[\phi_1]}(\mathsf{proj}_1 z), \mathbf{J}_{[\phi_2]}(\mathsf{proj}_2 z)\rangle & \text{if } \phi_1, \phi_2 \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ \lambda z^{\epsilon(\mathbf{O}_2(\phi_1))}.\langle \mathbf{J}_{[\phi_1]}z, \mathsf{d}^{\epsilon(\phi_2)}\rangle & \text{if } \phi_2 \in \mathbf{L}^{\mathrm{d}}_{\top} \\ \lambda z^{\epsilon(\mathbf{O}_2(\phi_2))}.\langle \mathsf{d}^{\epsilon(\phi_1)}, \mathbf{J}_{[\phi_2]}z\rangle & \text{if } \phi_1 \in \mathbf{L}^{\mathrm{d}}_{\top} \end{cases}$$

Figure 7.7: Mappings $\mathbf{I}_{[\![]\!]}$ and $\mathbf{J}_{[\![]\!]}$

*2.* $[\lambda\, x^{\epsilon(\chi)}.\mathbf{J}_{[\chi]}(\mathbf{I}_{[\chi]}x)] = [\lambda\, x^{\epsilon(\chi)}.x]$. □

It is easy to show that, for all $\phi, \psi \in \mathbf{L}^{\mathrm{d}} - \mathbf{L}^{\mathrm{d}}_{\top}$, $\mathbf{I}_{[\phi\to\psi]}$ and $\mathbf{J}_{[\phi\to\psi]}$ commute with application.

**Lemma 7.45** *Let $\phi, \psi \notin \mathbf{L}^{\mathrm{d}}_{\top}$, and $M, N$ be closed terms such that $M \sim_{\mathbf{O}_2(\phi\to\psi)} M$ and $N \sim_{\mathbf{O}_2(\phi)} N$. Then*

$$[\mathbf{J}_{[\psi]}(MN)] = [(\mathbf{J}_{[\phi\to\psi]}(M))(\mathbf{J}_{[\phi]}N)]$$

□

### The mapping on terms

We now define the strong simplification mapping $\mathbf{O}_2$ on terms. Note that the mapping $\mathbf{O}_2$ can modify the type of the term and of its subterms.

Let $\Lambda^{\vdash_2}_{\mathbf{T}}$ be the set of all decorated terms which are defined according to Definition 7.37, i.e.,

$$\Lambda^{\vdash_2}_{\mathbf{T}} = \{M^{\phi} \mid \Sigma \vdash_2 M^{\phi} \text{ for some d-type } \phi \text{ and basis } \Sigma\}.$$

In the following definition we assume that all free and bound variables of a term $M$ have different names, and that there is a mapping $\mathbf{o}$ which maps variables of type $\epsilon(\phi)$ in variables of type $\epsilon(\mathbf{O}_2(\phi))$. We assume that the domain and the range of $\mathbf{o}$ are disjoint.

**Definition 7.46 (Simplification mapping $\mathbf{O}_2$ on terms)** *1. The function*

$$\mathbf{O}_2 : \{M^{\phi} \mid M^{\phi} \in \Lambda^{\vdash_2}_{\mathbf{T}} \text{ and } \phi \notin \mathbf{L}^{\mathrm{d}}_{\top}\} \to \{M^{\phi} \mid M^{\phi} \in \Lambda^{\vdash_2}_{\mathbf{T}} \text{ and } \phi \notin \mathbf{L}^{\mathrm{d}}_{\top}\}$$

*is defined by the clauses in Fig. 7.8.*

*2. If $\Sigma$ is a basis then*

$$\mathbf{O}_2(\Sigma) = \{\mathbf{o}(x) : \mathbf{O}_2(\phi) \mid x : \phi \in \Sigma \text{ and } \phi \notin \mathbf{L}^{\mathrm{d}}_{\top}\}.$$

We have immediately that, if $\Sigma \vdash_2 M^{\phi}$ then $\mathbf{O}_2(\Sigma) \vdash_2 \mathbf{O}_2(M^{\phi})$.

Faithful $\vdash_2$-typings, assuring a correct use of the simplification mapping introduced in Section 7.3.3, are defined in the same way of faithful $\vdash_1$-typings (see Definition 7.28).

**Definition 7.47 (Faithful $\vdash_2$-typing)** $\Sigma \vdash_2 M^{\phi}$ *is a* faithful $\vdash_2$*-type assignment statement if $\phi \in \mathbf{L}^{\mathrm{d}}_{\delta}$, and for all $x : \psi \in \Sigma$, $\psi \in \mathbf{L}^{\mathrm{d}}_{\top} \cup \mathbf{L}^{\mathrm{d}}_{\delta}$.*

**Example 7.48** *Both the two $\vdash_2$-typings in Example 7.39 are faithful. By applying the simplification mapping $\mathbf{O}_2$ we get the following simplifications of the decorated version of $M$ and $N$:*

- $M'^{\delta} = 3^{\delta^{\mathsf{nat}}}$, *and*
- $N'^{\delta} = Q_1^{\delta^{\mathsf{nat}}}$.

$$\mathbf{O}_2(\mathsf{c}^{\phi}) = \mathsf{c}^{\phi}$$

$$\mathbf{O}_2(x^{\phi}) = \mathbf{o}(x)^{\mathbf{O}_2(\phi)}$$

$$\mathbf{O}_2(\langle {M_1}^{\phi_1}, {M_2}^{\phi_2}\rangle^{\phi_1\times\phi_2}) = \begin{cases} \langle \mathbf{O}_2({M_1}^{\phi_1}), \mathbf{O}_2({M_2}^{\phi_2})\rangle^{\mathbf{O}_2(\phi_1\times\phi_2)} & \text{if } \phi_1,\phi_2 \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ \mathbf{O}_2({M_1}^{\phi_1}) & \text{if } \phi_2 \in \mathbf{L}^{\mathrm{d}}_{\top} \\ \mathbf{O}_2({M_2}^{\phi_2}) & \text{if } \phi_1 \in \mathbf{L}^{\mathrm{d}}_{\top} \end{cases}$$

$$\mathbf{O}_2((\mathsf{proj}_i M^{\phi_1\times\phi_2})^{\phi_i}) = \begin{cases} (\mathsf{proj}_i \mathbf{O}_2(M^{\phi_1\times\phi_2}))^{\mathbf{O}_2(\phi_i)} & \text{if } \phi_1,\phi_2 \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ \mathbf{O}_2(M^{\phi_1\times\phi_2}) & \text{if } \phi_{(3-i)} \in \mathbf{L}^{\mathrm{d}}_{\top} \end{cases}$$
where $i \in \{1,2\}$

$$\mathbf{O}_2((M^{\phi_1\to\psi} N^{\phi_2})^{\psi}) = \begin{cases} (\mathbf{O}_2(M^{\phi_1\to\psi})\mathbf{O}_2(N^{\phi_2}))^{\mathbf{O}_2(\psi)} & \text{if } \phi_1 \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ \mathbf{O}_2(M^{\phi_1\to\psi}) & \text{if } \phi_1 \in \mathbf{L}^{\mathrm{d}}_{\top} \end{cases}$$

$$\mathbf{O}_2((\lambda x^{\phi}.M^{\psi})^{\phi\to\psi}) = \begin{cases} (\lambda \mathbf{o}(x)^{\mathbf{O}_2(\phi)}.\mathbf{O}_2(M^{\psi}))^{\mathbf{O}_2(\phi\to\psi)} & \text{if } \phi \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ \mathbf{O}_2(M^{\psi}) & \text{if } \phi \in \mathbf{L}^{\mathrm{d}}_{\top} \end{cases}$$

$$\mathbf{O}_2((\mathsf{fix}\, x^{\phi}.M^{\phi})^{\phi}) = (\mathsf{fix}\, \mathbf{O}_2(x^{\phi}).\mathbf{O}_2(M^{\phi}))^{\mathbf{O}_2(\phi)}$$

$$\mathbf{O}_2((\mathsf{if}\ N^{\delta^{\mathsf{bool}}}\ \mathsf{then}\ {M_1}^{\phi_1}\ \mathsf{else}\ {M_2}^{\phi_2})^{\psi}) = (\mathsf{if}\ \mathbf{O}_2(N^{\delta^{\mathsf{bool}}})\ \mathsf{then}\ \mathbf{O}_2({M_1}^{\phi_1})\ \mathsf{else}\ \mathbf{O}_2({M_2}^{\phi_2}))^{\mathbf{O}_2(\psi)}$$

$$\mathbf{O}_2(\mathsf{case}(N^{\delta^{\mathsf{nat}}}, M^{\phi_1}, F^{a^{\mathsf{nat}}\to\phi_2})^{\psi}) = \begin{cases} \mathsf{case}(\mathbf{O}_2(N^{\delta^{\mathsf{nat}}}), \mathbf{O}_2(M^{\phi_1}), \mathbf{O}_2(F^{a^{\mathsf{nat}}\to\phi_2}))^{\mathbf{O}_2(\psi)} & \text{if } a \neq \top \\ (\mathsf{if}\ (\mathbf{O}_2(N^{\delta^{\mathsf{nat}}}) =^{\delta^{\mathsf{nat}}\times\delta^{\mathsf{nat}}\to\delta^{\mathsf{nat}}} 0^{\delta^{\mathsf{nat}}})^{\delta^{\mathsf{bool}}}\ \mathsf{then}\ \mathbf{O}_2(M^{\phi_1})\ \mathsf{else}\ \mathbf{O}_2(F^{a^{\mathsf{nat}}\to\phi_2}))^{\mathbf{O}_2(\psi)} & \text{if } a = \top \end{cases}$$

$$\mathbf{O}_2(\mathsf{it}(N^{\delta^{\mathsf{nat}}}, M^{\phi_1}, F^{\phi_2\to\phi_3})^{\psi}) = \begin{cases} \mathsf{it}(\mathbf{O}_2(N^{\delta^{\mathsf{nat}}}), \mathbf{O}_2(M^{\phi_1}), \mathbf{O}_2(F^{\phi_2\to\phi_3}))^{\mathbf{O}_2(\psi)} & \text{if } \phi_2 \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ (\mathsf{if}\ (\mathbf{O}_2(N^{\delta^{\mathsf{nat}}}) =^{\delta^{\mathsf{nat}}\times\delta^{\mathsf{nat}}\to\delta^{\mathsf{nat}}} 0^{\delta^{\mathsf{nat}}})^{\delta^{\mathsf{bool}}}\ \mathsf{then}\ \mathbf{O}_2(M^{\phi_1})\ \mathsf{else}\ \mathbf{O}_2(F^{\phi_2\to\phi_3}))^{\mathbf{O}_2(\psi)} & \text{if } \phi_2 \in \mathbf{L}^{\mathrm{d}}_{\top} \end{cases}$$

$$\mathbf{O}_2(\mathsf{rec}(N^{\delta^{\mathsf{nat}}}, M^{\phi_1}, F^{a^{\mathsf{nat}}\to\phi_2\to\phi_3})^{\psi}) = \begin{cases} \mathsf{rec}(\mathbf{O}_2(N^{\delta^{\mathsf{nat}}}), \mathbf{O}_2(M^{\phi_1}), \mathbf{O}_2(F^{a^{\mathsf{nat}}\to\phi_2\to\phi_3}))^{\mathbf{O}_2(\psi)} & \text{if } a \neq \top \text{ and } \phi_2 \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ \mathsf{it}(\mathbf{O}_2(N^{\delta^{\mathsf{nat}}}), \mathbf{O}_2(M^{\phi_1}), \mathbf{O}_2(F^{a^{\mathsf{nat}}\to\phi_2\to\phi_3}))^{\mathbf{O}_2(\psi)} & \text{if } a = \top \text{ and } \phi_2 \notin \mathbf{L}^{\mathrm{d}}_{\top} \\ \mathsf{case}(\mathbf{O}_2(N^{\delta^{\mathsf{nat}}}), \mathbf{O}_2(M^{\phi_1}), \mathbf{O}_2(F^{a^{\mathsf{nat}}\to\phi_2\to\phi_3}))^{\mathbf{O}_2(\psi)} & \text{if } a \neq \top \text{ and } \phi_2 \in \mathbf{L}^{\mathrm{d}}_{\top} \\ (\mathsf{if}\ (\mathbf{O}_2(N^{\delta^{\mathsf{nat}}}) =^{\delta^{\mathsf{nat}}\times\delta^{\mathsf{nat}}\to\delta^{\mathsf{nat}}} 0^{\delta^{\mathsf{nat}}})^{\delta^{\mathsf{bool}}}\ \mathsf{then}\ \mathbf{O}_2(M^{\phi_1})\ \mathsf{else}\ \mathbf{O}_2(F^{a^{\mathsf{nat}}\to\phi_2\to\phi_3}))^{\mathbf{O}_2(\psi)} & \\ \qquad \text{if } a = \top \text{ and } \phi_2 \in \mathbf{L}^{\mathrm{d}}_{\top} & \end{cases}$$

Figure 7.8: Mapping $\mathbf{O}_2$ on terms

*Then, by erasing all the d-type decorations, we get:*

- $M' = 3$, *and*
- $N' = Q_1$.

Note that, if $\Sigma \vdash_2 M^\phi$ is a faithful $\vdash_2$-typing, then $\epsilon(\mathbf{O}_2(\phi)) = \epsilon(\phi)$ and $\mathbf{O}_2(\Sigma) \subseteq \Sigma$. Moreover the analogous of Theorem 7.30 holds.

**Theorem 7.49** *Let $\Sigma \vdash_2 M^\phi$ and $\Sigma \vdash_2 N^\phi$ be faithful $\vdash_2$-typings. Then $\epsilon(M^\phi) \sim_\phi^\Sigma \epsilon(M^\phi)$ implies $\epsilon(M^\phi) \simeq_{\mathrm{obs}}^{\mathrm{gnd}} \epsilon(N^\phi)$.*

**Proof.** A faithful $\vdash_2$-typing is also a faithful $\vdash_1$-typing. So the result follows immediately from Theorem 7.30. □

The correctness of the map $\mathbf{O}_2$ is rely on the following lemma.

**Lemma 7.50** *Let $\Sigma \vdash_2 M^\phi$ (where $\phi \notin \mathbf{L}^{\mathrm{d}}_\top$) and $e_1$, $e_2$ be two environments such that, for all $x : \psi \in \Sigma$, $\psi \notin \mathbf{L}^{\mathrm{d}}_\top$ implies $e_1(x) \sim_\psi [\mathbf{J}_{[\psi]}P]$, where $[P] = e_2(\mathbf{o}(x))$. Then $[\![M]\!]_{e_1} \sim_\phi [\![\mathbf{J}_{[\phi]}\epsilon(\mathbf{O}_2(M^\phi))]\!]_{e_2}$.*

**Proof.** By induction on the derivation of $\Sigma \vdash_2 M^\phi$. □

**Theorem 7.51** *Let $\Sigma \vdash_2 M^\phi$ be a faithful $\vdash_2$-typing. Then $\epsilon(M^\phi) \sim_\phi^\Sigma \epsilon(\mathbf{O}_2(M^\phi))$.*

**Proof.** Immediate by Lemma 7.50. □

As a consequence of this result we have that if $\Sigma \vdash_2 M^\phi$ is a faithful $\vdash_2$-type assignment then $\epsilon(M^\phi)$ and $\epsilon(\mathbf{O}_2(M^\phi))$ are observationally equivalent.

**Theorem 7.52 (Correctness of the mapping $\mathbf{O}_2$)** *Let $\Sigma \vdash_2 M^\phi$ be a faithful $\vdash_2$-typing. Then $\epsilon(M^\phi) \simeq_{\mathrm{obs}}^{\mathrm{gnd}} \epsilon(\mathbf{O}_2(M^\phi))$.*

**Proof.** It easy to see that if $\Sigma \vdash_2 M^\phi$ is a faithful $\vdash_2$-typing then also $\Sigma \vdash_2 \mathbf{O}_2(M^\phi)$ is a faithful $\vdash_2$-typing. So the result follows immediately from Theorems 7.51 and 7.49. □

**Example 7.53** *Let*

$$M = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.fg + gx + (\lambda y^{\mathsf{nat}}.1)P)(\lambda z^{\mathsf{nat}}.3)Q$$

*be the term of Example 7.34. It is easy to check that $\Sigma'' \vdash_2 M''^{\delta^{\mathsf{nat}}}$ is a faithful $\vdash_2$-typing, where (writing, for short, $\delta$ and $\top$ instead of $\delta^{\mathsf{nat}}$ and $\top^{\mathsf{nat}}$): $\Sigma'' = \{f : (\delta \to \delta) \to \delta, u : \top, v : \delta\}$ and $M'^\delta =$*

$$\begin{array}{ll}(((\lambda g^{\delta\to\delta}. & (\lambda x^\delta.\\ & (((f^{(\delta\to\delta)\to\delta} g^{\delta\to\delta})^\delta +^{\delta\times\delta\to\delta} (g^{\delta\to\delta} x^\delta)^\delta)^\delta\\ & \qquad +^{\delta\times\delta\to\delta}((\lambda y^\top.1^\delta)^{\top\to\delta} P^\top)^\delta)^\delta)^{\delta\to\delta})^{(\delta\to\delta)\to\top\to\delta}\\ & (\lambda z^\delta.3^\delta)^{\delta\to\delta})^{\delta\to\delta} Q^\delta)^\delta\ .\end{array}$$

*Applying the* $\mathbf{O}_2$ *simplification mapping we get* $\mathbf{O}_2(\Sigma'') \vdash_2 \mathbf{O}_2(M''^{\delta})$, *where* $\mathbf{O}_2(\Sigma'') = \{f : (\delta \to \delta) \to \delta,\ v : \delta\}$ *and* $\mathbf{O}_2(M''^{\delta}) =$

$$\begin{array}{l}(((\lambda g^{\delta\to\delta}.(\lambda x^{\delta}.\\ \qquad (((f^{(\delta\to\delta)\to\delta} g^{\delta\to\delta})^{\delta} +^{\delta\times\delta\to\delta} (g^{\delta\to\delta} x^{\delta})^{\delta})^{\delta} +^{\delta\times\delta\to\delta} 1)^{\delta})^{\delta\to\delta})^{(\delta\to\delta)\to\delta\to\delta}\\ \qquad\qquad (\lambda z^{\delta}.3^{\delta})^{\delta\to\delta})^{\delta\to\delta} Q^{\delta})^{\delta}.\end{array}$$

*Let* $N_2$ *be the term obtained from* $\mathbf{O}_2(\mathrm{FV}(M''^{\delta}))$ *by erasing the d-type decorations, i.e.,* $N_2 = \epsilon(\mathbf{O}_2(\mathrm{FV}(M''^{\delta})))$. *We have* $\vdash_{\mathbf{T}} N_2 : \mathsf{nat}$, $\mathrm{FV}(N_2) = \{f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}}, v^{\mathsf{nat}}\}$ *and*

$$N_2 = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.fg + gx + 1)(\lambda z^{\mathsf{nat}}.3)Q \ .$$

## 7.4 On dead-code detection and elimination

In this section we show that the output of the simplification mapping $\mathbf{O}_1$ of Section 7.2.3 can be further simplified by using the system $\vdash_2$ and the simplification mapping $\mathbf{O}_2$ of Section 7.3.

The dummy variables introduced by the simplification mapping $\mathbf{O}_1$ are just placeholders for some dead code removed. So they can be handled in a special by the system $\vdash_2$. In particular, given a term $M$ obtained by a correct application of the mapping $\mathbf{O}_1$ (see Theorem 7.33), it is safe to allow that any d-type $\phi \in \mathbf{L}^{\mathrm{d}}(\rho)$ can be assigned to a dummy variable $\mathsf{d}^{\rho}$ occurring in $M$. To ensure this we:

- forbid dummy variables to occur in bases, and
- add to the rules in Fig. 7.4 the following rule

$$(\mathsf{d})\ \frac{\vdash_{\mathbf{T}} \mathsf{d}^{\rho} : \rho \qquad \phi \in \mathbf{L}^{\mathrm{d}}(\rho)}{\vdash \mathsf{d}^{\phi}} \quad .$$

For every term $M$, define $\mathrm{FV}'(M)$ be the set of the free non dummy variables in $M$.

**Example 7.54** *Let*

$$M = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.(fg + gx) + (\lambda y^{\mathsf{nat}}.1)P)(\lambda z^{\mathsf{nat}}.3)Q$$

*be the term of Example 7.34.*

*Let* $N_1$ *be the term obtained from* $M$ *as shown in Example 7.34, i.e., by*

- *decorating* $M$ *using the system* $\vdash_1$,
- *applying the simplification mapping* $\mathbf{O}_1$, *and*
- *erasing the d-type decorations.*

*We have* $\vdash_{\mathbf{T}} N_1 : \mathsf{nat}$, $\mathrm{FV}(N_1) = \{f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}}, \mathsf{d}_1^{\mathsf{nat}}, \mathsf{d}_2^{\mathsf{nat}}, \mathsf{d}_3^{\mathsf{nat}}\}$, *and*

$$N_1 = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.fg + g\mathsf{d}_1^{\mathsf{nat}} + (\lambda y^{\mathsf{nat}}.1)\mathsf{d}_2^{\mathsf{nat}})(\lambda z^{\mathsf{nat}}.3)\mathsf{d}_3^{\mathsf{nat}} \ .$$

*It is easy to check that* $\Sigma \vdash_2 N_1'^{\delta^{\mathsf{nat}}}$ *is a faithful* $\vdash_2$*-typing, where (writing, for short,* $\delta$ *and* $\top$ *instead of* $\delta^{\mathsf{nat}}$ *and* $\top^{\mathsf{nat}}$*):* $\Sigma = \{f : (\delta \to \delta) \to \delta\}$ *and* $N_1'^{\delta} =$

$$\begin{array}{ll}(((\lambda g^{\delta\to\delta}. & (\lambda x^{\top}.\\ & (((f^{(\delta\to\delta)\to\delta} g^{\delta\to\delta})^{\delta} +^{\delta\times\delta\to\delta} (g^{\delta\to\delta}\mathsf{d}_1^{\delta})^{\delta})^{\delta}\\ & \qquad +^{\delta\times\delta\to\delta}((\lambda y^{\top}.1^{\delta})^{\top\to\delta}\mathsf{d}_2^{\top})^{\delta})^{\delta\to\delta})^{(\delta\to\delta)\to\top\to\delta}\\ & (\lambda z^{\delta}.3^{\delta})^{\delta\to\delta})^{\delta\to\delta}\mathsf{d}_3^{\top})^{\delta} \ .\end{array}$$

*Applying the* $\mathbf{O}_2$ *simplification mapping we get* $\mathbf{O}_2(\Sigma) \vdash_2 \mathbf{O}_2(N_1'^{\delta})$*, where* $\mathbf{O}_2(\Sigma) = \{f^{(\delta\to\delta)\to\delta}\}$ *and* $\mathbf{O}_2(N_1'^{\delta}) =$

$$\begin{array}{ll}(((\lambda g^{\delta\to\delta}. & \\ & (((f^{(\delta\to\delta)\to\delta} g^{\delta\to\delta})^{\delta} +^{\delta\times\delta\to\delta} (g^{\delta\to\delta}\mathsf{d}_1^{\delta})^{\delta})^{\delta}\\ & \qquad +^{\delta\times\delta\to\delta} 1^{\delta})^{\delta})^{\delta\to\delta})^{(\delta\to\delta)\to\delta\to\delta}\\ & (\lambda z^{\delta}.3^{\delta})^{\delta\to\delta})^{\delta}\end{array}$$

*Let* $N$ *be the term obtained from* $\mathbf{O}_2(N_1'^{\delta})$ *by erasing the d-type decorations, i.e.,* $N = \epsilon(\mathbf{O}_2(\mathrm{FV}(N_1'^{\delta})))$*. We have* $\vdash_{\mathbf{T}} N : \mathsf{nat}$, $\mathrm{FV}(N) = \{f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}}, \mathsf{d}_1^{\mathsf{nat}}\}$ *and*

$$N = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.fg + g\mathsf{d}_1^{\mathsf{nat}} + 1)(\lambda z^{\mathsf{nat}}.3) \ .$$

*Comparing* $N$ *with the term* $N_1$ *above and with the term* $N_2$ *of Example 7.53 we note that this combined use of the simplification mappings* $\mathbf{O}_1$ *and* $\mathbf{O}_2$ *improves the dead-code elimination performed by each of them.*

As we will see in Section 7.6, for every PCFP$^{\mathcal{T}}$ term $M$, there are

1. a faithful $\vdash_1$-typing showing the maximum amount of dead-code (see Theorem 7.70), and

2. a faithful $\vdash_2$-typing showing the maximum amount of dead-code (see Theorem 7.73).

Indeed (as it is easy to check) all the faithful typings considered in Example 7.54 above are faithful typings showing the maximum amount of dead-code.

Let $\mathbf{O}_1^{\vdash_1} : \Lambda_{\mathbf{T}} \to \Lambda_{\mathbf{T}}$ be the mapping that given a PCFP$^{\mathcal{T}}$term $M$ finds a faithful $\vdash_1$-typing showing the maximum amount of dead-code, $\Sigma_1 \vdash_1 M^{\phi_1}$, and then returns $\epsilon(\mathbf{O}_1(M^{\phi_1}))$. Similarly, let $\mathbf{O}_2^{\vdash_2} : \Lambda_{\mathbf{T}} \to \Lambda_{\mathbf{T}}$ be the mapping that given a PCFP$^{\mathcal{T}}$term $M$ finds a faithful $\vdash_2$-typing showing the maximum amount of dead-code, $\Sigma_2 \vdash_2 M^{\phi_2}$, and then returns $\epsilon(\mathbf{O}_2(M^{\phi_1}))$. We have that the following fact holds.

**Fact 7.55** *For every PCFP*$^{\mathcal{T}}$ *term* $M$*,*

1. $\mathbf{O}_1^{\vdash_1}(M) = \mathbf{O}_1^{\vdash_1}(\mathbf{O}_1^{\vdash_1}(M))$,
2. $\mathbf{O}_2^{\vdash_2}(M) = \mathbf{O}_2^{\vdash_2}(\mathbf{O}_2^{\vdash_2}(M))$, *and*
3. $\mathbf{O}_2^{\vdash_2}(\mathbf{O}_1^{\vdash_1}(M)) = \mathbf{O}_1^{\vdash_1}(\mathbf{O}_2^{\vdash_2}(\mathbf{O}_1^{\vdash_1}(M))) = \mathbf{O}_2^{\vdash_2}(\mathbf{O}_1^{\vdash_1}(\mathbf{O}_2^{\vdash_2}(M)))$. □

So we can conclude that the more convenient way of using the dead-code detection and elimination techniques described in Sections 7.2 and 7.3 is to apply first the simplification mapping $\mathbf{O}_1^{\vdash_1}$ and then the simplification mapping $\mathbf{O}_2^{\vdash_2}$ (as done in Example 7.54).

In particular, if we apply the simplification mappings in the reverse order, i.e., first $\mathbf{O}_2^{\vdash_2}$ and then $\mathbf{O}_1^{\vdash_1}$, we may loose some simplification, as the following example shows.

**Example 7.56** *Consider the term of Example 7.54,*

$$M = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.(fg + gx) + (\lambda y^{\mathsf{nat}}.1)P)(\lambda z^{\mathsf{nat}}.3)Q.$$

*It is easy to check that:*

- $\mathbf{O}_1^{\vdash_1}(M) = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.fg + g\mathsf{d}_1^{\mathsf{nat}} + (\lambda y^{\mathsf{nat}}.1)\mathsf{d}_2^{\mathsf{nat}})(\lambda z^{\mathsf{nat}}.3)\mathsf{d}_3^{\mathsf{nat}}$
  *(which is the term $N_1$ in Example 7.54),*
- $\mathbf{O}_2^{\vdash_2}(M) = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.fg + gx + 1)(\lambda z^{\mathsf{nat}}.3)Q$
  *(which is the term $N_2$ in Example 7.34),*
- $\mathbf{O}_2^{\vdash_2}(\mathbf{O}_1^{\vdash_1}(M)) = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.fg + g\mathsf{d}_1^{\mathsf{nat}} + 1)(\lambda z^{\mathsf{nat}}.3)$
  *(which is the term $N$ in Example 7.54), and*
- $\mathbf{O}_1^{\vdash_1}(\mathbf{O}_2^{\vdash_2}(M)) = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.fg + g\mathsf{d}_1^{\mathsf{nat}} + 1)(\lambda z^{\mathsf{nat}}.3)\mathsf{d}_3^{\mathsf{nat}}$.

## 7.5 Some examples of dead-code elimination

The first example that we present is due to C. Mohring and has been further developed by Berardi and Boerio, see [Berardi, 1996] and [Berardi and Boerio, 1995].

**Example 7.57** *Let $\rho = \mathsf{nat} \to \mathsf{nat}$ and $\sigma = \mathsf{nat} \to (\mathsf{nat} \times \mathsf{nat}) \to (\mathsf{nat} \times \mathsf{nat})$. Take term*

$$M \quad = \quad \lambda x^{\mathsf{nat}}.\mathsf{proj}_1(\mathsf{rec}(x, \langle P, Q\rangle, F)),$$

*of type $\rho$ where*

- $F = \lambda n^{\mathsf{nat}}.\lambda m^{\mathsf{nat}\times\mathsf{nat}}.\langle f(\mathsf{proj}_1 m), g(\mathsf{proj}_1 m)(\mathsf{proj}_2 m)\rangle$ *is a term of type $\sigma$, and*
- *$P$, $Q$ are closed terms of type* $\mathsf{nat}$.

*We have* $\mathrm{FV}(M) = \mathrm{FV}(F) = \{f^{\mathsf{nat}\to\mathsf{nat}}, g^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}\}$. *Moreover, it is easy to see that* $\Sigma \vdash_1 M'^{\phi}$ *is a faithful* $\vdash_1$*-typing, where*

$$\begin{array}{l}
\Sigma = \{f : \delta \to \delta, g : \top \to \top \to \top\} \ , \\
\phi = \delta \to \delta \ , \\
M'^{\phi} = (\lambda x^{\delta}.\mathsf{proj}_1(\mathsf{rec}(x^{\delta}, \langle P^{\delta}, Q^{\top}\rangle^{\delta\times\top}, F'^{\psi})^{\delta})^{\delta})^{\phi} \ , \\
\psi = \top \to (\delta \times \top) \to (\delta \times \top) \ , \\
F'^{\psi} = (\lambda n^{\top}.(\lambda m^{\delta\times\top}. \\
\qquad \langle f^{\delta\to\delta}(\mathsf{proj}_1 m^{\delta\times\top})^{\delta}, g^{\top\to\top\to\top}(\mathsf{proj}_1 m^{\top\times\top})^{\top}(\mathsf{proj}_2 m^{\top\times\top})^{\top}\rangle^{\delta\times\top})^{\delta\times\top\to\delta\times\top})^{\psi} \ .
\end{array}$$

*Let* $N$ *and* $G$ *be the terms obtained from* $M'^{\phi}$ *and* $F'^{\psi}$ *by applying the* $\mathbf{O}_1$ *mapping and erasing all the d-type decorations, i.e.,* $N = \epsilon(\mathbf{O}_1(M'^{\phi}))$ *and* $G = \epsilon(\mathbf{O}_1(F'^{\psi}))$. *We have*

$$\begin{array}{rcll}
N & = & \lambda x^{\mathsf{nat}}.\mathsf{proj}_1(\mathsf{rec}(x, \langle P, \mathsf{d}_1^{\mathsf{nat}}\rangle, G)), & \text{where} \\
G & = & \lambda n^{\mathsf{nat}}.\lambda m^{\mathsf{nat}\times\mathsf{nat}}.\langle f(\mathsf{proj}_1 m), \mathsf{d}_2^{\mathsf{nat}}\rangle. &
\end{array}$$

*Moreover* $\vdash_{\mathbf{T}} N : \rho$, $\vdash_{\mathbf{T}} G : \sigma$, *and* $\mathrm{FV}(N) = \mathrm{FV}(G) = \{f^{\mathsf{nat}\to\mathsf{nat}}, \mathsf{d}_1^{\mathsf{nat}}, \mathsf{d}_2^{\mathsf{nat}}\}$. *It is easy to see that* $\Sigma \vdash_2 N'^{\phi}$ *is a faithful* $\vdash_2$*-typing, where*

$$\begin{array}{l}
\Sigma = \{f : \delta \to \delta, \mathsf{d}_1 : \top, \mathsf{d}_2 : \top\} \ , \\
\phi = \delta \to \delta \ , \\
N'^{\phi} = (\lambda x^{\delta}.\mathsf{proj}_1(\mathsf{rec}(x^{\delta}, \langle P^{\delta}, \mathsf{d}_1^{\top}\rangle^{\delta\times\top}, G'^{\psi})^{\delta})^{\delta})^{\phi} \ , \\
\psi = \top \to (\delta \times \top) \to (\delta \times \top) \ , \\
G'^{\psi} = (\lambda n^{\top}.(\lambda m^{\delta\times\top}. \\
\qquad \langle f^{\delta\to\delta}(\mathsf{proj}_1 m^{\delta\times\top})^{\delta}, \mathsf{d}_2^{\top}\rangle)^{\delta\times\top\to\delta\times\top})^{\psi} \ .
\end{array}$$

*Let* $R$ *and* $H$ *be the terms obtained from* $N'^{\phi}$ *and* $G'^{\psi}$ *by applying the* $\mathbf{O}_2$ *mapping and erasing all the d-type decorations, i.e.,* $R = \epsilon(\mathbf{O}_2(N'^{\phi}))$ *and* $h = \epsilon(\mathbf{O}_2(G'^{\psi}))$. *We have*

$$\begin{array}{rcll}
R & = & \lambda x^{\mathsf{nat}}.\mathsf{it}(x, P, H) \ , & \text{where} \\
H & = & \lambda p^{\mathsf{nat}}.fp \ . &
\end{array}$$

*Note that the* $\vdash_1$*-typing* $\Sigma \vdash_1 M'^{\phi}$ *considered above is also a faithful* $\vdash_2$*-typing, i.e., we have that* $\Sigma \vdash_2 M'^{\phi}$. *So by applying directly* $\mathbf{O}_2$ *to* $M'^{\phi}$ *we get the same optimized program* $R$.

The second example involves the construct $\mathsf{fix}$.

**Example 7.58** *Take the term* $F = \mathsf{fix}\, f^{\rho}.M$, *where* $\rho = (\mathsf{nat} \to \mathsf{nat}) \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ *and*

$$M = \lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\lambda z^{\mathsf{nat}}.\mathsf{if}\ z = 0\ \mathsf{then}\ gx\ \mathsf{else}\ f(\lambda u^{\mathsf{nat}}.3)yx(z-1) \quad .$$

*Let* $\phi_1 = (\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$. *It is easy to check that* $\emptyset \vdash_1 (\mathsf{fix}\, f^{\phi_1}.M'^{\phi_1})^{\phi_1}$, *where (writing, for short,* $\delta$ *and* $\top$ *instead of* $\delta^{\mathsf{nat}}$ *and* $\top^{\mathsf{nat}}$*):*

$$\begin{array}{l}
M'^{\phi_1} = (\lambda g^{\top\to\delta}.(\lambda x^{\top}.(\lambda y^{\top}.(\lambda z^{\delta}. \\
\quad (\mathsf{if}\ (z^{\delta} =^{\delta\times\delta\to\delta} 0^{\delta})^{\delta} \\
\qquad \mathsf{then}\ (g^{\top\to\delta}x^{\top})^{\delta} \\
\qquad \mathsf{else}\ ((((f^{\phi_1}(\lambda u^{\top}.3^{\delta})^{\top\to\delta})^{\top\to\top\to\delta\to\delta}y^{\top})^{\top\to\delta\to\delta}x^{\top})^{\delta\to\delta}(z^{\delta} -^{\delta\times\delta\to\delta} 1^{\delta})^{\delta})^{\delta} \\
\quad)^{\delta})^{\delta\to\delta})^{\top\to\delta\to\delta})^{\top\to\top\to\delta\to\delta})^{\phi_1} \ .
\end{array}$$

*So, if we consider the term*

$$P = (\mathsf{fix}\, f^{\rho}.M)(\lambda v^{\mathsf{nat}}.5)Q_1Q_2Q_3,$$

*for some (for simplicity closed) terms $Q_1$, $Q_2$, and $Q_3$, we can derive the following faithful $\vdash_1$-typing of $P$:*

$$\emptyset \vdash_1 (((((\mathsf{fix}\, f^{\phi_1}.M'^{\phi_1})^{\phi_1}(\lambda v^{\top}.5^{\delta})^{\top\to\delta})^{\top\to\top\to\delta\to\delta}Q_1^{\top})^{\top\to\delta\to\delta}Q_2^{\top})^{\delta\to\delta}Q_3^{\delta})^{\delta}. \tag{7.1}$$

*Then by applying the simplification mapping $\mathbf{O}_1$ and erasing all the d-type decorations we get the following simplified version of $P$:*

$$P' = (\mathsf{fix}\, f^{\rho}.M')(\lambda v^{\mathsf{nat}}.5)\mathsf{d}_4^{\mathsf{nat}}\mathsf{d}_5^{\mathsf{nat}}Q_3,$$

*where*

$$M' = \lambda g^{\,\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\,\mathsf{nat}}.\lambda y^{\,\mathsf{nat}}.\lambda z^{\,\mathsf{nat}}.\mathsf{if}\ z = 0\ \mathsf{then}\ g\mathsf{d}_1^{\mathsf{nat}}\ \mathsf{else}\ f(\lambda u^{\mathsf{nat}}.3)\mathsf{d}_2^{\mathsf{nat}}\mathsf{d}_3^{\mathsf{nat}}(z-1) \quad .$$

*Note that the $\vdash_1$-typing (7.1) is also a faithful $\vdash_2$-typing. So we can apply directly $\mathbf{O}_2$ and we get the following simplified version of $P$:*

$$P'' = (\mathsf{fix}\, s^{\mathsf{nat}\to\mathsf{nat}\to\mathsf{nat}}.M'')5Q_3,$$

*where*

$$M'' = \lambda t^{\,\mathsf{nat}}.\lambda z^{\,\mathsf{nat}}.\mathsf{if}\ z = 0\ \mathsf{then}\ t\ \mathsf{else}\ s3(z-1) \quad .$$

## 7.6 A dead-code detection algorithm

In this section we deal with the problem of defining an algorithm for program simplification based on the systems $\vdash_1$ and $\vdash_2$ defined in the previous sections. To this aim the main problem is to use the inference rules to detect the maximal subterms to which $\top$-d-types can be assigned. The application of the simplification mappings $\mathbf{O}_1$ and $\mathbf{O}_2$ is then trivial. The inference algorithm, given a PCFP$^{\mathcal{T}}$ term, returns a decoration of the term with dead-code type patterns and a set of constraints involving basic property constructor variables. The output of the algorithm characterizes all the possible $\vdash_1$-typings and $\vdash_2$-typings of the term.

We start by defining the notions of d-type pattern and d-type scheme.

### 7.6.1 Dead-code type schemes

**Definition 7.59 (Dead-code type patterns)** *Let $\mathcal{A}$ be the set of* basic property constructor variables *(*constructor variables *for short), ranged by $\alpha$, $\beta$, $\gamma$, .... The language* $\mathbf{P}$ *of* d-type patterns *(*d-patterns *for short), ranged over by $\theta$, $\eta$,..., is defined from the grammar of Definition 7.1 by replacing $a \in \{\delta, \top\}$ by $\alpha \in \mathcal{A}$, i.e., $\theta ::= \alpha^{\iota} \mid \theta \to \theta \mid \theta \times \theta$, where $\alpha \in \mathcal{A}$ and $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$.*

The constructor variables can be renamed or instantiated to basic property constructors.

**Definition 7.60 (Renamings and instantiations)** *1. A* renaming *is a one–to–one mapping* $\mathbf{r} : \mathcal{A} \to \mathcal{A}$.

*2. An* instantiation *is a mapping* $\mathbf{i} : \mathcal{A} \to \{\delta, \top\}$.

Both renaming and instantiation can be extended to basic property constructors (by defining $\mathbf{i}(a) = a$ and $\mathbf{r}(a) = a$, for $a \in \{\delta, \top\}$) and to d-types and d-patterns (in the obvious way). For example: $\mathbf{i}(\alpha^{\mathsf{nat}} \to \beta^{\mathsf{nat}}) = \mathbf{i}(\alpha)^{\mathsf{nat}} \to \mathbf{i}(\beta)^{\mathsf{nat}}$. Of course, for any d-type $\phi \in \mathbf{L}^{\mathrm{d}}$, $\mathbf{i}(\phi) = \phi$ and $\mathbf{r}(\phi) = \phi$.

**Definition 7.61 (D-constraints)** *A* d-constraint *is a formula of one of the following shapes:*

- $\zeta_1 \equiv \zeta_2$
- $\zeta_1 \sqsubseteq \zeta_2$
- $(\delta \ \mathsf{in}\ \mathcal{G}) \Rightarrow \mathcal{E}$

*where* $\zeta_1, \zeta_2 \in \{\delta\} \cup \mathcal{A}$, $\mathcal{G}$ *is a finite not empty subset of* $\{\delta\} \cup \mathcal{A}$ *and* $\mathcal{E}$ *is a finite set of d-constraints.*

The symbol $\equiv$ denotes the equality on the set of basic property constructors $\{\delta, \top\}$, while $\sqsubseteq$ denotes the order relation defined by: $\delta \sqsubseteq \delta$, $\delta \sqsubseteq \top$ and $\top \sqsubseteq \top$. A d-constraint is simply an equality or an inequality (between constructor variables or the constant $\delta$) or a guarded set of d-constraints. The meaning of the d-constraints is given by the satisfaction relation that follows.

**Definition 7.62** *An instantiation* $\mathbf{i}$ *satisfies a set of constraints* $\mathcal{E}$ *if*

- $\zeta_1 \equiv \zeta_2 \in \mathcal{E}$ *implies* $\mathbf{i}(\zeta_1) \equiv \mathbf{i}(\zeta_2)$ *(that is either* $\mathbf{i}(\zeta_1) = \mathbf{i}(\zeta_2) = \delta$ *or* $\mathbf{i}(\zeta_1) = \mathbf{i}(\zeta_2) = \top$*), and*
- $\zeta_1 \sqsubseteq \zeta_2 \in \mathcal{E}$ *implies* $\mathbf{i}(\zeta_1) \sqsubseteq \mathbf{i}(\zeta_2)$ *(that is either* $\mathbf{i}(\zeta_1) = \delta$ *or* $\mathbf{i}(\zeta_2) = \top$*), and*
- $(\delta \ \mathsf{in}\ \mathcal{G}) \Rightarrow \mathcal{E}' \in \mathcal{E}$ *implies that, if* $\delta \in \mathbf{i}(\mathcal{G})$*, then* $\mathbf{i}$ *satisfies* $\mathcal{E}'$.

The set of all the instantiations that satisfy $\mathcal{E}$ is denoted by $\mathbf{sat}(\mathcal{E})$.

**Definition 7.63 (D-type schemes)** *A* d-type scheme *(*d-scheme *for short) is a pair* $\langle \theta, \mathcal{E} \rangle$ *where* $\theta$ *is a d-pattern and* $\mathcal{E}$ *is a finite set of d-constraints.*

A d-scheme $\langle \theta, \mathcal{E} \rangle$ represents the set of d-types that can be obtained from the d-pattern $\theta$ by replacing constructor variables with basic property constructors in such a way that the d-constraints in $\mathcal{E}$ are satisfied, i.e., it represents all the d-types $\mathbf{i}(\theta)$, for any $\mathbf{i} \in \mathbf{sat}(\mathcal{E})$.

$$(\text{STOP})\ \frac{\text{no other rule can be applied}}{\mathcal{E} \rightsquigarrow \emptyset} \qquad (\Rightarrow)\ \frac{\mathcal{E} \cup \mathcal{E}' \rightsquigarrow \mathcal{I}}{\mathcal{E},\ (\delta \ \mathsf{in}\ \mathcal{G}) \Rightarrow \mathcal{E}' \rightsquigarrow \mathcal{I}}\ \delta \in \mathcal{G}$$

$$(\equiv)\ \frac{\{\zeta_1, \zeta_2\} = \{\alpha, \delta\} \quad \mathcal{E}[\delta/\alpha] \rightsquigarrow \mathcal{I}}{\mathcal{E},\ \zeta_1 \equiv \zeta_2 \rightsquigarrow \mathcal{I} \cup \{\alpha\}} \qquad (\sqsubseteq)\ \frac{\mathcal{E}[\delta/\alpha] \rightsquigarrow \mathcal{I}}{\mathcal{E},\ \alpha \sqsubseteq \delta \rightsquigarrow \mathcal{I} \cup \{\alpha\}}$$

Figure 7.9: "Natural semantics" rules for d-constraints solution (system $\rightsquigarrow$)

**Definition 7.64** *Let* $\mathbf{i}_1$, $\mathbf{i}_2$ *be instantiations. We write* $\mathbf{i}_1 \sqsubseteq \mathbf{i}_2$ *if, for all* $\alpha \in \mathcal{A}$, $\mathbf{i}_1(\alpha) \sqsubseteq \mathbf{i}_2(\alpha)$.

**Fact 7.65** *Let* $\mathcal{E}$ *be a finite set of d-constraints. The set* $\mathbf{sat}(\mathcal{E})$ *is not empty and has a maximum element.* □

**Example 7.66** *Consider the set of d-constraints:*

$$\begin{array}{rcll} \mathcal{E} & = & \{ & \alpha_1 \sqsubseteq \delta, \\ & & & (\delta \ \mathsf{in}\ \{\alpha_1\}) \Rightarrow \{\alpha_1 \equiv \alpha_2, \alpha_3 \sqsubseteq \alpha_4\}, \\ & & & (\delta \ \mathsf{in}\ \{\alpha_3, \alpha_5\}) \Rightarrow \{\alpha_4 \sqsubseteq \delta, \alpha_6 \equiv \delta\}, \\ & & & (\delta \ \mathsf{in}\ \{\alpha_2\}) \Rightarrow \{\alpha_5 \sqsubseteq \alpha_6, \alpha_7 \equiv \delta\} \qquad \} \ . \end{array}$$

*To find the maximum element* $\mathbf{i}$ *of* $\mathbf{sat}(\mathcal{E})$ *observe that from the first constraint of* $\mathcal{E}_1$ *we get* $\mathbf{i}(\alpha_1) = \delta$. *Then from the second constraint we get* $\mathbf{i}(\alpha_2) = \delta$, *and finally from the last constraint of* $\mathcal{E}$ *we have* $\mathbf{i}(\alpha_7) = \delta$. *Let* $\mathcal{I} = \{\alpha_1, \alpha_2, \alpha_7\}$, *then* $\mathbf{i}$ *defined by:* $\mathbf{i}(\alpha) = \delta$ *if* $\alpha \in \mathcal{I}$ *and* $\mathbf{i}(\alpha) = \top$ *otherwise, is the maximum instantiation in* $\mathbf{sat}(\mathcal{E})$.

Dead-code inference of a term is reduced to the solution of a finite set of constraints. The maximal instantiation satisfying the d-constraints then corresponds to a decoration that shows the maximal amount of dead-code. The algorithm for finding the maximal instantiation $\mathbf{i}$ that satisfies a finite set of d-constraints $\mathcal{E}$ is presented in natural semantics style using judgements $\mathcal{E} \rightsquigarrow \mathcal{I}$, where $\mathcal{I}$ is the set of constructor variables that *represents* $\mathbf{i}$, i.e., such that $\mathbf{i}(\alpha) = \delta$ if $\alpha \in \mathcal{I}$ and $\mathbf{i}(\alpha) = \top$ otherwise. The idea is simply that of recognizing, following the equalities and the inequalities, all the constructor variables that are forced to be instantiated to $\delta$. All other constructor variables are then replaced by $\top$ in the maximal solution.

**Definition 7.67 (D-constraints solution)** *Let* $\mathcal{E}$ *be a finite set of d-constraints, and let* cns *be a d-constraint. The expression* $\mathcal{E}, \text{cns}$ *denotes the set of constraints* $\mathcal{E} \cup \{\text{cns}\}$ *where it is assumed that* $\text{cns} \notin \mathcal{E}$. *We write* $\mathcal{E} \rightsquigarrow \mathcal{I}$ *to mean that this judgement is derivable by the rules in Fig. 7.9.*

**Theorem 7.68 (Correctness of algorithm $\rightsquigarrow$)** *Let* $\mathcal{E}$ *be a finite set of d-constraints. Then* $\mathcal{E} \rightsquigarrow \mathcal{I}$ *if and only if* $\mathcal{I}$ *represents the maximum of* $\mathbf{sat}(\mathcal{E})$. □

$$\mathbf{cs}_{=}({\zeta_1}^{\iota}, {\zeta_2}^{\iota}) = \{\zeta_1 \equiv \zeta_2\}, \ \textit{where } \zeta_1, \zeta_2 \in \{\delta\} \cup \mathcal{A}$$

$$\mathbf{cs}_{=}(\theta_1 \to \eta_1, \theta_2 \to \eta_2) = \mathbf{cs}_{=}(\theta_1, \theta_2) \cup \mathbf{cs}_{=}(\eta_1, \eta_2)$$

$$\mathbf{cs}_{=}(\theta_1 \times \eta_1, \theta_2 \times \eta_2) = \mathbf{cs}_{=}(\theta_1, \theta_2) \cup \mathbf{cs}_{=}(\eta_1, \eta_2)$$

Figure 7.10: Function $\mathbf{cs}_{=}$

### 7.6.2 An algorithm for inferring $\vdash_1$-typings

To define the algorithm we need some preliminary notations.

- Let $\rho$ be a type. By $\mathbf{fresh}(\rho)$ we denote a d-pattern obtained from $\rho$ by assigning a fresh constructor variable to each occurrence of any ground type in $\rho$. For example: $\mathbf{fresh}(\mathsf{nat} \to \mathsf{nat}) = \alpha^{\mathsf{nat}} \to \beta^{\mathsf{nat}}$. For a set of term variables $\Gamma$, $\mathbf{fresh}(\Gamma) = \{x : \mathbf{fresh}(\rho) \mid x^{\rho} \in \Gamma\}$.

- The function **vars** maps an d-pattern $\theta$ to its finite set of constructor variables. For instance, $\mathbf{vars}(\alpha^{\mathsf{nat}} \to \beta^{\mathsf{nat}}) = \{\alpha, \beta\}$.

- The function **tail**, that maps d-patterns and d-types (not containing $\top$) to finite subsets of $\{\delta\} \cup \mathcal{A}$, is inductively defined by: $\mathbf{tail}({\zeta}^{\iota}) = \{\zeta\}$ (for $\zeta \in \{\delta\} \cup \mathcal{A}$), $\mathbf{tail}(\theta \times \eta) = \mathbf{tail}(\theta) \cup \mathbf{tail}(\eta)$, and $\mathbf{tail}(\theta \to \eta) = \mathbf{tail}(\eta)$.

- Let $\theta$, $\theta'$ be d-patterns or d-types (not containing $\top$) such that $\epsilon(\theta) = \epsilon(\theta')$. Then $\mathbf{cs}_{=}(\theta, \theta')$, $\mathbf{cs}_{\leq_1}(\theta, \theta')$, and $\mathbf{ucs}_{\leq_1}(\theta, \theta')$ denote the sets of constraints inductively defined by the clauses in Fig. 7.10, 7.11, and 7.12. We have that for all instantiations $\mathbf{i}$:

  - $\mathbf{i}(\theta) = \mathbf{i}(\eta)$ if and only if $\mathbf{i} \in \mathbf{sat}(\mathbf{cs}_{=}(\theta, \eta))$,
  - $\mathbf{i}(\theta) \leq_1 \mathbf{i}(\eta)$ if and only if $\mathbf{i} \in \mathbf{sat}(\mathbf{cs}_{\leq_1}(\theta, \eta))$, and
  - $\mathbf{i}(\theta_2) \notin \mathbf{L}^{\mathrm{d}}_{\top}$ implies: $\mathbf{i}(\theta_1) \leq \mathbf{i}(\theta_2)$ if and only if $\mathbf{i} \in \mathbf{sat}(\mathbf{ucs}_{\leq_1}(\theta_1, \theta_2))$.

  In particular $\mathbf{cs}_{\leq_1}(\theta_1, \theta_2)$ consists of the single d-constraint $(\delta \,\mathsf{in}\, \mathbf{tail}(\theta_2)) \Rightarrow \mathbf{ucs}_{\leq_1}(\theta_1, \theta_2)$, which is trivially satisfied by all the instantiations $\mathbf{i}$ such that $\mathbf{i}(\mathbf{tail}(\theta_2)) = \{\top\}$, i.e., such that $\mathbf{i}(\theta_2) \in \mathbf{L}^{\mathrm{d}}_{\top}$.

- For each constant $\mathsf{c}$ a d-scheme $\mathbf{sts}(\mathsf{c})$ is specified. For example, for any integer $\mathsf{n}$, $\mathbf{sts}(\mathsf{n}) = \langle \alpha^{\mathsf{nat}}, \emptyset \rangle$ and $\mathbf{sts}(+) = \langle {\alpha_1}^{\mathsf{nat}} \times {\alpha_2}^{\mathsf{nat}} \to \beta^{\mathsf{nat}}, \{(\delta \,\mathsf{in}\, \{\beta\}) \Rightarrow \{\{\alpha_1 \equiv \delta,\, \alpha_2 \equiv \delta\}\}\} \rangle$.

We can now proceed to define the dead-code inference algorithm $\mathcal{W}_1$. This algorithm is presented in Figs. 7.13, 7.14, and 7.15. Let $\vdash_{\mathbf{T}} M : \rho$, if $\mathcal{W}_1(M) = \langle \Theta, {M'}^{\theta}, \mathcal{E} \rangle$ then $\Theta$ is a basis that associates to each term variable in $\mathrm{FV}(M)$ a d-pattern, ${M'}^{\theta}$ is a term

$$\mathbf{cs}_{\leq_1}({\zeta_1}^{\iota}, {\zeta_2}^{\iota}) = \{\zeta_1 \sqsubseteq \zeta_2\}, \text{ where } \zeta_1, \zeta_2 \in \{\delta\} \cup \mathcal{A}$$

$$\mathbf{cs}_{\leq_1}(\theta'_1 \to \cdots \to \theta'_n \to \eta', \theta''_1 \to \cdots \to \theta''_n \to \eta'') = \{(\delta \text{ in } \mathbf{tail}(\eta'')) \Rightarrow (\mathbf{cs}_{\leq_1}(\eta', \eta'') \cup \bigcup_{1\leq i\leq n} \mathbf{cs}_{\leq_1}(\theta''_i, \theta'_i))\},$$
*where $n \geq 1$ and $\eta', \eta''$ are not arrow d-patterns or arrow d-types.*

$$\mathbf{cs}_{\leq_1}(\theta_1 \times \eta_1, \theta_2 \times \eta_2) = \mathbf{cs}_{\leq_1}(\theta_1, \theta_2) \cup \mathbf{cs}_{\leq_1}(\eta_1, \eta_2)$$

Figure 7.11: Function $\mathbf{cs}_{\leq_1}$

$$\mathbf{ucs}_{\leq_1}(\theta'_1 \to \cdots \to \theta'_n \to \eta', \theta''_1 \to \cdots \to \theta''_n \to \eta'') = \mathbf{ucs}_{\leq_1}(\eta', \eta'') \cup \bigcup_{1\leq i\leq n} \mathbf{cs}_{\leq_1}(\theta''_i, \theta'_i),$$
*where $n \geq 0$ and $\eta', \eta''$ are not arrow d-patterns or arrow d-types.*

$$\mathbf{ucs}_{\leq_1}(\theta_1 \times \eta_1, \theta_2 \times \eta_2) = \mathbf{cs}_{\leq_1}(\theta_1, \theta_2) \cup \mathbf{cs}_{\leq_1}(\eta_1, \eta_2)$$

Figure 7.12: Function $\mathbf{ucs}_{\leq_1}$

decorated with d-patterns, and $\mathcal{E}$ is a finite set of d-constraints. We will prove that the triple $\langle \Theta, {M'}^{\theta}, \mathcal{E} \rangle$ represents all the $\vdash_1$-typings of $M$. More precisely, for any $\Sigma$ and ${M''}^{\phi}$ such that $\epsilon(\Sigma) = \mathrm{FV}(M)$ and $\epsilon({M''}^{\phi}) = M$, we have that $\Sigma \vdash_1 {M''}^{\phi}$ implies $\Sigma = \mathbf{i}_1(\Theta)$ and ${M''}^{\phi} = \mathbf{i}_1({M'}^{\theta})$, for some $\mathbf{i}_1$ that satisfies $\mathcal{E}$.

Correctness and completeness (w.r.t. the $\vdash_1$-typings containing as many $\top$ constructors as possible, i.e., showing a maximal amount of dead-code) of the inference are expressed by the following lemma.

**Lemma 7.69** *$\vdash_{\mathbf{T}} M : \rho$ and $\mathcal{W}_1(M) = \langle \Theta, {M'}^{\theta}, \mathcal{E} \rangle$ imply*

1. *for all instantiations $\mathbf{i}$, if $\mathbf{i}$ is the maximum of $\mathbf{sat}(\mathcal{E})$, then $\mathbf{i}(\Theta) \vdash_1 \mathbf{i}({M'}^{\theta})$,*
2. *for all $\Sigma$ and ${M''}^{\phi}$ such that $\epsilon(\Sigma) = \mathrm{FV}(\epsilon({M''}^{\phi}))$ and $\epsilon({M''}^{\phi}) = M$, if $\Sigma \vdash_1 {M''}^{\phi}$ then exists $\mathbf{i} \in \mathbf{sat}(\mathcal{E})$ such that $\mathbf{i}(\Theta) = \Sigma$ and $\mathbf{i}({M'}^{\theta}) = {M''}^{\phi}$.* □

$$\begin{array}{lll} \mathcal{W}_1(P) = & \text{let} & \Theta = \mathbf{fresh}(\mathrm{FV}(P)) \\ & & \text{and } \langle P', \mathcal{E} \rangle = \mathcal{W}(\Theta, P) \\ & \text{in} & \langle \Theta, P', \mathcal{E} \rangle \text{ end} \end{array}$$

Figure 7.13: Algorithm $\mathcal{W}_1$

$$
\begin{array}{lll}
\multicolumn{3}{l}{\mathcal{W}(\Theta, P) = \text{case } P \text{ of}} \\
\quad \mathsf{c}: & \text{let} & \langle \theta, \mathcal{E} \rangle = \mathbf{sts}(\mathsf{c}) \\
& \text{in} & \langle \mathsf{c}^{\theta}, \mathcal{E} \rangle \ \text{end} \\
\quad \mathsf{d}^{\rho}: & & \langle \mathsf{d}^{\mathbf{fresh}(\rho)}, \emptyset \rangle \\
\quad x: & \text{let} & \theta_1 = \Theta(x) \quad \text{and } \theta_2 = \mathbf{fresh}(\epsilon(\theta_1)) \\
& \text{in} & \langle x^{\theta_2}, (\delta \in \mathbf{tail}(\theta_2)) \Rightarrow \mathbf{ucs}_{\leq_1}(\theta_1, \theta_2) \rangle \ \text{end} \\
\quad \lambda\, x^{\rho}.M: & \text{let} & \theta = \mathbf{fresh}(\rho) \\
& & \text{and } \langle M'^{\eta}, \mathcal{E} \rangle = \mathcal{W}(\Theta \cup \{x : \theta\}, M) \\
& \text{in} & \langle (\lambda\, x^{\theta}.M'^{\eta})^{\theta \to \eta}, \mathcal{E} \rangle \ \text{end} \\
\quad MN: & \text{let} & \langle M'^{\theta_1 \to \eta}, \mathcal{E}_1 \rangle = \mathcal{W}(\Theta, M) \\
& & \text{and } \langle N'^{\theta_2}, \mathcal{E}_2 \rangle = \mathcal{W}(\Theta, N) \\
& \text{in} & \langle (M'^{\theta_1 \to \eta} N'^{\theta_2})^{\eta}, \{(\delta \ \mathsf{in}\ \mathbf{tail}(\eta)) \Rightarrow (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathbf{cs}_{\leq_1}(\theta_2, \theta_1))\} \rangle \ \text{end} \\
\quad \langle M_1, M_2 \rangle: & \text{let} & \langle M_1'^{\theta_1}, \mathcal{E}_1 \rangle = \mathcal{W}(\Theta, M_1) \\
& & \text{and } \langle M_2'^{\theta_2}, \mathcal{E}_2 \rangle = \mathcal{W}(\Theta, M_2) \\
& \text{in} & \langle (\langle M_1'^{\theta_1}, M_2'^{\theta_2} \rangle)^{\theta_1 \times \theta_2}, \mathcal{E}_1 \cup \mathcal{E}_2 \rangle \ \text{end} \\
\quad \mathsf{proj}_i M: & \text{let} & \langle M'^{\theta_1 \times \theta_2}, \mathcal{E} \rangle = \mathcal{W}(\Theta, M) \\
& \text{in} & \langle (\mathsf{proj}_i M'^{\theta_1 \times \theta_2})^{\theta_i}, \mathcal{E} \rangle \ \text{end} \\
\quad \mathsf{fix}\ x^{\rho}.M: & & \\
& \text{let} & \theta = \mathbf{fresh}(\rho) \\
& & \text{and } \langle M'^{\eta}, \mathcal{E} \rangle = \mathcal{W}(\Theta \cup \{x : \theta\}, M) \\
& \text{in} & \langle (\mathsf{fix}\, x^{\eta}.M'^{\eta})^{\eta}, \{(\delta \ \mathsf{in}\ \mathbf{tail}(\eta)) \Rightarrow (\mathcal{E} \cup \mathbf{cs}_{=}(\theta, \eta))\} \rangle \ \text{end} \\
\quad \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2: & & \\
& \text{let} & \langle N'^{\alpha^{\mathsf{bool}}}, \mathcal{E}_0 \rangle = \mathcal{W}(\Theta, N) \\
& & \text{and } \langle M_1'^{\theta_1}, \mathcal{E}_1 \rangle = \mathcal{W}(\Theta, M_1) \\
& & \text{and } \langle M_2'^{\theta_2}, \mathcal{E}_2 \rangle = \mathcal{W}(\Theta, M_2) \\
& & \text{and } \theta = \mathbf{fresh}(\epsilon(\theta_1)) \\
& \text{in} & \langle (\mathsf{if}\ N'^{\alpha^{\mathsf{bool}}}\ \mathsf{then}\ M_1'^{\theta_1}\ \mathsf{else}\ M_2'^{\theta_2})^{\theta}, \\
& & \{(\delta \ \mathsf{in}\ \mathbf{tail}(\theta)) \Rightarrow (\{\alpha = \delta\} \cup \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathbf{ucs}_{\leq_1}(\theta_1, \theta) \cup \mathbf{ucs}_{\leq_1}(\theta_2, \theta))\} \rangle \ \text{end}
\end{array}
$$

Figure 7.14: Algorithm $\mathcal{W}$ (continue in Fig. 7.15)

$\mathsf{case}(N, M, F)$ :

$$\begin{array}{ll}
& \text{let} \quad \langle N'^{\alpha_0\,\mathsf{nat}}, \mathcal{E}_0\rangle = \mathcal{W}(\Theta, N) \\
& \quad \text{and} \ \langle M'^{\theta_1}, \mathcal{E}_1\rangle = \mathcal{W}(\Theta, M) \\
& \quad \text{and} \ \langle F'^{\alpha_1\,\mathsf{nat}\to\theta_2}, \mathcal{E}_2\rangle = \mathcal{W}(\Theta, F) \\
& \quad \text{and} \ \theta = \mathbf{fresh}(\epsilon(\theta_1)) \\
& \text{in} \quad \langle(\mathsf{case}(N'^{\alpha_0\,\mathsf{nat}}, M'^{\theta_1}, F'^{\alpha_1\,\mathsf{nat}\to\theta_2})^{\theta}, \\
& \quad \{(\delta \ \mathsf{in}\ \mathbf{tail}(\theta)) \Rightarrow (\{\alpha_0 = \delta\} \cup \mathcal{E}_0 \cup \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathbf{ucs}_{\leq_1}(\theta_1, \theta) \cup \\
& \qquad\qquad \mathbf{ucs}_{\leq_1}(\alpha_1{}^{\mathsf{nat}} \to \theta_2, \delta^{\mathsf{nat}} \to \theta))\}\rangle \ \ \text{end}
\end{array}$$

$\mathsf{it}(N, M, F)$ :

$$\begin{array}{ll}
& \text{let} \quad \langle N'^{\alpha_0\,\mathsf{nat}}, \mathcal{E}_0\rangle = \mathcal{W}(\Theta, N) \\
& \quad \text{and} \ \langle M'^{\theta_1}, \mathcal{E}_1\rangle = \mathcal{W}(\Theta, M) \\
& \quad \text{and} \ \langle F'^{\theta_2\to\theta_3}, \mathcal{E}_2\rangle = \mathcal{W}(\Theta, F) \\
& \quad \text{and} \ \theta = \mathbf{fresh}(\epsilon(\theta_1)) \\
& \text{in} \quad \langle(\mathsf{it}(N'^{\alpha_0\,\mathsf{nat}}, M'^{\theta_1}, F'^{\theta_2\to\theta_3})^{\theta}, \\
& \quad \{(\delta \ \mathsf{in}\ \mathbf{tail}(\theta)) \Rightarrow (\{\alpha_0 = \delta\} \cup \mathcal{E}_0 \cup \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathbf{ucs}_{\leq_1}(\theta_1, \theta) \cup \\
& \qquad\qquad \mathbf{ucs}_{\leq_1}(\theta_2 \to \theta_3, \theta \to \theta))\}\rangle \ \ \text{end}
\end{array}$$

$\mathsf{rec}(N, M, F)$ :

$$\begin{array}{ll}
& \text{let} \quad \langle N'^{\alpha_0\,\mathsf{nat}}, \mathcal{E}_0\rangle = \mathcal{W}(\Theta, N) \\
& \quad \text{and} \ \langle M'^{\theta_1}, \mathcal{E}_1\rangle = \mathcal{W}(\Theta, M) \\
& \quad \text{and} \ \langle F'^{\alpha_1\,\mathsf{nat}\to\theta_2\to\theta_3}, \mathcal{E}_2\rangle = \mathcal{W}(\Theta, F) \\
& \quad \text{and} \ \theta = \mathbf{fresh}(\epsilon(\theta_1)) \\
& \text{in} \quad \langle(\mathsf{rec}(N'^{\alpha_0\,\mathsf{nat}}, M'^{\theta_1}, F'^{\alpha_1\,\mathsf{nat}\to\theta_2\to\theta_3})^{\theta}, \\
& \quad \{(\delta \ \mathsf{in}\ \mathbf{tail}(\theta)) \Rightarrow (\{\alpha_0 = \delta\} \cup \mathcal{E}_0 \cup \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathbf{ucs}_{\leq_1}(\theta_1, \theta) \cup \\
& \qquad\qquad \mathbf{ucs}_{\leq_1}(\alpha_1{}^{\mathsf{nat}} \to \theta_2 \to \theta_3, \delta^{\mathsf{nat}} \to \theta \to \theta))\}\rangle \ \ \text{end}
\end{array}$$

Figure 7.15: Algorithm $\mathcal{W}$ (continue from Fig. 7.14)

We are interested in finding a faithful $\vdash_1$-typing showing the maximum amount of dead-code, i.e., all the dead-code that can be detected by using system $\vdash_1$. This can be done as shown by the following theorem.

**Theorem 7.70** *Let* $\vdash_{\mathbf{T}} M : \rho$, $\mathcal{W}(M) = \langle \Theta, M'^{\theta}, \mathcal{E} \rangle$ *and* $\mathbf{faithful}(\Theta, \theta) =$

$$\bigcup_{x:\eta \in \Theta} \{(\delta \text{ in } \mathbf{tail}(\eta)) \Rightarrow \{\gamma \equiv \delta | \gamma \in \mathbf{vars}(\eta)\}\} \ \cup \ \{\alpha \equiv \delta | \alpha \in \mathbf{vars}(\theta)\} \ .$$

*If* $\mathbf{i}$ *is the maximum of* $\mathbf{sat}(\mathcal{E} \cup \mathbf{faithful}(\Theta, \theta))$ *then* $\mathbf{i}(\Theta) \vdash_1 \mathbf{i}(M'^{\theta})$ *is a faithful* $\vdash_1$*-assignment showing the maximum amount of dead-code.* □

The constraint $(\delta \text{ in } \mathbf{tail}(\eta)) \Rightarrow \{\gamma \equiv \delta | \gamma \in \mathbf{vars}(\eta)\}$ forces $\eta$ to be instantiated either to an $\top$-d-type or to a $\delta$-d-type.

**Example 7.71** *Let*

$$M = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.(fg + gx) + (\lambda y^{\mathsf{nat}}.1)P)(\lambda z^{\mathsf{nat}}.3)Q$$

*be the term of Example 7.34.*

Assume for simplicity $P = u^{\mathsf{nat}}$ and $Q = v^{\mathsf{nat}}$.

*We have* $\mathcal{W}_1(M) = \langle \Theta_1, M''^{\gamma_6}, \mathcal{E}_1 \rangle$*, where (writing, for short,* $\zeta$ *instead of* $\zeta^{\mathsf{nat}}$*):* $\Theta_1 = \{f : (\alpha_1 \to \alpha_2) \to \alpha_3,\ u : \alpha_4,\ v : \alpha_5\}$, $M''^{\gamma_6} =$

$$\begin{array}{l}(((\lambda g^{\beta_1\to\beta_2}.(\lambda x^{\beta_3}. \\ \qquad (((f^{(\alpha_1'\to\alpha_2')\to\alpha_3'} g^{\beta_1'\to\beta_2'})^{\alpha_3'} +^{\gamma_1\times\gamma_2\to\gamma_3} (g^{\beta_1''\to\beta_2''} x^{\beta_3'})^{\beta_2''})^{\gamma_3} \\ \qquad\qquad +^{\gamma_4\times\gamma_5\to\gamma_6}((\lambda y^{\beta_4}.1^{\gamma_7})^{\beta_4\to\gamma_7} u^{\alpha_4'})^{\gamma_7})^{\gamma_6})^{\beta_3\to\gamma_6})^{(\beta_1\to\beta_2)\to\beta_3\to\gamma_6} \\ \qquad\qquad\qquad (\lambda z^{\beta_5}.3^{\gamma_8})^{\beta_5\to\gamma_8})^{\beta_3\to\gamma_6} v^{\alpha_5'})^{\gamma_6} \ ,\end{array}$$

*and* $\mathcal{E}_1$ *is as in Fig. 7.16. We also have* $\mathbf{faithful}(\Theta_1, \gamma_6) =$

$$\{ \ (\delta \text{ in } \{\alpha_3\}) \Rightarrow \{\alpha_1 \equiv \delta, \alpha_2 \equiv \delta\}, \ \ (\delta \text{ in } \{\alpha_4\}) \Rightarrow \{\alpha_4 \equiv \delta\}, \ \ \gamma_6 \equiv \delta \ \} \ .$$

*Applying the algorithm of Fig. 7.9 we get* $\mathcal{E}_1 \cup \mathbf{faithful}(\Theta_1, \gamma_3) \rightsquigarrow \mathcal{I}_1$*, where*

$$\begin{array}{rcll} \mathcal{I}_1 & = & \{ & \alpha_1, \alpha_2, \alpha_3, \alpha_2', \alpha_3', \\ & & & \beta_2, \beta_2', \beta_2'', \\ & & & \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6, \gamma_7, \gamma_8 \ \ \} \ . \end{array}$$

*Note that, if* $\mathbf{i}_1$ *is the instantiation represented by* $\mathcal{I}_1$*, then* $\mathbf{i}_1(\Theta) \vdash_1 \mathbf{i}_1(M''^{\gamma_3})$ *is the faithful* $\vdash_1$*-typing* $\Sigma_1 \vdash_1 M'^{\delta}$ *of Example 7.34.*

$$\mathcal{E}_1 \;=\; \{(\delta \;\mathsf{in}\; \{\gamma_6\}) \Rightarrow (\mathcal{E}_{1,1} \cup \mathcal{E}_{1,2} \cup \{\alpha'_5 \sqsubseteq \beta_3\})\}$$

where

$$\begin{array}{rcl}
\mathcal{E}_{1,1} & = & \{(\delta \;\mathsf{in}\; \{\gamma_6\}) \Rightarrow (\mathcal{E}_{1,3} \cup \mathcal{E}_{1,4} \cup \mathbf{cs}_{\leq_1}(\beta_5 \to \gamma_8,\ \beta_1 \to \beta_2))\} \\
\quad \mathcal{E}_{1,3} & = & \{(\delta \;\mathsf{in}\; \{\gamma_6\}) \Rightarrow (\mathcal{E}_{1,5} \cup \mathcal{E}_{1,6} \cup \mathbf{cs}_{\leq_1}(\gamma_3 \times \gamma_7,\ \gamma_4 \times \gamma_5))\} \\
\quad\quad \mathcal{E}_{1,5} & = & \{(\delta \;\mathsf{in}\; \{\gamma_3\}) \Rightarrow (\mathcal{E}_{1,7} \cup \mathcal{E}_{1,8} \cup \mathbf{cs}_{\leq_1}(\alpha'_3 \times \beta''_2,\ \gamma_1 \times \gamma_2))\} \\
\quad\quad\quad \mathcal{E}_{1,7} & = & \{(\delta \;\mathsf{in}\; \{\gamma_3\}) \Rightarrow \{\gamma_1 \equiv \delta,\ \gamma_2 \equiv \delta\}\} \\
\quad\quad\quad \mathcal{E}_{1,8} & = & \mathcal{E}_{1,9} \cup \mathcal{E}_{1,10} \\
\quad\quad\quad\quad \mathcal{E}_{1,9} & = & \{(\delta \;\mathsf{in}\; \{\alpha'_3\}) \Rightarrow (\mathcal{E}_{1,11} \cup \mathcal{E}_{1,12} \cup \mathbf{cs}_{\leq_1}(\beta'_1 \to \beta'_2,\ \alpha'_1 \to \alpha'_2))\} \\
\quad\quad\quad\quad\quad \mathcal{E}_{1,11} & = & \mathbf{ucs}_{\leq_1}((\alpha_1 \to \alpha_2) \to \alpha_3,\ (\alpha'_1 \to \alpha'_2) \to \alpha'_3) \\
\quad\quad\quad\quad\quad \mathcal{E}_{1,12} & = & \{(\delta \;\mathsf{in}\; \{\beta'_2\}) \Rightarrow \mathbf{ucs}_{\leq_1}(\beta_1 \to \beta_2,\ \beta'_1 \to \beta'_2)\} \\
\quad\quad\quad\quad \mathcal{E}_{1,10} & = & \{(\delta \;\mathsf{in}\; \{\beta''_2\}) \Rightarrow (\mathcal{E}_{1,15} \cup \mathcal{E}_{1,16} \cup \{\beta'_3 \to \beta''_1\})\} \\
\quad\quad\quad\quad\quad \mathcal{E}_{1,15} & = & \{(\delta \;\mathsf{in}\; \{\beta''_2\}) \Rightarrow \mathbf{ucs}_{\leq_1}(\beta_1 \to \beta_2,\ \beta''_1 \to \beta''_2)\} \\
\quad\quad\quad\quad\quad \mathcal{E}_{1,16} & = & \{(\delta \;\mathsf{in}\; \{\beta'_3\}) \Rightarrow \{\beta_3 \sqsubseteq \beta'_3\}\} \\
\quad\quad \mathcal{E}_{1,6} & = & \{(\delta \;\mathsf{in}\; \{\beta_4\}) \Rightarrow (\mathcal{E}_{1,13} \cup \mathcal{E}_{1,14} \cup \{\alpha_4 \sqsubseteq \beta_4\})\} \\
\quad\quad\quad \mathcal{E}_{1,13} & = & \emptyset \\
\quad\quad\quad \mathcal{E}_{1,14} & = & \{(\delta \;\mathsf{in}\; \{\alpha_4\}) \Rightarrow \{\alpha_4 \sqsubseteq \alpha'_4\}\} \\
\quad \mathcal{E}_{1,4} & = & \emptyset \\
\mathcal{E}_{1,2} & = & \{(\delta \;\mathsf{in}\; \{\alpha'_5\}) \Rightarrow \{\alpha_5 \sqsubseteq \alpha'_5\}\}
\end{array}$$

Figure 7.16: The set $\mathcal{E}_1$

### 7.6.3 An algorithm for inferring $\vdash_2$-typings

Let $\mathcal{W}_2$ be the algorithm obtained from the algorithm $\mathcal{W}_1$ in Fig. 7.13 by replacing (in the algorithm $\mathcal{W}$ in Figs. 7.14 and 7.15) all the occurrences of $\mathbf{ucs}_{\leq_1}$ and $\mathbf{cs}_{\leq_1}$ by $\mathbf{cs}_{=}$.

The inference algorithm $\mathcal{W}_2$ is correct and complete w.r.t. the $\vdash_2$ assignment system.

**Lemma 7.72** *$\vdash_{\mathbf{T}} M : \rho$ and $\mathcal{W}_2(M) = \langle \Theta, M'^{\theta}, \mathcal{E} \rangle$ imply*

1. *for all instantiations $\mathbf{i}$, if $\mathbf{i}$ is the maximum of $\mathbf{sat}(\mathcal{E})$, then $\mathbf{i}(\Theta) \vdash_2 \mathbf{i}(M'^{\theta})$,*
2. *for all $\Sigma$ and $M''^{\phi}$ such that $\epsilon(\Sigma) = \mathrm{FV}(\epsilon(M''^{\phi}))$ and $\epsilon(M''^{\phi}) = M$, if $\Sigma \vdash_2 M''^{\phi}$ then exists $\mathbf{i} \in \mathbf{sat}(\mathcal{E})$ such that $\mathbf{i}(\Theta) = \Sigma$ and $\mathbf{i}(M'^{\theta}) = M''^{\phi}$.* □

We are interested finding a faithful $\vdash_2$-typing showing the maximum amount of dead-code. This can be done as for $\vdash_1$ (see Theorem 7.70) by the following theorem.

**Theorem 7.73** *Let $\vdash_{\mathbf{T}} M : \rho$, and $\mathcal{W}_2(M) = \langle \Theta, M'^{\theta}, \mathcal{E} \rangle$. If $\mathbf{i}$ is the maximum of $\mathbf{sat}(\mathcal{E} \cup \mathbf{faithful}(\Theta, \theta))$ then $\mathbf{i}(\Theta) \vdash_2 \mathbf{i}(M'^{\theta})$ is a faithful $\vdash_2$-assignment showing the maximum amount of dead-code.* □

**Example 7.74** *Let*

$$N_1 = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.(fg + g\mathsf{d}_1^{\mathsf{nat}}) + (\lambda y^{\mathsf{nat}}.1)\mathsf{d}_2^{\mathsf{nat}})(\lambda z^{\mathsf{nat}}.3)\mathsf{d}_3^{\mathsf{nat}}$$

*be the term of Example 7.54.*

*We have $\mathcal{W}_2(N_1) = \langle \Theta_2, N_1''^{\gamma_6}, \mathcal{E}_2 \rangle$, where (writing, for short, $\zeta$ instead of $\zeta^{\mathsf{nat}}$): $\Theta_2 = \{f : (\alpha_1 \to \alpha_2) \to \alpha_3\}$, $Q''^{\gamma_3} =$*

$$\begin{array}{l}(((\lambda g^{\beta_1\to\beta_2}.(\lambda x^{\beta_3}.\\ \qquad (((f^{(\alpha_1'\to\alpha_2')\to\alpha_3'} g^{\beta_1'\to\beta_2'})^{\alpha_3'} +^{\gamma_1\times\gamma_2\to\gamma_3} (g^{\beta_1''\to\beta_2''}\mathsf{d}_1^{\beta_3'})^{\beta_2''})^{\gamma_3}\\ \qquad\qquad +^{\gamma_4\times\gamma_5\to\gamma_6}(\lambda y^{\beta_4}.1^{\gamma_7})\mathsf{d}_2^{\alpha_4'})^{\gamma_6})^{\beta_3\to\gamma_6})^{(\beta_1\to\beta_2)\to\beta_3\to\gamma_6}\\ \qquad\qquad (\lambda z^{\beta_5}.3^{\gamma_8})^{\beta_5\to\gamma_8})^{\beta_3\to\gamma_6}\mathsf{d}_3^{\alpha_5'})^{\gamma_6}\quad ,\end{array}$$

*and $\mathcal{E}_2$ is as in Fig. 7.17. We also have*

$$\mathbf{faithful}(\Theta_2, \gamma_6) \;=\; \{\;\; (\delta \;\mathsf{in}\; \{\alpha_3\}) \Rightarrow \{\alpha_1 \equiv \delta, \alpha_2 \equiv \delta\},\;\; \gamma_6 \equiv \delta \;\;\}\;\; .$$

*Applying the algorithm of Definition 7.67 we get $\mathcal{E}_2 \cup \mathbf{faithful}(\Theta_2, \gamma_6) \rightsquigarrow \mathcal{I}_1 \cup \mathcal{I}_2$, where $\mathcal{I}_1$ is as in Example 7.71 and*

$$\mathcal{I}_2 = \{\alpha_1', \beta_1, \beta_1', \beta_1'', \beta_3', \beta_5\}\;\; .$$

*Note that, if $\mathbf{i}_2$ is the instantiation represented by $\mathcal{I}_1 \cup \mathcal{I}_2$, then $\mathbf{i}_2(\Theta_2) \vdash_1 \mathbf{i}_2(N_1''^{\gamma_3})$ is the faithful $\vdash_2$-typing $\Sigma_2 \vdash_2 N_1'^{\delta}$ of Example 7.54.*

$$\mathcal{E}_2 \;=\; \{(\delta \;\mathsf{in}\; \{\gamma_6\}) \Rightarrow (\mathcal{E}_{2,1} \cup \mathcal{E}_{2,2} \cup \{\alpha'_5 \equiv \beta_3\})\}$$

where

$$\begin{array}{rcl}
\mathcal{E}_{2,1} & = & \{(\delta \;\mathsf{in}\; \{\gamma_6\}) \Rightarrow (\mathcal{E}_{2,3} \cup \mathcal{E}_{2,4} \cup \mathbf{cs}_{=}(\beta_5 \to \gamma_8,\ \beta_1 \to \beta_2))\} \\
\mathcal{E}_{2,3} & = & \{(\delta \;\mathsf{in}\; \{\gamma_6\}) \Rightarrow (\mathcal{E}_{2,5} \cup \mathcal{E}_{2,6} \cup \mathbf{cs}_{=}(\gamma_3 \times \gamma_7,\ \gamma_4 \times \gamma_5))\} \\
\mathcal{E}_{2,5} & = & \{(\delta \;\mathsf{in}\; \{\gamma_3\}) \Rightarrow (\mathcal{E}_{2,7} \cup \mathcal{E}_{2,8} \cup \mathbf{cs}_{=}(\alpha'_3 \times \beta''_2,\ \gamma_1 \times \gamma_2))\} \\
\mathcal{E}_{2,7} & = & \{(\delta \;\mathsf{in}\; \{\gamma_3\}) \Rightarrow \{\gamma_1 \equiv \delta,\ \gamma_2 \equiv \delta\}\} \\
\mathcal{E}_{2,8} & = & \mathcal{E}_{2,9} \cup \mathcal{E}_{2,10} \\
\mathcal{E}_{2,9} & = & \{(\delta \;\mathsf{in}\; \{\alpha'_3\}) \Rightarrow (\mathcal{E}_{2,11} \cup \mathcal{E}_{2,12} \cup \mathbf{cs}_{=}(\beta'_1 \to \beta'_2,\ \alpha'_1 \to \alpha'_2))\} \\
\mathcal{E}_{2,11} & = & \mathbf{cs}_{=}((\alpha_1 \to \alpha_2) \to \alpha_3,\ (\alpha'_1 \to \alpha'_2) \to \alpha'_3) \\
\mathcal{E}_{2,12} & = & \{(\delta \;\mathsf{in}\; \{\beta'_2\}) \Rightarrow \mathbf{cs}_{=}(\beta_1 \to \beta_2,\ \beta'_1 \to \beta'_2)\} \\
\mathcal{E}_{2,10} & = & \{(\delta \;\mathsf{in}\; \{\beta''_2\}) \Rightarrow (\mathcal{E}_{2,15} \cup \mathcal{E}_{2,16} \cup \{\beta'_3 \to \beta''_1\})\} \\
\mathcal{E}_{2,15} & = & \{(\delta \;\mathsf{in}\; \{\beta''_2\}) \Rightarrow \mathbf{cs}_{=}(\beta_1 \to \beta_2,\ \beta''_1 \to \beta''_2)\} \\
\mathcal{E}_{2,16} & = & \{(\delta \;\mathsf{in}\; \{\beta'_3\}) \Rightarrow \{\beta_3 \equiv \beta'_3\}\} \\
\mathcal{E}_{2,6} & = & \emptyset \\
\mathcal{E}_{2,13} & = & \emptyset \\
\mathcal{E}_{2,14} & = & \{(\delta \;\mathsf{in}\; \{\alpha_4\}) \Rightarrow \{\alpha_4 \equiv \alpha'_4\}\} \\
\mathcal{E}_{2,4} & = & \emptyset \\
\mathcal{E}_{2,2} & = & \emptyset
\end{array}$$

Figure 7.17: The set $\mathcal{E}_2$

## 7.7 An incremental dead-code detection an elimination procedure

In this section we show how the dead-code detection algorithms developed in Section 7.6 can be used to build an incremental dead-code detection and elimination procedure.

### 7.7.1 $\vdash_1$ and $\vdash_2$ inference in a single step

Let $\mathcal{W}^\bullet$ be the algorithm obtained from the algorithm $\mathcal{W}_1$ in Fig. 7.13 by replacing (in the algorithm $\mathcal{W}$ in Figs. 7.14 and 7.15 but *not* in the definition of the functions $\mathbf{cs}_{\leq_1}$ and $\mathbf{ucs}_{\leq_1}$ in Fig. 7.12) each occurrence of $\Rightarrow$ by $\Rightarrow^\bullet$.

The notation $\Rightarrow^\bullet$ is simply a way for keeping track (in the set of constraints generated by the inference algorithm) of the "guards" encapsulating the whole set of constraints associated to a particular subterm. The idea is that, whenever one of such guarded constraint is trivially satisfied (i.e., the guard is not opened) by an instantiation corresponding to a faithful $\vdash_1$-typing, then the corresponding subterm would be removed when the mapping $\mathbf{O}_1$ is applied to that faithful $\vdash_1$-decorated term.

The d-constraints that may contain occurrences of $\Rightarrow$ marked by $\bullet$ are called $\bullet$-d-constraints. For every set of $\bullet$-d-constraint $\mathcal{E}$ define $\mathcal{R}_\bullet(\mathcal{E})$ be the set of d-constraint obtained from $\mathcal{E}$ by

- removing all the marked constraints (i.e., by throwing away all the constraints of the form $(\delta \,\mathsf{in}\, \mathcal{G}) \Rightarrow^\bullet \mathcal{E}'$ for some $\mathcal{G}$ and $\mathcal{E}'$),
- removing all the guards (i.e., by repeatedly replacing a given set of d-constrains of the form $\mathcal{E}', (\delta \,\mathsf{in}\, \mathcal{G}) \Rightarrow \mathcal{E}''$ by $\mathcal{E}' \cup \mathcal{E}''$), and
- replacing every occurrence of $\sqsubseteq$ by $\equiv$.

Let $\mathbf{faithful}^\bullet(\Theta, \theta)$ be the the set of $\bullet$-d-constraints obtained from $\mathbf{faithful}(\Theta, \theta)$ (as defined in Corollary 7.70) by replacing each occurrence of $\Rightarrow$ by $\Rightarrow^\bullet$.

We now define an algorithm for solving the $\bullet$-d-constraints generated by the algorithm $\mathcal{W}^\bullet$. The algorithm is presented in natural semantics style using judgements $\mathcal{E} \rightsquigarrow_1 \mathcal{I}_1, \mathcal{I}_2$ (see Fig. 7.18). The underlying idea is that, for every PCFP$^\mathcal{T}$ term $M$ of type $\rho$, if $\mathcal{W}^\bullet(M) = \langle \Theta, M'^\theta, \mathcal{E} \rangle$ and $\mathcal{E} \cup \mathbf{faithful}^\bullet(\Theta, \theta) \rightsquigarrow_1 \mathcal{I}_1, \mathcal{I}_2$, then

1. the instantiation $\mathbf{i}_1$ (represented by $\mathcal{I}_1$) is such that $\mathbf{i}_1(\Theta) \vdash_1 \mathbf{i}_1(M'^\theta)$ is a faithful $\vdash_1$-typing showing the maximum amount of dead-code in $M$, and
2. the instantiation $\mathbf{i}_2$ (represented by $\mathcal{I}_1 \cup \mathcal{I}_2$) is such that $\mathbf{i}_2(\Theta) \vdash_2 \mathbf{i}_2(M''^\theta)$ is a faithful $\vdash_2$-typing showing the maximum amount of dead-code in $\epsilon(M''^\theta)$,

where $M''^\theta$ is the d-pattern decorated term obtained from $M'^\theta$ by replacing all maximal subterm decorated by d-$\top$-types in $\mathbf{i}_1(M'^\theta)$ with fresh dummy variables, i.e., such that $\epsilon(M''^\theta) = \mathbf{O}_1(\mathbf{i}_1(M'^\theta))$.

$$\text{(STOP}') \ \frac{\text{no other } \rightsquigarrow_2\text{-rule can be applied}}{\mathcal{E}\rightsquigarrow_2\emptyset} \qquad (\equiv') \ \frac{\{\zeta_1,\zeta_2\}=\{\alpha,\delta\} \qquad \mathcal{E}[\delta/\alpha]\rightsquigarrow\mathcal{I}}{\mathcal{E},\ \zeta_1\equiv\zeta_2\rightsquigarrow\mathcal{I}\cup\{\alpha\}}$$

$$\text{(CHANGE)} \ \frac{\text{no other } \rightsquigarrow_1\text{-rule can be applied} \qquad \mathcal{R}_\bullet(\mathcal{E})\rightsquigarrow_2\mathcal{I}}{\mathcal{E}\rightsquigarrow_1\emptyset,\mathcal{I}}$$

$$(\equiv) \ \frac{\mathcal{E}[\delta/\alpha]\rightsquigarrow_1\mathcal{I}_1,\mathcal{I}_2}{\mathcal{E},\ \zeta_1\equiv\zeta_2\rightsquigarrow_1\mathcal{I}_1\cup\{\alpha\},\mathcal{I}_2}\ \{\zeta_1,\zeta_2\}=\{\alpha,\delta\} \quad (\sqsubseteq) \ \frac{\mathcal{E}[\delta/\alpha]\rightsquigarrow_1\mathcal{I}_1,\mathcal{I}_2}{\mathcal{E},\ \alpha\sqsubseteq\delta\rightsquigarrow_1\mathcal{I}_1\cup\{\alpha\},\mathcal{I}_2}$$

$$(\Rightarrow) \ \frac{\mathcal{E}\cup\mathcal{E}''\rightsquigarrow_1\mathcal{I}_1,\mathcal{I}_2}{\mathcal{E},\ (\delta\ \mathsf{in}\ \mathcal{G})\Rightarrow\mathcal{E}''\rightsquigarrow_1\mathcal{I}_1,\mathcal{I}_2}\ \delta\in\mathcal{G}$$

$$(\Rightarrow^\bullet) \ \frac{\mathcal{E}\cup\mathcal{E}''\rightsquigarrow_1\mathcal{I}_1,\mathcal{I}_2}{\mathcal{E},\ (\delta\ \mathsf{in}\ \mathcal{G})\Rightarrow^\bullet\mathcal{E}''\rightsquigarrow_1\mathcal{I}_1,\mathcal{I}_2}\ \delta\in\mathcal{G}$$

Figure 7.18: "Natural semantics" rules for •-d-constraints solution (system $\rightsquigarrow_1$)

### 7.7.2 Dead-code elimination on pattern-decorated terms

Let $\vdash_{\mathbf{T}} M:\rho$, $\mathcal{W}^\bullet(M)=\langle\Theta,M'^\theta,\mathcal{E}\rangle$ and $\mathcal{E}\cup\mathbf{faithful}^\bullet(\Theta,\theta)\rightsquigarrow_1\mathcal{I}_1,\mathcal{I}_2$. Let $\mathbf{i}_1$ and $\mathbf{i}_2$ be the instantiations represented by $\mathcal{I}_1$ and $\mathcal{I}_1\cup\mathcal{I}_2$.

Define $\mathbf{O}_1^{\mathcal{I}_1}(M'^\theta)$ to be the term obtained from $M'^\theta$ following Definition 7.27. I.e., by

- replacing all the maximal subterms of $M'^\theta$ which are assigned a d-pattern $\eta$ such that $\mathbf{i}_1(\eta)\in\mathbf{L}_\top^{\mathrm{d}}$
- by a fresh dummy variable decorated by $\eta$.

Define $\mathbf{O}_1^{\mathcal{I}_1}(\Theta)=\{x:\eta\mid x:\eta\in\Sigma \text{ and } \mathbf{i}_1(\eta)\notin\mathbf{L}_\top^{\mathrm{d}}\}$. Then, by the results of Section 7.2 and 7.3, $\mathbf{i}_1(\mathbf{O}_1^{\mathcal{I}_1}(\Theta))\vdash_1\mathbf{i}_1(\mathbf{O}_1^{\mathcal{I}_1}(M'^\theta))$ and $\mathbf{i}_2(\mathbf{O}_1^{\mathcal{I}_1}(\Theta))\vdash_2\mathbf{i}_2(\mathbf{O}_1^{\mathcal{I}_1}(M'^\theta))$.

Similarly, define $\mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(M'^\theta)$ and $\mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(\Theta)$ to be the (pattern decorated) term and basis obtained from $\mathbf{O}_1^{\mathcal{I}_1}(\Theta)$ and $\mathbf{O}_1^{\mathcal{I}_1}(M'^\theta)$ by removing, following Definition 7.46, the subterms which are assigned a d-pattern $\theta'$ such that $\mathbf{i}_2(\theta')\in\mathbf{L}_\top^{\mathrm{d}}$. Then $\mathbf{i}_2(\mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(\Theta))\vdash_2\mathbf{i}_2(\mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(M'^\theta))$.

So $\epsilon(\mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(M'^\theta))$ is the best simplification of $M$ that can be obtained by a combined use of the mappings $\mathbf{O}_1$ and $\mathbf{O}_2$ (see Section 7.4). Notice that $\mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(\Theta)=\mathbf{O}_1^{\mathcal{I}_1}(\Theta)$.

**Example 7.75** *Let*

$$M=(\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.(fg+gx)+(\lambda y^{\mathsf{nat}}.1)P)(\lambda z^{\mathsf{nat}}.3)Q$$

*be the term of Example 7.54.*

Assume for simplicity (as in example 7.71) that $P=u^{\mathsf{nat}}$ and $Q=v^{\mathsf{nat}}$.

*We have* $\mathcal{W}^\bullet(M)=\langle\Theta_1,M''^{\gamma_6},\mathcal{E}\rangle$, *where* $\Theta_1$ *and* $M''^{\gamma_6}$ *are as in Example 7.71, while* $\mathcal{E}$ *is as in Fig. 7.19. We also have* $\mathbf{faithful}^\bullet(\Theta_1,\gamma_6)=$

$$\{\ (\delta\ \mathsf{in}\ \{\alpha_3\})\Rightarrow^\bullet\{\alpha_1\equiv\delta,\alpha_2\equiv\delta\},\ \ (\delta\ \mathsf{in}\ \{\alpha_4\})\Rightarrow^\bullet\{\alpha_4\equiv\delta\},\ \ \gamma_6\equiv\delta\ \}\ ,$$

*(compare the set above with* $\mathbf{faithful}(\Theta_1, \gamma_6)$ *in Example 7.71). Applying the algorithm of Fig. 7.18 we get* $\mathcal{E} \cup \mathbf{faithful}^\bullet(\Theta_1, \gamma_6) \rightsquigarrow_1 \mathcal{I}_1, \mathcal{I}_2$*, where* $\mathcal{I}_1$ *and* $\mathcal{I}_2$ *are as in Example 7.74.*

*Let* $\mathbf{i}_1$ *and* $\mathbf{i}_2$ *be the instantiation represented by* $\mathcal{I}_1$ *and* $\mathcal{I}_1 \cup \mathcal{I}_2$*, respectively. Then* $\mathbf{i}_1(\Theta_1) \vdash_1 \mathbf{i}_1(M''^{\gamma_3})$ *and* $\mathbf{i}_2(\mathbf{O}_1^{\mathcal{I}_1}(\Theta_1)) \vdash_2 \mathbf{i}_2(\mathbf{O}_1^{\mathcal{I}_1}(M''^{\gamma_3}))$ *are the faithful typings (*$\Sigma_1 \vdash_1 M'^{\delta}$ *and* $\Sigma_2 \vdash_2 N'^{\delta}$*, respectively) in Example 7.54.*

*We can use the mappings* $\mathbf{O}_1^{\mathcal{I}_1}$ *and* $\mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}$ *to remove the dead-code directly from the d-pattern decorated term* $P''^{\gamma_3}$ *(without instantiating the constructor variables). So we get* $\mathbf{O}_1^{\mathcal{I}_1}(\Theta) = \{f : (\alpha_1 \to \alpha_2) \to \alpha_3\}$,
$\mathbf{O}_1^{\mathcal{I}_1}(M''^{\gamma_3}) =$

$$
\begin{array}{l}
(((\lambda g^{\beta_1 \to \beta_2}.(\lambda x^{\beta_3}. \\
\qquad (((f^{(\alpha_1' \to \alpha_2') \to \alpha_3'} g^{\beta_1' \to \beta_2'})^{\alpha_3'} +^{\gamma_1 \times \gamma_2 \to \gamma_3} (g^{\beta_1'' \to \beta_2''} \mathsf{d}_1^{\beta_3'})^{\beta_2''})^{\gamma_3} \\
\qquad\qquad +^{\gamma_4 \times \gamma_5 \to \gamma_6} (\lambda y^{\beta_4}.1^{\gamma_7}) \mathsf{d}_2^{\alpha_4'})^{\gamma_6})^{\beta_3 \to \gamma_6})^{(\beta_1 \to \beta_2) \to \beta_3 \to \gamma_6} \\
\qquad\qquad\qquad (\lambda z^{\beta_5}.3^{\gamma_8})^{\beta_5 \to \gamma_8})^{\beta_3 \to \gamma_6} \mathsf{d}_3^{\alpha_5'})^{\gamma_6} \quad ,
\end{array}
$$

*and* $\mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(M''^{\gamma_3}) =$

$$
\begin{array}{l}
(((\lambda g^{\beta_1 \to \beta_2}. \\
\qquad (((f^{(\alpha_1' \to \alpha_2') \to \alpha_3'} g^{\beta_1' \to \beta_2'})^{\alpha_3'} +^{\gamma_1 \times \gamma_2 \to \gamma_3} (g^{\beta_1'' \to \beta_2''} \mathsf{d}_1^{\beta_3'})^{\beta_2''})^{\gamma_3} \\
\qquad\qquad +^{\gamma_4 \times \gamma_5 \to \gamma_6} 1^{\gamma_7})^{\gamma_6})^{(\beta_1 \to \beta_2) \to \gamma_6} \\
\qquad\qquad\qquad (\lambda z^{\beta_5}.3^{\gamma_8})^{\beta_5 \to \gamma_8})^{\beta_3 \to \gamma_6})^{\gamma_6} \quad ,
\end{array}
$$

### 7.7.3 Constraints simplification

When using the inference algorithm $\mathcal{W}^\bullet$ to analyze a term $M$ we will indeed produce as output a set of constraints that characterizes the $\vdash_1$-typings of $M$ and the $\vdash_2$-typings of a simplified (by the mapping $\mathbf{O}_1^{\vdash_1}$ of Section 7.4) version of $M$. The subterms of $M$ which are assigned $\top$-d-types in a faithful $\vdash_1$-typing are indeed certainly useless in any use of $M$, and can then be simplified. Moreover the constraints associated to them are not relevant to further analysis of $M$ and can then be removed from the set of constraints returned by the algorithm $\mathcal{W}^\bullet$. More precisely, if $\vdash_{\mathbf{T}} M : \rho$, $\mathcal{W}^\bullet(M) = \langle \Theta, M'^{\theta}, \mathcal{E} \rangle$ and $\mathcal{E} \cup \mathbf{faithful}^\bullet(\Theta, \theta) \rightsquigarrow_1 \mathcal{I}_1, \mathcal{I}_2$, then the set of constraints $\mathcal{E}$ can be replaced by any set of constraints $\mathcal{E}'$ such that,

(∗) for every $\mathcal{E}_0$, if $\mathcal{E} \cup \mathcal{E}_0 \rightsquigarrow_1 \mathcal{I}_1', \mathcal{I}_2'$, $\mathbf{i}_1 \sqsubseteq \mathbf{i}_1'$ and $\mathbf{i}_2 \sqsubseteq \mathbf{i}_2'$, then $\mathcal{E}' \cup \mathcal{E}_0 \rightsquigarrow_1 \mathcal{I}_1', \mathcal{I}_2'$, where $\mathbf{i}_1$, $\mathbf{i}_2$, $\mathbf{i}_1'$ and $\mathbf{i}_2'$ are the instantiations represented by $\mathcal{I}_1$, $\mathcal{I}_1 \cup \mathcal{I}_2$, $\mathcal{I}_1'$ and $\mathcal{I}_1' \cup \mathcal{I}_2'$, respectively.

We now introduce a constraints simplification algorithm that, given the finite set of constraints $\mathcal{E}$ and the sets $\mathcal{I}_1$ and $\mathcal{I}_2$, returns a simplified set of constraints $\mathcal{E}'$ such that the condition (∗) above holds.

The algorithm is presented in natural semantics style using judgements $\mathcal{E}; \mathcal{I}_1; \mathcal{I}_2 \gg \mathcal{E}'$ (see Fig. 7.20).

$$\mathcal{E} \;=\; \{(\delta \text{ in } \{\gamma_6\}) \Rightarrow^\bullet (\mathcal{E}_1 \cup \mathcal{E}_2 \cup \{\alpha'_5 \sqsubseteq \beta_3\})\}$$

where

$$\begin{array}{rcl}
\mathcal{E}_1 & = & \{(\delta \text{ in } \{\gamma_6\}) \Rightarrow^\bullet (\mathcal{E}_3 \cup \mathcal{E}_4 \cup \mathbf{cs}_{\leq_1}(\beta_5 \to \gamma_8,\ \beta_1 \to \beta_2))\} \\
\mathcal{E}_3 & = & \{(\delta \text{ in } \{\gamma_6\}) \Rightarrow^\bullet (\mathcal{E}_5 \cup \mathcal{E}_6 \cup \mathbf{cs}_{\leq_1}(\gamma_3 \times \gamma_7,\ \gamma_4 \times \gamma_5))\} \\
\mathcal{E}_5 & = & \{(\delta \text{ in } \{\gamma_3\}) \Rightarrow^\bullet (\mathcal{E}_7 \cup \mathcal{E}_8 \cup \mathbf{cs}_{\leq_1}(\alpha'_3 \times \beta''_2,\ \gamma_1 \times \gamma_2))\} \\
\mathcal{E}_7 & = & \{(\delta \text{ in } \{\gamma_3\}) \Rightarrow^\bullet \{\gamma_1 \equiv \delta,\ \gamma_2 \equiv \delta\}\} \\
\mathcal{E}_8 & = & \mathcal{E}_9 \cup \mathcal{E}_{10} \\
\mathcal{E}_9 & = & \{(\delta \text{ in } \{\alpha'_3\}) \Rightarrow^\bullet (\mathcal{E}_{11} \cup \mathcal{E}_{12} \cup \mathbf{cs}_{\leq_1}(\beta'_1 \to \beta'_2,\ \alpha'_1 \to \alpha'_2))\} \\
\mathcal{E}_{11} & = & \mathbf{ucs}_{\leq_1}((\alpha_1 \to \alpha_2) \to \alpha_3,\ (\alpha'_1 \to \alpha'_2) \to \alpha'_3) \\
\mathcal{E}_{12} & = & \{(\delta \text{ in } \{\beta'_2\}) \Rightarrow^\bullet \mathbf{ucs}_{\leq_1}(\beta_1 \to \beta_2,\ \beta'_1 \to \beta'_2)\} \\
\mathcal{E}_{10} & = & \{(\delta \text{ in } \{\beta''_2\}) \Rightarrow^\bullet (\mathcal{E}_{15} \cup \mathcal{E}_{16} \cup \{\beta'_3 \to \beta''_1\})\} \\
\mathcal{E}_{15} & = & \{(\delta \text{ in } \{\beta''_2\}) \Rightarrow^\bullet \mathbf{ucs}_{\leq_1}(\beta_1 \to \beta_2,\ \beta''_1 \to \beta''_2)\} \\
\mathcal{E}_{16} & = & \{(\delta \text{ in } \{\beta'_3\}) \Rightarrow^\bullet \{\beta_3 \sqsubseteq \beta'_3\}\} \\
\mathcal{E}_6 & = & \{(\delta \text{ in } \{\beta_4\}) \Rightarrow^\bullet (\mathcal{E}_{13} \cup \mathcal{E}_{14} \cup \{\alpha_4 \sqsubseteq \beta_4\})\} \\
\mathcal{E}_{13} & = & \emptyset \\
\mathcal{E}_{14} & = & \{(\delta \text{ in } \{\alpha_4\}) \Rightarrow^\bullet \{\alpha_4 \sqsubseteq \alpha'_4\}\} \\
\mathcal{E}_4 & = & \emptyset \\
\mathcal{E}_2 & = & \{(\delta \text{ in } \{\alpha'_5\}) \Rightarrow^\bullet \{\alpha_5 \sqsubseteq \alpha'_5\}\}
\end{array}$$

Figure 7.19: The set $\mathcal{E}$

$$(\text{AX})\ \frac{\text{no other rule can be applied}}{\mathcal{E};\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}}$$

$$(\equiv)\ \frac{\mathcal{E};\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}' \quad \zeta_1,\zeta_2 \notin \mathcal{I}_1 \cup \mathcal{I}_2 \cup \{\delta\}}{\mathcal{E},\ \alpha_1 \equiv \alpha_2;\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}'} \qquad (\sqsubseteq)\ \frac{\mathcal{E};\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}' \quad \zeta_1,\zeta_2 \notin \mathcal{I}_1 \cup \mathcal{I}_2 \cup \{\delta\}}{\mathcal{E},\ \zeta_1 \sqsubseteq \zeta_2;\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}'}$$

$$(\Rightarrow_1)\ \frac{\mathcal{E};\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}'}{\mathcal{E},\ (\delta \text{ in } \mathcal{G}) \Rightarrow \mathcal{E}_0;\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}'}\ \mathcal{G} \cap (\mathcal{I}_1 \cup \mathcal{I}_2 \cup \{\delta\}) = \emptyset$$

$$(\Rightarrow_2)\ \frac{\mathcal{E};\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}' \quad \mathcal{E}_0;\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}'_0}{\mathcal{E},\ (\delta \text{ in } \mathcal{G}) \Rightarrow \mathcal{E}_0;\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}' \cup \{(\delta \text{ in } \mathcal{G}) \Rightarrow \mathcal{E}'_0\}}\ \mathcal{G} \cap (\mathcal{I}_1 \cup \mathcal{I}_2 \cup \{\delta\}) \neq \emptyset$$

$$(\Rightarrow^\bullet_1)\ \frac{\mathcal{E};\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}'}{\mathcal{E},\ (\delta \text{ in } \mathcal{G}) \Rightarrow^\bullet \mathcal{E}_0;\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}'}\ \mathcal{G} \cap (\mathcal{I}_1 \cup \{\delta\}) = \emptyset$$

$$(\Rightarrow^\bullet_2)\ \frac{\mathcal{E};\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}' \quad \mathcal{E}_0;\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}'_0}{\mathcal{E},\ (\delta \text{ in } \mathcal{G}) \Rightarrow^\bullet \mathcal{E}_0;\mathcal{I}_1;\mathcal{I}_2 \gg \mathcal{E}' \cup \{(\delta \text{ in } \mathcal{G}) \Rightarrow^\bullet \mathcal{E}'_0\}}\ \mathcal{G} \cap (\mathcal{I}_1 \cup \{\delta\}) \neq \emptyset$$

Figure 7.20: "Natural semantics" rules for $\bullet$-d-constraints simplification (system $\gg$)

$$\mathcal{E}' \;=\; \{(\delta \text{ in } \{\gamma_6\}) \Rightarrow^\bullet (\mathcal{E}'_1 \cup \mathcal{E}'_2 \cup \{\alpha'_5 \sqsubseteq \beta_3\})\}$$

where

$$\begin{array}{rcl}
\mathcal{E}'_1 & = & \{(\delta \text{ in } \{\gamma_6\}) \Rightarrow^\bullet (\mathcal{E}'_3 \cup \mathcal{E}'_4 \cup \mathbf{cs}_{\leq_1}(\beta_5 \to \gamma_8,\ \beta_1 \to \beta_2))\} \\
\mathcal{E}'_3 & = & \{(\delta \text{ in } \{\gamma_6\}) \Rightarrow^\bullet (\mathcal{E}'_5 \cup \mathcal{E}'_6 \cup \mathbf{cs}_{\leq_1}(\gamma_3 \times \gamma_7,\ \gamma_4 \times \gamma_5))\} \\
\mathcal{E}'_5 & = & \{(\delta \text{ in } \{\gamma_3\}) \Rightarrow^\bullet (\mathcal{E}'_7 \cup \mathcal{E}'_8 \cup \mathbf{cs}_{\leq_1}(\alpha'_3 \times \beta''_2,\ \gamma_1 \times \gamma_2))\} \\
\mathcal{E}'_7 & = & \{(\delta \text{ in } \{\gamma_3\}) \Rightarrow^\bullet \{\gamma_1 \equiv \delta,\ \gamma_2 \equiv \delta\}\} \\
\mathcal{E}'_8 & = & \mathcal{E}'_9 \cup \mathcal{E}'_{10} \\
\mathcal{E}'_9 & = & \{(\delta \text{ in } \{\alpha'_3\}) \Rightarrow^\bullet (\mathcal{E}'_{11} \cup \mathcal{E}'_{12} \cup \mathbf{cs}_{\leq_1}(\beta'_1 \to \beta'_2,\ \alpha'_1 \to \alpha'_2))\} \\
\mathcal{E}'_{11} & = & \mathbf{ucs}_{\leq_1}((\alpha_1 \to \alpha_2) \to \alpha_3,\ (\alpha'_1 \to \alpha'_2) \to \alpha'_3) \\
\mathcal{E}'_{12} & = & \{(\delta \text{ in } \{\beta'_2\}) \Rightarrow^\bullet \mathbf{ucs}_{\leq_1}(\beta_1 \to \beta_2,\ \beta'_1 \to \beta'_2)\} \\
\mathcal{E}'_{10} & = & \{(\delta \text{ in } \{\beta''_2\}) \Rightarrow^\bullet (\mathcal{E}'_{15} \cup \mathcal{E}'_{16} \cup \{\beta'_3 \to \beta''_1\})\} \\
\mathcal{E}'_{15} & = & \{(\delta \text{ in } \{\beta''_2\}) \Rightarrow^\bullet \mathbf{ucs}_{\leq_1}(\beta_1 \to \beta_2,\ \beta''_1 \to \beta''_2)\} \\
\mathcal{E}'_{16} & = & \emptyset \\
\mathcal{E}'_6 & = & \{(\delta \text{ in } \{\beta_4\}) \Rightarrow^\bullet (\mathcal{E}'_{13} \cup \mathcal{E}'_{14} \cup \{\alpha_4 \sqsubseteq \beta_4\})\} \\
\mathcal{E}'_{13} & = & \emptyset \\
\mathcal{E}'_{14} & = & \emptyset \\
\mathcal{E}'_4 & = & \emptyset \\
\mathcal{E}'_2 & = & \emptyset
\end{array}$$

Figure 7.21: The set $\mathcal{E}'$

**Proposition 7.76** *Let $\mathcal{E}$ be a finite set of constraints and let $\mathbf{i}_1$, $\mathbf{i}_2$ be represented by $\mathcal{I}_1$ and $\mathcal{I}_1 \cup \mathcal{I}_2$. Then $\mathcal{E}; \mathcal{I}_1; \mathcal{I}_2 \gg \mathcal{E}'$ implies that the condition $(*)$ holds.* □

**Example 7.77** *Consider the output of the inference algorithm $\mathcal{W}^\bullet$ in Example 7.75, i.e., the triple $\langle \Theta_1, M''^{\gamma_3}, \mathcal{E} \rangle$, and the sets $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $\mathcal{E} \cup \mathbf{faithful}^\bullet(\Theta_1, \gamma_3) \rightsquigarrow_1 \mathcal{I}_1, \mathcal{I}_2$. The dead-code showed by the maximum faithful $\vdash_1$-typing can be immediately removed. Moreover $\mathcal{E}; \mathcal{I}_1; \mathcal{I}_2 \gg \mathcal{E}'$, where $\mathcal{E}'$ is as in Fig. 7.21. It is easy to see that the condition $(*)$ holds. This means that we can replace $\mathcal{E}$ by the simplified set of constraint $\mathcal{E}'$ without loss of information.*

*The triple $\langle \mathbf{O}_1^{\mathcal{I}_1}(\Theta_1), \mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(M''^{\gamma_3}), \mathcal{E}' \rangle$, where $\mathbf{O}_1^{\mathcal{I}_1}(\Theta_1)$ and $\mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(M''^{\gamma_3})$ are as in Example 7.75, is indeed all we need in the analysis of programs that use $M$.*

### 7.7.4 On pattern-decorated terms

Let $\mathcal{W}^\star$ be the algorithm obtained by modifying the algorithm $\mathcal{W}^\bullet$ (described at the beginning of Section 7.7.1) in such a way that, in the pattern-decorated term (second component of the triple $\langle \Theta, M'^{\theta}, \mathcal{E} \rangle$) returned by the algorithm, all the proper subterms are decorated just by the **tail** of the corresponding d-patterns.

Note that the decorated terms returned by $\mathcal{W}^\star$ give the same information (w.r.t. the possibility of removing dead-code by using the mappings $\mathbf{O}_2^{\mathcal{I}_1}$ and $\mathbf{O}_1^{\mathcal{I}_1\cup\mathcal{I}_2}$) of the ones returned by $\mathcal{W}^\bullet$, since, for every instantiation $\mathbf{i}$ and d-pattern $\theta$, $\mathbf{i}(\theta) \in \mathbf{L}^{\mathrm{d}}_{\top}$ if and only if $\mathbf{i}(\mathbf{tail}(\theta)) = \{\top\}$.

**Example 7.78** *Let*

$$M = (\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.(fg + gx) + (\lambda y^{\mathsf{nat}}.1)u)(\lambda z^{\mathsf{nat}}.3)v$$

*be the term of Example 7.54.*

Assume for simplicity (as in example 7.71) that $P = u^{\mathsf{nat}}$ and $Q = v^{\mathsf{nat}}$.

*We have* $\mathcal{W}^\star(M) = \langle \Theta_1, M^{\star\gamma_6}, \mathcal{E}\rangle$, *where (writing, for short,* $\zeta$ *instead of* $\zeta^{\mathsf{nat}}$*):* $\Theta_1 = \{f^{\,(\alpha_1\to\alpha_2)\to\alpha_3}, v^{\,\alpha_4}\}$, $M^{\star\gamma_6} =$

$$\begin{array}{l}(((\lambda g^{\{\beta_2\}}.(\lambda x^{\{\beta_3\}}.\\ \qquad (((f^{\{\alpha'_3\}}g^{\{\beta'_2\}})^{\{\alpha'_3\}} +^{\{\gamma_3\}} (g^{\{\beta''_2\}}x^{\{\beta'_3\}})^{\{\beta''_2\}})^{\{\gamma_3\}})^{\{\gamma_3\}})^{\{\gamma_3\}}\\ \qquad\qquad +^{\{\gamma_6\}}((\lambda y^{\{\beta_4\}}.1^{\{\gamma_7\}})^{\{\gamma_7\}}u^{\{\alpha'_4\}})^{\{\gamma_7\}})^{\{\gamma_6\}}\\ \qquad\qquad\qquad (\lambda z^{\{\beta_5\}}.3^{\{\gamma_8\}})^{\{\gamma_8\}})^{\{\gamma_6\}}v^{\{\alpha'_5\}})^{\gamma_6}\ ,\end{array}$$

*and* $\mathcal{E}$ *is as in Fig. 7.19.*
*Let* $\mathcal{I}_1$ *and* $\mathcal{I}_2$ *be as in Example 7.77, we have* $\mathbf{O}_1^{\mathcal{I}_1}(\Theta_1) = \{f^{\,(\alpha_1\to\alpha_2)\to\alpha_3}\}$, $\mathbf{O}_1^{\mathcal{I}_1}(M^{\star\gamma_6}) =$

$$\begin{array}{l}(((\lambda g^{\{\beta_2\}}.(\lambda x^{\{\beta_3\}}.\\ \qquad (((f^{\{\alpha'_3\}}g^{\{\beta'_2\}})^{\{\alpha'_3\}} +^{\{\gamma_3\}} (g^{\{\beta''_2\}}\mathsf{d}_1^{\{\beta'_3\}})^{\{\beta''_2\}})^{\{\gamma_3\}})^{\{\gamma_3\}})^{\{\gamma_3\}}\\ \qquad\qquad +^{\{\gamma_6\}}((\lambda y^{\{\beta_4\}}.1^{\{\gamma_7\}})^{\{\gamma_7\}}\mathsf{d}_2^{\{\alpha'_4\}})^{\{\gamma_7\}})^{\{\gamma_6\}}\\ \qquad\qquad\qquad (\lambda z^{\{\beta_5\}}.3^{\{\gamma_8\}})^{\{\gamma_8\}})^{\{\gamma_6\}}\mathsf{d}_3^{\{\alpha'_5\}})^{\gamma_6}\ ,\end{array}$$

*and* $\mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(M^{\star\gamma_6}) =$

$$\begin{array}{l}(((\lambda g^{\{\beta_2\}}.\\ \qquad (((f^{\{\alpha'_3\}}g^{\{\beta'_2\}})^{\{\alpha'_3\}} +^{\{\gamma_3\}} (g^{\{\beta''_2\}}\mathsf{d}_1^{\{\beta'_3\}})^{\{\beta''_2\}})^{\{\gamma_3\}})^{\{\gamma_3\}})^{\{\gamma_3\}}\\ \qquad\qquad +^{\{\gamma_6\}}1^{\{\gamma_7\}})^{\{\gamma_6\}}\\ \qquad\qquad\qquad (\lambda z^{\{\beta_5\}}.3^{\{\gamma_8\}})^{\{\gamma_8\}})^{\gamma_6}\ .\end{array}$$

### 7.7.5 The procedure

We have now all the components for the definition of an incremental dead-code detection and elimination procedure on PCFP$^{\mathcal{T}}$ terms. Let $\vdash_{\mathbf{T}} M : \rho$, $\mathcal{W}^\star(M) = \langle\Theta, M^{\star\theta}, \mathcal{E}\rangle$, $\mathcal{E} \cup \mathbf{faithful}(\Theta, \theta) \rightsquigarrow_1 \mathcal{I}_1, \mathcal{I}_2$, and $\mathcal{E}; \mathcal{I}_1; \mathcal{I}_2 \gg \mathcal{E}'$. We can define a simplification algorithm $\mathbf{O}$ such that

$$\mathbf{O}(\langle\Theta, M^{\star\theta}, \mathcal{E}\rangle, \mathcal{I}_1, \mathcal{I}_2) = \langle \mathbf{O}_1^{\mathcal{I}_1}(\Theta), \mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(M^{\star\theta}), \mathcal{E}'\rangle\ .$$

The triple $\langle \mathbf{O}_1^{\mathcal{I}_1}(\Theta), \mathbf{O}_2^{\mathcal{I}_1,\mathcal{I}_2}(M^{\star\theta}), \mathcal{E}'\rangle$ is all we need to perform further analysis of a program containing $M$, since the simplification performed by $\mathbf{O}$ are those which are possible in any context.

## 7.8 Dead-code analysis for Eager PCFP$^{\mathcal{T}}$

When evaluating a simplified program the dummy variables (introduced by the simplification mapping $\mathbf{O}_1$), can be handled as special values without operational meaning (to emphasize this, in the rest of this section, we refer to them as *dummy constants*). In fact

- the evaluation of a simplified term under the call-by-name evaluation strategy never requires the evaluation of a dummy constant, and
- the evaluation of a simplified term under any evaluation strategy never requires the evaluation of an application of a dummy constant (i.e., $\mathsf{d}M_1 \cdots M_n$ with $n \geq 1$) or of a projection of a dummy constant (i.e., $\mathsf{proj}_1\mathsf{d}$ or $\mathsf{proj}_2\mathsf{d}$).

This implies that the dead-code detection and elimination techniques introduced in this chapter (and also the ones that will be developed in Chapters 8 and 9) can be applied straightforwardly to eager[1] versions of PCFP$^{\mathcal{T}}$ (see Section 2.7). However, in this case, a simplified program may result not ground observationally equivalent, but only ground observationally greater, than the original one.

**Example 7.79** *Take the PCFP$^{\mathcal{T}}$ term*

$$M = (\lambda z^{\mathsf{nat}}.3)(\mathsf{fix}\, x^{\mathsf{nat}}.x).$$

*By applying the simplification mappings $\mathbf{O}_1^{\vdash_1}$ and $\mathbf{O}_2^{\vdash_2}$ of Section 7.4 we get*

1. $\mathbf{O}_1^{\vdash_1}(M) = (\lambda z^{\mathsf{nat}}.3)\mathsf{d}^{\mathsf{nat}}$, *and*
2. $\mathbf{O}_2^{\vdash_2}(M) = 3$.

*It is easy to see that, under the call-by-value evaluation strategy, we have that the original program $M$ is ground observationally less equal than (and not equivalent to) both $\mathbf{O}_1^{\vdash_1}(M)$ and $\mathbf{O}_2^{\vdash_2}(M)$.*

However, when considering terminating programs (which is the typical case for programs extracted from proof) we have that, independently from the evaluation strategy of the language (and so also for an eager language), the simplified programs are ground observationally equivalent to the original ones.

## 7.9 Relations with the technique of Berardi and Boerio

The main difference between our approach and that of [Berardi, 1996] and [Berardi and Boerio, 1995] is in the algorithm that finds the dead-code in a given term. The algorithm

[1] And, more in general, independently from the evaluation strategy of the language.

presented in these papers is a kind of backtracking algorithm. It finds the best simplification of a given term by trying to type simplified versions of the term without changing the original type and context. Such algorithm is rather difficult understand, and this makes its proof of correctness even more difficult to follow.

In our approach instead we assign to each term a system of constraints between constructor variables which can be seen as representing all the dead-code type decorations of the term. This system has always a maximal solution corresponding to the decoration of the term showing all the dead-code that can be proved by the corresponding dead-code assignment system. Detecting the best simplification of a term corresponds to looking for a solution that maximizes the number of $\top$'s constructors with the restriction that the whole term must be useful (the typing is faithful[2]). An important feature of our algorithm is that it is naturally compositional (see Section 7.7) while that of [Berardi, 1996] and [Berardi and Boerio, 1995] is not.

The language considered in [Berardi, 1996] (a simply typed $\lambda$-calculus, which can be seen as the language obtained from PCFP$^{\mathcal{T}}$ by removing the constructor $\mathsf{fix}$) is strongly normalizing. In [Berardi, 1996] page 676 some miscellaneous results concerning a simplification mapping, which is essentially the mapping $\mathbf{O}_2^{\vdash_2}$ (in Section 7.4 of this thesis), are presented. In particular two time and space complexity measures are introduced, and the gain introduced by the simplification mapping (w.r.t. these time and space measures) is examined for different evaluation strategies. These results can be summarized as follows.

1. The simplification mapping never increases space an time complexity for the call-by-name and the call-by-value evaluation strategies.

2. The highest advantages are when the evaluation strategy is close to call-by-value. Dually, the lowest advantages are when the evaluation strategy is closed to call-by-name.

3. In practice the simplification mapping decreases space and time consuming also when a call-by-need evaluation strategy is used.

## 7.10 Summary

The contents of this chapter can be summarized as follows.

[2]The condition of being a faithful $\vdash_1$-typing (Definition 7.28) is simply the translation in our framework of the condition introduced in [Berardi, 1993b; Berardi, 1996] to find dead-code. Namely, in the Berardi's type assignment system a subterm is dead-code if once removed (replaced by a dummy constant having a special type, corresponding to our $\top$-d-types) the global type of the term is unchanged. More precisely, in a faithful d-type assignment, the fact that the global d-type of the term is in $\mathbf{L}_\delta^{\mathrm{d}}$, reflects the Berardi's requirement that the global type of the term does not contain any occurrence of the special type denoting useless computations.

- The introduction of the two non-standard type assignment for dead-code detection $\vdash_1$ and $\vdash_2$ (with the associated dead-code removing mapping $\mathbf{O}_1$ and $\mathbf{O}_2$), which are sound w.r.t. the model $\mathcal{M}^{\mathrm{gnd}}$ of Section 2.4.1.

- The proof of completeness for the entailment relation for d-types $\leq_1$ w.r.t. the model $\mathcal{M}^{\mathrm{gnd}}$ of Section 2.4.1.

- The the presentation of an incremental dead-code detection and elimination procedure.

# Chapter 8

# Conjunctive dead-code analysis

In this chapter, we extend the d-type inference system $\vdash_1$ presented in Chapter 7 by allowing the use of conjunctions of d-types. In other words, given two different d-types $\phi_1$ and $\phi_2$ (such that $\epsilon(\phi_1) = \epsilon(\phi_2)$), we allow to assign both them (i.e., $\phi_1 \wedge \phi_2$) to the same function $f$. As a first example (for a more perspicacious example we refer to Section 8.3) of use for this extension consider the term:

$$N = (\lambda f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}\to\mathsf{nat}}.f(\lambda x^{\mathsf{nat}}.3)P + f(\lambda y^{\mathsf{nat}}.y)Q)\,(\lambda h^{\mathsf{nat}\to\mathsf{nat}}.\lambda z^{\mathsf{nat}}.hz) \ ,$$

where the application map $\lambda h^{\mathsf{nat}\to\mathsf{nat}}.\lambda z^{\mathsf{nat}}.hz$ is assigned to the formal parameter $f$. It is easy to see that the subterm $P$ is dead-code, i.e., in a call-by-name language, $N$ is observational equivalent to the simplified term

$$N' = (\lambda f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}\to\mathsf{nat}}.f(\lambda x^{\mathsf{nat}}.3)\mathsf{d} + f(\lambda y^{\mathsf{nat}}.y)Q)\,(\lambda h^{\mathsf{nat}\to\mathsf{nat}}.\lambda z^{\mathsf{nat}}.hz) \ ,$$

where the subterm $P$ has been replaced by the placeholder $\mathsf{d}$. To prove this by the d-type assignment system $\vdash_1$ of Section 7.2 we need to assign d-type $\phi_1 = (\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$ to the first occurrence of $f$ in the body of the $\lambda$-abstraction. On the other hand, since $Q$ is useful to the computation of the final value of $N$, we are forced to assign the d-type: $\phi_2 = (\delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}) \to \delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$ to the the second occurrence of $f$ in the body of the $\lambda$-abstraction. Both the d-types $\phi_1$ (function to function) and $\phi_2$ (constant function to constant function) can be assigned to $\lambda h^{\mathsf{nat}\to\mathsf{nat}}.\lambda z^{\mathsf{nat}}.hz$ by using the system $\vdash_1$, but they are not comparable using the d-type entailment relation $\leq_1$, moreover, in the language of properties $\mathbf{L}^{\mathrm{d}}$ considered in Chapter 7 there is no property $\phi$ of $\lambda h^{\mathsf{nat}\to\mathsf{nat}}.\lambda z^{\mathsf{nat}}.hz$ that implies both them. So with the system $\vdash_1$ it is not possible to prove that $P$ is dead-code, since for doing this it is necessary to assume such a property $\phi$ for the $\lambda$-abstracted variable $f$. As we will see (in Example 8.11), the system proposed in this chapter allows to assume the conjunction $\phi_1 \wedge \phi_2$ for $f$, and so allows to prove that $P$ is dead-code.

In general, the use of conjunction dead-code types allows to handle the situation in which (as in the example above) the argument of a function $\lambda f^{\rho}.M$ can be assigned different (non comparable) d-types types that are both needed to discover some dead-code in the body $M$ of the function.

$$(a)\ \frac{a \in \{\delta, \top\} \quad \iota \in \{\mathsf{nat}, \mathsf{bool}\}}{a^\iota \in \mathbf{L}^{\mathrm{cd}}(\iota)} \qquad (\to)\ \frac{\xi \in \mathbf{C}^{\mathrm{cd}}(\rho) \quad \psi \in \mathbf{L}^{\mathrm{cd}}(\sigma)}{\xi \to \psi \in \mathbf{L}^{\mathrm{cd}}(\rho \to \sigma)}$$

$$(\times)\ \frac{\phi_1 \in \mathbf{L}^{\mathrm{cd}}(\rho_1) \quad \phi_2 \in \mathbf{L}^{\mathrm{cd}}(\rho_2)}{\phi_1 \times \phi_2 \in \mathbf{L}^{\mathrm{cd}}(\rho_1 \times \rho_2)} \qquad (\wedge)\ \frac{\phi_1, \ldots, \phi_n \in \mathbf{L}^{\mathrm{cd}}(\rho)}{\phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{C}^{\mathrm{cd}}(\rho)}\ n \geq 1$$

Figure 8.1: Conjunctive dead-code types

The first section of this chapter introduces the language of conjunctive dead-code types and its semantics. Section 8.2 presents the conjunctive dead-code type assignment system. In Sect. 8.3 we propose a more perspicacious example of optimization of a program extracted from proof. Section 8.4 presents some other examples.

## 8.1 Conjunctive dead-code types

**Definition 8.1 (Conjunctive dead-code types)** *The language of* conjunctive dead-code types *(*cd-types *for short)* $\mathbf{L}^{\mathrm{cd}}$*, ranged over by* $\phi$ *and* $\psi$ *(with superscripts and subscripts if needed), and the set of* conjunctions of cd-types *(*cd-conjunctions *for short)* $\mathbf{C}^{\mathrm{cd}}$*, ranged over by* $\xi$*, are defined by:*

$$\mathbf{L}^{\mathrm{cd}} = \bigcup_{\rho \in \mathbf{T}} \mathbf{L}^{\mathrm{cd}}(\rho) \qquad\qquad \mathbf{C}^{\mathrm{cd}} = \bigcup_{\rho \in \mathbf{T}} \mathbf{C}^{\mathrm{cd}}(\rho)\ ,$$

*where* $\mathbf{L}^{\mathrm{cd}}(\rho)$ *and* $\mathbf{C}^{\mathrm{cd}}(\rho)$ *are defined by the rules in Fig. 5.1. Note that* $\mathbf{L}^{\mathrm{cd}} \subset \mathbf{C}^{\mathrm{cd}}$*.*

Since cd-types are properties of terms, in the following we use the words cd-type and property interchangeably.

The meaning of cd-types is formalized by interpreting each cd-type, $\phi$, as a *partial equivalence relation* (*p.e.r.* for short) over the interpretation of the type $\epsilon(\phi)$ in the model $\mathcal{M}^{\mathrm{gnd}}$ of Section 2.4.1, i.e., over the set of ground observational equivalence classes of closed terms of type $\epsilon(\phi)$.

**Definition 8.2 (Semantics of cd-types)** *1. The interpretation* $[\![\phi]\!]$ *of a cd-type* $\phi$ *is defined by:*

$$\begin{aligned}
[\![\delta^\iota]\!] &= \{\langle [N], [N] \rangle \mid [N] \in \mathbf{I}(\iota)\} \\
[\![\top^\iota]\!] &= \mathbf{I}(\iota) \times \mathbf{I}(\iota) \\
[\![\xi \to \psi]\!] &= \{\langle [M], [N] \rangle \mid \forall \langle [P], [Q] \rangle \in [\![\xi]\!] . \langle [MP], [NQ] \rangle \in [\![\psi]\!]\} \\
[\![\phi_1 \times \phi_2]\!] &= \{\langle [M], [N] \rangle \mid \forall i \in \{1, 2\}.\ \langle [\mathsf{proj}_i M], [\mathsf{proj}_i N] \rangle \in [\![\phi_i]\!]\},
\end{aligned}$$

*where the interpretation of an cd-conjunction* $\phi_1 \wedge \cdots \wedge \phi_n$ *is defined by:*

$$[\![\phi_1 \wedge \cdots \wedge \phi_n]\!] = \bigcap_{1 \leq i \leq n} [\![\phi_i]\!]\ .$$

2. *Let $\xi \in \mathbf{C}^{\mathrm{cd}}$, by $\sim_\xi$ we denote the relation between closed terms defined by: $M \sim_\xi N$ if and only if $\langle [M], [N] \rangle \in [\![\xi]\!]$.*

The subclasses of the *$\delta$-cd-types* and of the *$\top$-cd-types* collect all the cd-types characterizing the terms having values with and without a precise identity, respectively.

**Definition 8.3 ($\delta$-cd-types and $\top$-cd-types)** *1. The set $\mathbf{L}^{\mathrm{cd}}_\delta$ of $\delta$-cd-types is the subset of $\mathbf{L}^{\mathrm{cd}}$ containing only the basic property constructor $\delta$.*

2. *The set $\mathbf{L}^{\mathrm{cd}}_\top$ of $\top$-cd-types is inductively defined by:*

- $\top^\iota \in \mathbf{L}^{\mathrm{cd}}_\top$, *for* $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$
- $\xi \to \psi \in \mathbf{L}^{\mathrm{cd}}_\top$, *for for* $\xi \in \mathbf{C}^{\mathrm{cd}}$ *and* $\psi \in \mathbf{L}^{\mathrm{cd}}_\top$
- $\phi \times \psi \in \mathbf{L}^{\mathrm{cd}}_\top$, *for* $\phi, \psi \in \mathbf{L}^{\mathrm{cd}}_\top$.

The following proposition states that, if $\phi$ is an $\top$-cd-type, then $[\![\phi]\!]$ is the p.e.r. which relates all the elements of $\mathbf{I}(\epsilon(\phi))$, and, if $\phi$ is an $\delta$-cd-type, then $[\![\phi]\!]$ is the p.e.r. which relates each element of $\mathbf{I}(\epsilon(\phi))$ with itself.

**Proposition 8.4** *1. $\phi \in \mathbf{L}^{\mathrm{cd}}_\top$ if and only if $[\![\phi]\!] = \mathbf{I}(\epsilon(\phi)) \times \mathbf{I}(\epsilon(\phi))$, and*

2. *$\phi \in \mathbf{L}^{\mathrm{cd}}_\delta$ imply $[\![\phi]\!] = \{\langle [M], [M] \rangle \mid [M] \in \mathbf{I}(\epsilon(\phi))\}$.*

**Proof.** Point 1 follows by induction on cd-types from Definition 8.2. Point 2 derives directly from Proposition 7.4 by observing that, for every $\delta$-cd-type $\phi$, $[\![\phi]\!] = [\![\delta(\epsilon(\phi))]\!]$. □

## 8.2 Detecting and removing dead-code

### 8.2.1 An entailment relation for conjunctive dead-code types

In this section we introduce an entailment between cd-conjunctions, denoted $\leq_{\mathrm{cd}}$.

**Definition 8.5 (Entailment relation $\leq_{\mathrm{cd}}$)** *Let $\xi_1, \xi_2 \in \mathbf{C}^{\mathrm{cd}}$. We write $\xi_1 \leq_{\mathrm{cd}} \xi_2$ to mean that $\xi_1 \leq \xi_2$ is derivable by the rules in Fig. 8.2. By $\cong_{\mathrm{cd}}$ we denote the equivalence relation induced by $\leq_{\mathrm{cd}}$.*

It is immediate to show that $\leq_{\mathrm{cd}}$ is reflexive and transitive. Notice that, if $\xi_1$ and $\xi_2$ are conjunctions of $\top$-cd-types such that $\epsilon(\xi_1) = \epsilon(\xi_2)$, then $\xi_1 \cong_{\mathrm{cd}} \xi_2$.

The $\leq_{\mathrm{cd}}$ relation between cd-types is sound w.r.t. the p.e.r. interpretation.

**Theorem 8.6 (Soundness of $\leq_{\mathrm{cd}}$)** *$\xi_1 \leq_{\mathrm{cd}} \xi_2$ implies $[\![\xi_1]\!] \subseteq [\![\xi_2]\!]$.*

**Proof.** By induction on the derivation of the judgement $\xi_1 \leq_{\mathrm{cd}} \xi_2$. □

**Remark 8.7** *At the moment we do not known whether the $\leq_{\mathrm{cd}}$ entailment relation is complete w.r.t. $\mathcal{M}^{\mathrm{gnd}}$.*

$$\text{(Ref)}\ \phi \leq \phi \qquad (\top)\ \frac{\psi \in \mathbf{L}^{\mathrm{d}}_{\top} \quad \epsilon(\phi) = \epsilon(\psi)}{\phi \leq \psi}$$

$$(\to)\ \frac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_2 \to \psi_1 \leq \phi_1 \to \psi_2} \qquad (\times)\ \frac{\phi_1 \leq \phi_2 \quad \psi_1 \leq \psi_2}{\phi_1 \times \psi_1 \leq \phi_2 \times \psi_2}$$

$$(\wedge)\ \frac{\forall i \in \{1,\ldots,m\}.\exists j \in \{1,\ldots,n\}.\phi_j \leq \psi_i}{\phi_1 \wedge \cdots \wedge \phi_n \leq \psi_1 \wedge \cdots \wedge \psi_m}$$

Figure 8.2: Entailment rules for cd-types (system $\leq_{\mathrm{cd}}$)

### 8.2.2 An assignment system for detecting dead-code

If $x^\rho$ is a term variable of type $\rho$, an assumption for $x^\rho$ is an expression of the shape $x^\rho : \xi$, or $x : \xi$ for short, where $\xi \in \mathbf{C}^{\mathrm{cd}}$, and $\epsilon(\xi) = \rho$. A basis is a set $\Sigma$ of cd-conjunctions assumptions for term variables.

For each constant $\mathsf{c}$ a minimal cd-0-type $\mathbf{L}^{\mathrm{cd}}(\mathsf{c})$ for $\mathsf{c}$ is specified. For example, for all integers $\mathsf{n}$, $\mathbf{L}^{\mathrm{cd}}(\mathsf{n}) = \delta^{\mathsf{nat}}$ and $\mathbf{L}^{\mathrm{cd}}(+) = \delta^{\mathsf{nat}} \times \delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$. In the following we choose, for every PCFP$^{\mathcal{T}}$ constant $\mathsf{c}$ of type $\rho$ (see Remark 7.21.1), $\mathbf{L}^{\mathrm{cd}}(\mathsf{c}) = \delta(\rho)$.

**Definition 8.8 (Conjunctive dead-code type assignment system $\vdash_{\mathrm{cd}}$)** *We write $\Sigma \vdash_{\mathrm{cd}} M^\psi$ to mean that $\Sigma \vdash M^\psi$ can be derived by the rules in Fig. 8.3.*

**Remark 8.9** *1. It is worth mentioning that, in the rule ($\to$ E), the conditions*

- $\epsilon(\phi'_1 \wedge \cdots \wedge \phi'_n) = \epsilon(\phi_1 \wedge \cdots \wedge \phi_n)$, *and*
- $\psi \notin \mathbf{L}^{\mathrm{cd}}_{\top}$ implies $\forall i \in \{1,\ldots,n\}.\phi'_i \leq_{\mathrm{cd}} \phi_i$,

*are used instead of*

$$\forall i \in \{1,\ldots,n\}.\phi'_i \leq_{\mathrm{cd}} \phi_i \ .$$

*This is done to take into account the fact that: if $\phi_1 \wedge \cdots \wedge \phi_n \to \psi$ is an $\top$-cd-type then $\phi_1, \ldots, \phi_n$ can be any cd-types such that $\epsilon(\phi_1) = \cdots \epsilon(\phi_n) = \epsilon(\phi'_1) = \cdots = \epsilon(\phi'_n)$.*

*2. A $\{\!|\}$-sequence $\{\!|N^{\phi'_1}; \cdots; N^{\phi'_n}|\!\}$, is just a way of storing $n$ ($n \geq 1$) decorations of a subterm. These decorations correspond to different uses of the subterm. When $n = 1$ we will often write, for simplicity, $N^{\phi'_1}$ instead of $\{\!|N^{\phi'_1}|\!\}$.*

If $\Sigma \vdash_{\mathrm{cd}} M^\psi$ then $M^\psi$ has written in it the cd-types assigned to its subterms. We say that $M^\psi$ is a *decorated* term. Note that, being $\vdash_{\mathrm{cd}}$ an inference system, the same terms can have different decorations. Let $\Lambda^{\vdash_{\mathrm{cd}}}_{\mathbf{T}}$ be the set of $\vdash_{\mathrm{cd}}$-decorated PCFP$^{\mathcal{T}}$ terms, i.e.,

$$\Lambda^{\vdash_{\mathrm{cd}}}_{\mathbf{T}} = \{M^\phi \mid \Sigma \vdash_{\mathrm{cd}} M^\phi \text{ for some cd-type } \phi \text{ and basis } \Sigma\}.$$

The functions $\epsilon(\cdot)$, $\delta(\cdot)$, and $\top(\cdot)$ are naturally extended to decorated terms. In particular $\epsilon(M^\phi)$ is simply the term obtained from $M^\phi$ by erasing all the d-type decorations, while, for

$$\text{(Var)}\ \frac{\phi_i \leq_{\mathrm{cd}} \phi_i'}{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash x^{\phi_i'}}\ 1 \leq i \leq n \qquad \text{(Con)}\ \frac{\mathbf{L}^{\mathrm{cd}}(\mathsf{c}) \leq_{\mathrm{cd}} \phi}{\Sigma \vdash \mathsf{c}^{\phi}}$$

$$(\to \mathrm{I})\ \frac{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash M^{\psi}}{\Sigma \vdash (\lambda x^{\phi_1 \wedge \cdots \wedge \phi_n}.M^{\psi})^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi}}$$

$$(\to \mathrm{E})\ \frac{\begin{array}{c}\Sigma \vdash M^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi} \qquad \Sigma \vdash N^{\phi_1'}\ \cdots\ \Sigma \vdash N^{\phi_n'} \\ \epsilon(\phi_1' \wedge \cdots \wedge \phi_n') = \epsilon(\phi_1 \wedge \cdots \wedge \phi_n) \\ \psi \notin \mathbf{L}^{\mathrm{cd}}_{\top} \text{ implies } \forall i \in \{1, \ldots, n\}.\phi_i' \leq_{\mathrm{cd}} \phi_i\end{array}}{\Sigma \vdash (M^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi} \{\!| N^{\phi_1'}; \cdots; N^{\phi_n'} |\!\})^{\psi}}$$

$$(\times \mathrm{I})\ \frac{\Sigma \vdash {M_1}^{\psi_1} \qquad \Sigma \vdash {M_2}^{\psi_2}}{\Sigma \vdash \langle {M_1}^{\psi_1}, {M_2}^{\psi_2} \rangle^{\psi_1 \times \psi_2}} \qquad (\times \mathrm{E}_i)\ \frac{\Sigma \vdash M^{\psi_1 \times \psi_2}}{\Sigma \vdash (\mathsf{proj}_i M^{\psi_1 \times \psi_2})^{\psi_i}}\ i \in \{1, 2\}$$

$$\text{(Fix)}\ \frac{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash M^{\phi_1} \ \cdots\ \Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash M^{\phi_n}}{\Sigma \vdash (\mathsf{fix}\, x^{\phi_i}.\{\!| M^{\phi_1}; \cdots; M^{\phi_n} |\!\})^{\phi_i}}\ 1 \leq i \leq n$$

$$\text{(If)}\ \frac{\begin{array}{c}\Sigma \vdash N^{{a_0}^{\mathsf{bool}}} \qquad \Sigma \vdash {M_1}^{\psi_1} \qquad \Sigma \vdash {M_2}^{\psi_2} \\ \psi_1 \leq_{\mathrm{cd}} \psi \qquad \psi_2 \leq_{\mathrm{cd}} \psi \\ a_0 = \top \text{ implies } \psi \in \mathbf{L}^{\mathrm{cd}}_{\top}\end{array}}{\Sigma \vdash (\mathsf{if}\, N^{{a_0}^{\mathsf{bool}}}\, \mathsf{then}\, {M_1}^{\psi_1}\, \mathsf{else}\, {M_2}^{\psi_2})^{\psi}}$$

$$\text{(Case)}\ \frac{\begin{array}{c}\Sigma \vdash N^{{a_0}^{\mathsf{nat}}} \qquad \Sigma \vdash M^{\psi_1} \qquad \Sigma \vdash F^{a^{\mathsf{nat}} \to \psi_2} \\ \psi_1 \leq_{\mathrm{cd}} \psi \qquad \psi_2 \leq_{\mathrm{cd}} \psi \\ a_0 = \top \text{ implies } \psi \in \mathbf{L}^{\mathrm{cd}}_{\top}\end{array}}{\Sigma \vdash \mathsf{case}(N^{{a_0}^{\mathsf{nat}}}, M^{\psi_1}, F^{a^{\mathsf{nat}} \to \psi_2})^{\psi}}$$

$$\text{(It)}\ \frac{\begin{array}{c}\Sigma \vdash N^{{a_0}^{\mathsf{nat}}} \qquad \forall j \in \{1, \ldots, n\}.\Sigma \vdash M^{\phi_j} \\ \forall j \in \{1, \ldots, n\}.\Sigma \vdash F^{\phi_1 \wedge \cdots \wedge \phi_n \to \phi_j} \\ a_0 = \top \text{ implies } \forall j \in \{1, \ldots, n\}.\phi_j \in \mathbf{L}^{\mathrm{cd}}_{\top}\end{array}}{\begin{array}{l}\Sigma \vdash \mathsf{it}(N^{{a_0}^{\mathsf{nat}}}, \{\!| M^{\phi_1}; \cdots; M^{\phi_n} |\!\}, \\ \qquad \{\!| F^{\phi_1 \wedge \cdots \wedge \phi_n \to \phi_1}; \cdots; F^{\phi_1 \wedge \cdots \wedge \phi_n \to \phi_1} |\!\})^{\phi_i}\end{array}}\ 1 \leq i \leq n$$

$$\text{(Rec)}\ \frac{\begin{array}{c}\Sigma \vdash N^{{a_0}^{\mathsf{nat}}} \qquad \forall j \in \{1, \ldots, n\}.\Sigma \vdash M^{\phi_j} \\ \forall j \in \{1, \ldots, n\}.\Sigma \vdash F^{a^{\mathsf{nat}} \to \phi_1 \wedge \cdots \wedge \phi_n \to \phi_j} \\ a_0 = \top \text{ implies } \forall j \in \{1, \ldots, n\}.\phi_j \in \mathbf{L}^{\mathrm{cd}}_{\top}\end{array}}{\begin{array}{l}\Sigma \vdash \mathsf{rec}(N^{{a_0}^{\mathsf{nat}}}, \{\!| M^{\phi_1}; \cdots; M^{\phi_n} |\!\}, \\ \qquad \{\!| F^{a^{\mathsf{nat}} \to \phi_1 \wedge \cdots \wedge \phi_n \to \phi_1}; \cdots; F^{a^{\mathsf{nat}} \to \phi_1 \wedge \cdots \wedge \phi_n \to \phi_n} |\!\})^{\phi_i}\end{array}}\ 1 \leq i \leq n$$

Figure 8.3: Rules for cd-type assignment (system $\vdash_{\mathrm{cd}}$)

each $a \in \{\delta, \top\}$ and $M \in \Lambda_{\mathbf{T}}$, $a(M)$ is the decorated term obtained from $M$ by decorating each subterm with the corresponding d-$a$-type. The proof of the following fact (where $\epsilon : \Lambda_{\mathbf{T}}^{\vdash_{\mathrm{cd}}} \to \Lambda_{\mathbf{T}}$, $\delta : \Lambda_{\mathbf{T}} \to \Lambda_{\mathbf{T}}^{\vdash_{\mathrm{cd}}}$, and $\top : \Lambda_{\mathbf{T}} \to \Lambda_{\mathbf{T}}^{\vdash_{\mathrm{cd}}}$ are defined as in Notation 7.22) is immediate.

**Fact 8.10** *1. $\Sigma \vdash_{\mathrm{cd}} M^{\psi}$ implies $\vdash_{\mathbf{T}} \epsilon(M^{\psi}) : \epsilon(\psi)$ and $\epsilon(\Sigma) \supseteq \mathrm{FV}(M)$.*

*2. $\vdash_{\mathbf{T}} M : \rho$ implies*

*(a) $\delta(\mathrm{FV}(M)) \vdash_{\mathrm{cd}} \delta(M)$, and*

*(b) for all basis $\Sigma$ such that $\Sigma = \{x^{\epsilon(\xi)} : \xi \mid x^{\epsilon(\xi)} \in \mathrm{FV}(M)$ for some $\xi \in \mathbf{C}^{\mathrm{cd}}\}$, $\Sigma \vdash_{\mathrm{cd}} \top(M)$.* □

**Example 8.11** *Let $M$ be the term of type* nat *considered at the begin of the chapter, i.e., $M =$*

$$(\lambda f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}\to\mathsf{nat}}. \\ f(\lambda x^{\mathsf{nat}}.3)P + f(\lambda y^{\mathsf{nat}}.y)Q) \\ (\lambda h^{\mathsf{nat}\to\mathsf{nat}}.\lambda z^{\mathsf{nat}}.hz)\ .$$

*where $\mathrm{FV}(P) = \{u^{\mathsf{nat}}\}$, $\mathrm{FV}(Q) = \{v^{\mathsf{nat}}\}$. We have $\mathrm{FV}(M) = \{u^{\mathsf{nat}}, v^{\mathsf{nat}}\}$.*

*Let $\phi_1 = (\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$ and $\phi_2 = (\delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}) \to \delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$. It is easy to check that $\Sigma \vdash_{\mathrm{cd}} {M'}^{\delta^{\mathsf{nat}}}$, where (writing, for short, $\delta$ and $\top$ instead of $\delta^{\mathsf{nat}}$ and $\top^{\mathsf{nat}}$): $\Sigma = \{u : \top, v : \delta\}$ and $M'^{\delta} =$*

$$((\lambda f^{\phi_1\wedge\phi_2}. \\ (((f^{\phi_1}(\lambda x^{\top}.3^{\delta})^{\top\to\delta})^{\top\to\delta}u^{\top})^{\delta} +^{\delta\times\delta\to\delta} ((f^{\phi_2}(\lambda y^{\top}.y^{\delta})^{\delta\to\delta})^{\delta\to\delta}v^{\delta})^{\delta})^{\delta\times\delta})^{\delta})^{\phi_1\wedge\phi_2\to\delta} \\ \{\!|(\lambda h^{\top\to\delta}.(\lambda z^{\top}.(h^{\top\to\delta}z^{\top})^{\delta})^{\top\to\delta})^{\phi_1}; (\lambda h^{\delta\to\delta}.(\lambda z^{\delta}.(h^{\delta\to\delta}z^{\delta})^{\delta})^{\delta\to\delta})^{\phi_1}|\!\})^{\delta}\ .$$

To state the soundness of the cd-type assignment system w.r.t. the semantics we introduce the following definition.

**Definition 8.12** *1. Two environments $e_1$, $e_2$ are $\Sigma$-related if and only if, for all $x : \phi_1 \wedge \cdots \wedge \phi_n \in \Sigma$, $\langle e_1(x), e_2(x)\rangle \in [\![\phi_1 \wedge \cdots \wedge \phi_n]\!]$.*

*2. Let $\Sigma \vdash_{\mathrm{cd}} M^{\psi}$ and $\Sigma \vdash_{\mathrm{cd}} N^{\psi}$. We write $\epsilon(M^{\psi}) \sim_{\psi}^{\Sigma} \epsilon(N^{\psi})$ to mean that for all $e_1$, $e_2$, if $e_1$ and $e_2$ are $\Sigma$-related, then $\langle [\![\epsilon(M^{\psi})]\!]_{e_1}, [\![\epsilon(N^{\psi})]\!]_{e_2}\rangle \in [\![\psi]\!]$.*

Now we can state the soundness of the system $\vdash_{\mathrm{cd}}$.

**Theorem 8.13 (Soundness of $\vdash_{\mathrm{cd}}$)** *Let $\Sigma \vdash_{\mathrm{cd}} M^{\psi}$. Then $\epsilon(M^{\psi}) \sim_{\psi}^{\Sigma} \epsilon(M^{\psi})$.*

**Proof.** By induction on the derivation of $\Sigma \vdash_{\mathrm{cd}} M^{\phi}$, using Theorem 8.6 (soundness of $\leq_{\mathrm{cd}}$) for the rules involving entailment. □

### 8.2.3 Dead-code elimination

In this section we introduce a simplification mapping $\mathbf{O}^{\wedge}$ that, given a decorated term $M^{\psi}$, returns an optimized version of $\epsilon(M^{\psi})$.

To define the simplification mapping we introduce a notion of pruning[1] on the set of $\mathrm{PCFP}^{\mathcal{T}}$ terms.

#### Dummy variables and pruning relation

As in Section 7.2.3, for each type $\rho$, define $\mathsf{d}^{\rho}$ be a *dummy variable* of type $\rho$. Dummy variables are not present in the original programs, but they are introduced by the dead-code elimination algorithm of Definition 8.19, that replaces all the maximal subterms that are proved to be dead-code by (free) dummy variables of the same type.

Dummy variables are simply placeholders for some dead-code removed, therefore each occurrence of a dummy variable in a program is dead-code. So they can be handled in a special by the dead-code type assignment system $\vdash_{\mathrm{cd}}$. In particular, given a term $M$ obtained by a correct application of the mapping $\mathbf{O}_1$ (see Theorem 8.24), it is safe to allow that any cd-type $\phi \in \mathbf{L}^{\mathrm{cd}}(\rho)$ can be assigned to a dummy variable $\mathsf{d}^{\rho}$ occurring in $M$. This can be ensured by:

- forbidding dummy variables to occur in bases, and
- adding to the rules in Fig. 8.3 the following rule

$$(\mathsf{d})\ \frac{\vdash_{\mathbf{T}} \mathsf{d}^{\rho} : \rho \qquad \phi \in \mathbf{L}^{\mathrm{cd}}(\rho)}{\vdash \mathsf{d}^{\phi}} \quad .$$

For every term $M$, define $\mathrm{FV}'(M)$ be the set of the free non dummy variables in $M$.

We can now introduce the pruning relation on terms.

**Definition 8.14 (Pruning relation)** *Let $\vdash_{\mathbf{T}} M : \rho$. We say that a term $N$ is a* pruning *of $M$, and write $N \preceq_{\mathrm{prune}} M$, if $N$ can be obtained from $M$ by replacing some subterms by fresh dummy variables of the corresponding type.*

**Fact 8.15** *Let $\vdash_{\mathbf{T}} M : \rho$ and $N \preceq_{\mathrm{prune}} M$. Then $\vdash_{\mathbf{T}} N : \rho$ and $\mathrm{FV}'(N) \subseteq \mathrm{FV}'(M)$.* □

**Example 8.16** *Consider the terms $M = (\lambda x^{\mathsf{nat}}.3)P$ and $M' = (\lambda x^{\mathsf{nat}}.3)\mathsf{d}^{\mathsf{nat}}$. It is easy to check that $M' \preceq_{\mathrm{prune}} M$.*

**Definition 8.17 (Operation sup)** *1. Let $\vdash_{\mathbf{T}} M : \rho$, $M_1 \preceq_{\mathrm{prune}} M$, and $M_2 \preceq_{\mathrm{prune}} M$. Then $\mathbf{sup}(M_1, M_2)$ is the term defined by the clauses in Fig. 8.4 (note that this definition is exhaustive, i.e., all the possible cases are considered).*

[1] This notion of pruning is essentially the one introduced in [Berardi, 1993a; Berardi, 1996], where *dummy constants* (having special types) are used instead of *dummy variables*.

$\mathbf{sup}(M, \mathsf{d}^{\rho}) = \mathbf{sup}(\mathsf{d}^{\rho}, M) = M$, for every term $M$ of type $\rho$

$\mathbf{sup}(\mathsf{c}^{\rho}, \mathsf{c}^{\rho}) = \mathsf{c}^{\rho}$

$\mathbf{sup}(x^{\rho}, x^{\rho}) = x^{\rho}$

$\mathbf{sup}(\langle M_1, M_2\rangle, \langle N_1, N_2\rangle) = \langle \mathbf{sup}(M_1, N_1), \mathbf{sup}(M_2, N_2)\rangle$

$\mathbf{sup}(\mathsf{proj}_i M, \mathsf{proj}_i N) = \mathsf{proj}_i \mathbf{sup}(M, N), \quad$ where $i \in \{1, 2\}$

$\mathbf{sup}(M_1 M_2, N_1 N_2) = \mathbf{sup}(M_1, N_1)\mathbf{sup}(M_2, N_2)$

$\mathbf{sup}(\lambda x^{\rho}.M, \lambda x^{\rho}.N) = \lambda x^{\rho}.\mathbf{sup}(M, N)$

$\mathbf{sup}(\mathsf{fix}\, x^{\rho}.M, \mathsf{fix}\, x^{\rho}.N) = \mathsf{fix}\, x^{\rho}.\mathbf{sup}(M, N)$

$\mathbf{sup}(\mathsf{if}\ M\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2,\ \mathsf{if}\ N\ \mathsf{then}\ N_1\ \mathsf{else}\ N_2) =$
$\quad \mathsf{if}\ \mathbf{sup}(M, N)\ \mathsf{then}\ \mathbf{sup}(M_1, N_1)\ \mathsf{else}\ \mathbf{sup}(M_2, N_2)$

$\mathbf{sup}(\mathsf{case}(M, P, F), \mathsf{case}(N, Q, G)) = \mathsf{case}(\mathbf{sup}(M, N), \mathbf{sup}(P, Q), \mathbf{sup}(F, G))$

$\mathbf{sup}(\mathsf{it}(M, P, F), \mathsf{it}(N, Q, G)) = \mathsf{it}(\mathbf{sup}(M, N), \mathbf{sup}(P, Q), \mathbf{sup}(F, G))$

$\mathbf{sup}(\mathsf{rec}(M, P, F), \mathsf{rec}(N, Q, G)) = \mathsf{rec}(\mathbf{sup}(M, N), \mathbf{sup}(P, Q), \mathbf{sup}(F, G))$

Figure 8.4: Operation **sup**

2. *Let* $\vdash_{\mathbf{T}} M : \rho$, $M_1 \preceq_{\text{prune}} M$, $\ldots$, $M_n \preceq_{\text{prune}} M$, $(n \geq 1)$ *Define* $\mathbf{sup}(M_1)$ *be* $M_1$ *and, for* $n \geq 2$, *define* $\mathbf{sup}(M_1, \ldots, M_n)$ *be*

$$\mathbf{sup}(\cdots \mathbf{sup}(\mathbf{sup}(M_1, M_2), M_3) \cdots, M_n) \ .$$

**Theorem 8.18** *Let* $\vdash_{\mathbf{T}} M : \rho$. *The set*

$$\{M' \mid M' \preceq_{\text{prune}} M\} \ ,$$

*with the order relation* $\preceq_{\text{prune}}$ *is a finite lattice with bottom* $\mathsf{d}^{\rho}$ *and top* $M$. *The operation* **sup** *of Definition 8.17 is the join of the lattice.* □

### The simplification mapping

Let $\Lambda_{\mathbf{T}}^{\vdash_{\mathrm{cd}}}$ be the set of all decorated terms which are defined according to Definition 8.8, i.e., $\Lambda_{\mathbf{T}}^{\vdash_{\mathrm{cd}}} = \{M^{\psi} \mid \Sigma \vdash_{\mathrm{cd}} M^{\psi} \text{ for some cd-type } \psi \text{ and basis } \Sigma\}$.

**Definition 8.19 (Simplification mapping $\mathbf{O}^{\wedge}$ on terms)** *1. The function*

$$\mathbf{O}^{\wedge} : \Lambda_{\mathbf{T}}^{\vdash_{\mathrm{cd}}} \to \Lambda_{\mathbf{T}}$$

*is defined by the clauses in Fig. 8.5.*

*2. If $\Sigma$ is a basis then*

$$\mathbf{O}^{\wedge}(\Sigma) = \{x^{\epsilon(\phi_1)} \mid x : \phi_1 \wedge \cdots \wedge \phi_n \in \Sigma,\ n \geq 1 \text{ and } \exists i \in \{1, \ldots, n\}.\phi_i \notin \mathbf{L}_{\top}^{\mathrm{cd}}\}.$$

The simplification mapping produces well typed terms, as stated by the following fact.

**Fact 8.20** *If $\Sigma \vdash_{\mathrm{cd}} M^{\psi}$ then*

1. $\mathbf{O}^{\wedge}(M^{\psi}) \preceq_{\mathrm{prune}} \epsilon(M^{\psi})$,
2. $\vdash_{\mathbf{T}} \mathbf{O}^{\wedge}(M^{\psi}) : \epsilon(\psi)$, *and*
3. $\mathbf{O}^{\wedge}(\Sigma) \supseteq \mathrm{FV}'(\mathbf{O}^{\wedge}(M^{\psi}))$. □

To prove the correctness of the simplification mapping $\mathbf{O}^{\wedge}$ we identify a subset of $\vdash_{\mathrm{cd}}$-typings for which the $\sim_{\psi}^{\Sigma}$ relation implies the $\simeq_{\mathrm{obs}}$ relation.

**Definition 8.21 (Faithful cd-type assignment)** $\Sigma \vdash_{\mathrm{cd}} M^{\phi}$ *is a* faithful *cd-type assignment statement if $\phi \in \mathbf{L}_{\delta}^{\mathrm{d}}$, and for all $x : \phi_1 \wedge \cdots \wedge \phi_n \in \Sigma$, $n = 1$ and $\phi_1 \in \mathbf{L}_{\top}^{\mathrm{d}} \cup \mathbf{L}_{\delta}^{\mathrm{d}}$.*

The correctness proof of the simplification mappings $\mathbf{O}^{\wedge}$ rely on the following theorem.

**Theorem 8.22** *Let $\Sigma \vdash_{\mathrm{cd}} M^{\phi}$ and $\Sigma \vdash_{\mathrm{cd}} N^{\phi}$ be faithful cd-typings. Then $\epsilon(M^{\phi}) \sim_{\phi}^{\Sigma} \epsilon(N^{\phi})$ implies $\epsilon(M^{\phi}) \simeq_{\mathrm{obs}} \epsilon(N^{\phi})$.*

**Proof.** Similar to the one of Theorem 7.30. □

The following result can be proved using the cd-type semantics.

**Theorem 8.23** *If $\Sigma \vdash_{\mathrm{cd}} M^{\psi}$ and $N$ is such that $\mathbf{O}^{\wedge}(M^{\psi}) \preceq_{\mathrm{prune}} N \preceq_{\mathrm{prune}} \epsilon(M^{\psi})$, then $\epsilon(M^{\psi}) \sim_{\psi}^{\Sigma} N$.* □

This result is especially interesting when the typing of $M$ is faithful since, from the above theorem and Theorem 8.22, we get that if $\Sigma \vdash_{\mathrm{cd}} M^{\psi}$ is a faithful $\vdash_{\mathrm{cd}}$-typing statement then $\epsilon(M^{\psi})$ and $\mathbf{O}^{\wedge}(M^{\psi})$ are observationally equivalent.

**Theorem 8.24** *Let $\Sigma \vdash_{\mathrm{cd}} M^{\psi}$ be a faithful typing. Then $\epsilon(M^{\psi}) \simeq_{\mathrm{obs}} \mathbf{O}^{\wedge}(M^{\psi})$.*

$$\mathbf{O}^{\wedge}(M^{\psi}) = \mathsf{d}^{\epsilon(\psi)}, \quad \text{if } \psi \in \mathbf{L}^{\mathrm{cd}}_{\top}$$

otherwise:

$$\mathbf{O}^{\wedge}(\mathsf{c}^{\psi}) = \mathsf{c}^{\epsilon(\psi)}$$

$$\mathbf{O}^{\wedge}(x^{\psi}) = x^{\epsilon(\psi)}$$

$$\mathbf{O}^{\wedge}(\langle {M_1}^{\psi_1}, {M_2}^{\psi_2}\rangle^{\psi_1\times\psi_2}) = \langle \mathbf{O}^{\wedge}({M_1}^{\psi_1}), \mathbf{O}^{\wedge}({M_2}^{\psi_2})\rangle$$

$$\mathbf{O}^{\wedge}((\mathsf{proj}_i M^{\psi_1\times\psi_2})^{\psi_i}) = \mathsf{proj}_i \mathbf{O}^{\wedge}(M^{\psi_1\times\psi_2}), \quad \text{where } i \in \{1,2\}$$

$$\mathbf{O}^{\wedge}((M^{\phi_1\wedge\cdots\wedge\phi_n\to\psi}\{\!|N^{\phi'_1};\cdots;N^{\phi'_n}|\!\})^{\psi}) = \mathbf{O}^{\wedge}(M^{\phi_1\wedge\cdots\wedge\phi_n\to\psi})\mathbf{sup}(\mathbf{O}^{\wedge}(N^{\phi'_1}),\ldots,\mathbf{O}^{\wedge}(N^{\phi'_n}))$$

$$\mathbf{O}^{\wedge}((\lambda x^{\phi_1\wedge\cdots\wedge\phi_n}.M^{\psi})^{\phi_1\wedge\cdots\wedge\phi_n\to\psi}) = \lambda x^{\epsilon(\phi_1)}.\mathbf{O}^{\wedge}(M^{\psi})$$

$$\mathbf{O}^{\wedge}((\mathsf{fix}\, x^{\phi_i}.\{\!|M^{\phi_1};\cdots;M^{\phi_n}|\!\})^{\phi_i}) = \mathsf{fix}\, x^{\epsilon(\phi_i)}.\mathbf{sup}(\mathbf{O}^{\wedge}(M^{\phi_1}),\ldots,\mathbf{O}^{\wedge}(M^{\phi_n}))$$

$$\mathbf{O}^{\wedge}((\mathsf{if}\ N^{\delta^{\mathsf{bool}}}\ \mathsf{then}\ {M_1}^{\psi_1}\ \mathsf{else}\ {M_2}^{\psi_2})^{\psi}) = \mathsf{if}\ \mathbf{O}^{\wedge}(N^{\delta^{\mathsf{bool}}})\ \mathsf{then}\ \mathbf{O}^{\wedge}({M_1}^{\psi_1})\ \mathsf{else}\ \mathbf{O}^{\wedge}({M_2}^{\psi_2})$$

$$\mathbf{O}^{\wedge}(\mathsf{case}(N^{\delta^{\mathsf{nat}}}, M^{\psi_1}, F^{a^{\mathsf{nat}}\to\psi_2})^{\psi}) = \mathsf{case}(\mathbf{O}^{\wedge}(N^{\delta^{\mathsf{nat}}}), \mathbf{O}^{\wedge}(M^{\psi_1}), \mathbf{O}^{\wedge}(F^{a^{\mathsf{nat}}\to\psi_2}))$$

$$\begin{array}{l}\mathbf{O}^{\wedge}(\mathsf{it}(N^{\delta^{\mathsf{nat}}}, \{\!|M^{\phi_1};\cdots;M^{\phi_n}|\!\},\\ \quad \{\!|F^{\phi_1\wedge\cdots\wedge\phi_n\to\phi_1};\ldots;F^{\phi_1\wedge\cdots\wedge\phi_n\to\phi_n}|\!\})^{\phi_i}) =\\ \quad\quad \mathsf{it}(\mathbf{O}^{\wedge}(N^{\delta^{\mathsf{nat}}}), \mathbf{sup}(\mathbf{O}^{\wedge}(M^{\phi_1}),\ldots,\mathbf{O}^{\wedge}(M^{\phi_n})),\\ \quad\quad\quad \mathbf{sup}(\mathbf{O}^{\wedge}(F^{\phi_1\wedge\cdots\wedge\phi_n\to\phi_1}),\ldots,\mathbf{O}^{\wedge}(F^{\phi_1\wedge\cdots\wedge\phi_n\to\phi_n})))\end{array}$$

$$\begin{array}{l}\mathbf{O}^{\wedge}(\mathsf{rec}(N^{\delta^{\mathsf{nat}}}, \{\!|M^{\phi_1};\cdots;M^{\phi_n}|\!\},\\ \quad \{\!|F^{a^{\mathsf{nat}}\to\phi_1\wedge\cdots\wedge\phi_n\to\phi_1};\ldots;F^{a^{\mathsf{nat}}\to\phi_1\wedge\cdots\wedge\phi_n\to\phi_n}|\!\})^{\phi_i}) =\\ \quad\quad \mathsf{rec}(\mathbf{O}^{\wedge}(N^{\delta^{\mathsf{nat}}}), \mathbf{sup}(\mathbf{O}^{\wedge}(M^{\phi_1}),\ldots,\mathbf{O}^{\wedge}(M^{\phi_n})),\\ \quad\quad\quad \mathbf{sup}(\mathbf{O}^{\wedge}(F^{a^{\mathsf{nat}}\to\phi_1\wedge\cdots\wedge\phi_n\to\phi_1}),\ldots,\mathbf{O}^{\wedge}(F^{a^{\mathsf{nat}}\to\phi_1\wedge\cdots\wedge\phi_n\to\phi_n})))\end{array}$$

Figure 8.5: The simplification mapping $\mathbf{O}^{\wedge}$

**Proof.** Since $\mathbf{O}^\wedge(M^\psi) \preceq_{\text{prune}} \epsilon(M^\psi)$ and the $\preceq_{\text{prune}}$ relation is reflexive, we have that $\mathbf{O}^\wedge(M^\psi) \preceq_{\text{prune}} \mathbf{O}^\wedge(M^\psi) \preceq_{\text{prune}} \epsilon(M^\psi)$. So, by Theorem 8.23, $\epsilon(M^\psi) \sim_\psi^\Sigma \mathbf{O}^\wedge(M^\psi)$, and, by Theorem 8.22 (since $\Sigma \vdash_{\text{cd}} M^\psi$ is faithful), we get the result. □

**Example 8.25** *The $\vdash_{\text{cd}}$-typing $\Sigma \vdash_{\text{cd}} {M'}^{\delta^{\mathsf{nat}}}$ of Example 8.11 is faithful.*

*Applying the $\mathbf{O}^\wedge$ simplification mapping we get the following simplification of the term $M$.*

$$\mathbf{O}^\wedge(M'^\delta) = (\lambda f^{(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}\to\mathsf{nat}}. f(\lambda x^{\mathsf{nat}}.3)\mathsf{d}^{\mathsf{nat}} + f(\lambda y^{\mathsf{nat}}.y)v)(\lambda h^{\mathsf{nat}\to\mathsf{nat}}.\lambda z^{\mathsf{nat}}.hz)\ ,$$

*where* $\vdash_{\mathbf{T}} \mathbf{O}^\wedge(M'^\delta) : \mathsf{nat}$, *and* $\mathrm{FV}'(\epsilon(\mathbf{O}^\wedge(M'^\delta))) = \mathbf{O}^\wedge(\Sigma) = \{v^{\mathsf{nat}}\}$ .

## 8.3 A paradigmatic example

In the previous section we have showed a first example of use of cd-types handling a situation in which the argument of a function $\lambda f^\rho.M$ can be assigned different (non comparable) dead-code types that are both needed to discover some dead-code in the body $M$ of the function.

We believe that situations requiring the use of cd-types rise quite naturally in the case of the optimization of programs obtained by extraction from constructive proof of their specification. In fact, when proving theorems in a given theory using systems like Coq [Barras *et al.*, 1996] and Lego [Luo and Pollack, 1992], it is quite common to use lemmas not in their full generality, but in a weaker way. Usually, lemmas are proved by showing that, under some hypotheses (say, three hypothesis $x : \alpha$, $y : \beta$ and $z : \gamma$), there exists a tuple of elements (say a pair $\langle S, T\rangle : \sigma \times \tau$) satisfying a predicate. Consider now a situation in which

- $S$ and $T$ can computed independently, and $S$ depends just from $x, y$, and
- in some use of the lemma only $S$ is needed, while in some other $T$ is needed.

This is a situation in which conjunction is useful. In fact we may assign to the lemma both $\phi_1 = \delta(\alpha) \to \delta(\beta) \to \top(\gamma) \to (\delta(\sigma) \times \top(\tau))$ and $\phi_2 = \delta(\alpha) \to \delta(\beta) \to \delta(\gamma) \to (\delta(\sigma) \times \delta(\tau))$, hence $\phi_1 \wedge \phi_2$.

Frédéric Prost ([Prost, 1997]) has found some (quite complex) examples of the situation described above in the framework of computational geometry. For sake of readability we prefer to illustrate the kind of situation we have in mind by a sketchy example.

**Example 8.26** *Take a function*

$$F = \lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\lambda z^{\mathsf{nat}}.\langle S, T\rangle$$

*of type* $\rho = \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to (\mathsf{nat} \times \mathsf{nat})$, *with* $S = x+y$ *and* $T = x+y+z$. *The function* $F$ *takes* 3 *arguments an returns a pair. Note that* $S$ *and* $T$ *can be computed independently, moreover* $S$ *does not depend from* $z$.

*Consider now*

$$M = \lambda f^{\rho}.\langle \mathsf{proj}_1(fPQR), \mathsf{proj}_2(fUVW)\rangle$$

*of type* $\rho \to (\mathsf{nat} \times \mathsf{nat})$, *for some terms* $P$, $Q$, $R$, $U$, $V$, $W$ *of type* $\mathsf{nat}$. *In the left-hand side of* $MF$ *we only use* $S$, *while in the right-hand side we use only* $T$.

*It is easy to see that in the application*

$$MF$$

*the subterm* $R$ *of* $M$ *is dead-code. It is also easy to check that this can be proved with the system* $\vdash_{\mathrm{cd}}$ *by assuming the cd-conjunction* $\xi = \phi_1 \wedge \phi_2$, *where*

$$\begin{array}{l}\phi_1 = \delta \to \delta \to \omega \to (\delta \times \omega) \text{ and}\\ \phi_2 = \delta \to \delta \to \delta \to (\delta \times \delta),\end{array}$$

*for the* $\lambda$*-abstracted variable* $f$ *in* $M$, *and by assigning a-types* $\phi_1$ *and* $\phi_2$ *to the first and to the second occurrence of* $f$ *in the body of* $M$, *respectively (in such a way that* $R$ *is decorated by* $\top$*). In fact both the cd-types* $\phi_1$ *and* $\phi_2$ *can be assigned to* $F$.

*Note that, since there is no d-type (*$\in \mathbf{L}^{\mathrm{d}}$*) of* $F$ *which entails both* $\phi_1$ *and* $\phi_2$, *this dead-code cannot be found with the systems in Chapter 7.*

## 8.4 Other examples

In this section we present an example involving the construct $\mathsf{fix}$ and an example involving the construct $\mathsf{it}$.

**Example 8.27** *Take the term* $F = \mathsf{fix}\, f^{\rho}.M$, *where* $\rho = \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ *and*

$$M = \lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.\lambda y^{\mathsf{nat}}.\lambda z^{\mathsf{nat}}.\mathsf{if}\ z = 0\ \mathsf{then}\ gx\ \mathsf{else}\ f(\lambda u^{\mathsf{nat}}.3)yx(z-1) \quad .$$

*Let*

- $\phi_1 = (\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$, *and*
- $\phi_2 = (\delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}) \to \delta^{\mathsf{nat}} \to \top^{\mathsf{nat}} \to \delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$.

*It is easy to check that* $\emptyset \vdash_{\mathrm{cd}} (\mathsf{fix}\, f^{\phi_1\wedge\phi_2}.\{\!|M'^{\phi_1}; M''^{\phi_2}|\!\})^{\phi_2}$, *where (writing, for short,* $\delta$ *and* $\top$ *instead of* $\delta^{\mathsf{nat}}$ *and* $\top^{\mathsf{nat}}$*):*

- $M'^{\phi_1} = (\lambda g^{\top\to\delta}.(\lambda x^{\top}.(\lambda y^{\top}.(\lambda z^{\delta}.$
  $(\mathsf{if}\ (z^{\delta} =^{\delta\times\delta\to\delta} 0^{\delta})^{\delta}$
  $\mathsf{then}\ (g^{\top\to\delta}x^{\top})^{\delta}$
  $\mathsf{else}\ ((((f^{\phi_1}(\lambda u^{\top}.3^{\delta})^{\top\to\delta})^{\top\to\top\to\delta\to\delta}y^{\top})^{\top\to\delta\to\delta}x^{\top})^{\delta\to\delta}(z^{\delta} -^{\delta\times\delta\to\delta} 1^{\delta})^{\delta})^{\delta}$
  $)^{\delta})^{\delta\to\delta})^{\top\to\top\to\delta\to\delta})^{\top\to\top\to\delta\to\delta})^{\phi_1}$

- $M''^{\phi_2} = (\lambda g^{\delta\to\delta}.(\lambda x^{\,\delta}.(\lambda y^{\,\top}.(\lambda z^{\,\delta}.$
  $(\mathsf{if}\ (z^\delta =^{\delta\times\delta\to\delta} 0^\delta)^\delta$
  $\mathsf{then}\ (g^{\delta\to\delta}x^\delta)^\delta$
  $\mathsf{else}\ ((((f^{\phi_1}(\lambda u^{\top}.3^\delta)^{\top\to\delta})^{\top\to\top\to\delta\to\delta}y^{\top})^{\top\to\delta\to\delta}x^{\top})^{\delta\to\delta}(z^\delta -^{\delta\times\delta\to\delta} 1^\delta)^\delta)^\delta$
  $)^\delta)^{\delta\to\delta})^{\top\to\delta\to\delta})^{\delta\to\top\to\delta\to\delta})^{\phi_2}$ .

*So, if we consider the term*

$$R = (\mathsf{fix}\, f^\rho.M)(\lambda v^{\mathsf{nat}}.v)Q_1Q_2Q_3,$$

*for some (for simplicity closed) terms $Q_1$, $Q_2$, and $Q_3$, we can derive the following faithful $\vdash_{\mathrm{cd}}$-typing of $R$:*

$$\emptyset \vdash_{\mathrm{cd}} (((((\mathsf{fix}\, f^{\phi_1\wedge\phi_2}.\{\!|M'^{\phi_1}; M''^{\phi_2}|\!\})^{\phi_2}(\lambda v^\delta.v^\delta)^{\delta\to\delta})^{\delta\to\top\to\delta\to\delta}Q_1^\delta)^{\top\to\delta\to\delta}Q_2^\top)^{\delta\to\delta}Q_3^\delta)^\delta.$$

*Then by applying the simplification mapping $\mathbf{O}^\wedge$ and erasing all the d-type decorations we get the following simplified version of $R$:*

$$R' = (\mathsf{fix}\, f^\rho.N)(\lambda v^{\mathsf{nat}}.v)Q_1\mathsf{d}_3^{\mathsf{nat}}Q_3,$$

*where*

$$N = \lambda g^{\,\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\,\mathsf{nat}}.\lambda y^{\,\mathsf{nat}}.\lambda z^{\,\mathsf{nat}}.\mathsf{if}\ z = 0\ \mathsf{then}\ gx\ \mathsf{else}\ f(\lambda u^{\mathsf{nat}}.3)\mathsf{d}_1^{\mathsf{nat}}\mathsf{d}_2^{\mathsf{nat}}(z-1) \quad .$$

*Note that (see Example 7.58) $\emptyset \vdash_1 (\mathsf{fix}\, f^{\phi_1}.M'^{\phi_1})^{\phi_1}$, but* not *$\emptyset \vdash_1 (\mathsf{fix}\, f^{\phi_2}.M'^{\phi_2})^{\phi_2}$. So the the above simplification of $R$ cannot be done by using the systems of Chapter 7. However we can improve the simplification obtained with the mapping $\mathbf{O}^\wedge$ by applying the mapping $\mathbf{O}_2$ of Section 7.3.3.*

*It is in fact easy to see that*

$$\emptyset \vdash_2 (((((\mathsf{fix}\, f^{\phi_2}.N'^{\phi_2})^{\phi_2}(\lambda v^\delta.v^\delta)^{\delta\to\delta})^{\delta\to\top\to\delta\to\delta}Q_1^\delta)^{\top\to\delta\to\delta}\mathsf{d}_3^\top)^{\delta\to\delta},$$

*where*

$$\begin{array}{l} N'^{\phi_2} = (\lambda g^{\delta\to\delta}.(\lambda x^{\,\delta}.(\lambda y^{\,\top}.(\lambda z^{\,\delta}.\\ \quad (\mathsf{if}\ (z^\delta =^{\delta\times\delta\to\delta} 0^\delta)^\delta\\ \qquad \mathsf{then}\ (g^{\delta\to\delta}x^\delta)^\delta\\ \qquad \mathsf{else}\ ((((f^{\phi_1}(\lambda u^\delta.3^\delta)^{\delta\to\delta})^{\delta\to\top\to\delta\to\delta}\mathsf{d}_1^\delta)^{\top\to\delta\to\delta}\mathsf{d}_2^\top)^{\delta\to\delta}(z^\delta -^{\delta\times\delta\to\delta} 1^\delta)^\delta)^\delta\\ \quad )^\delta)^{\delta\to\delta})^{\top\to\delta\to\delta})^{\delta\to\top\to\delta\to\delta})^{\phi_2}, \end{array}$$

*is a faithful $\vdash_2$-typing. Then by applying the mapping $\mathbf{O}_2$ and erasing the d-type decoration we get the following simplification of $R'$:*

$$R'' = (\mathsf{fix}\, f^\rho.P)(\lambda v^{\mathsf{nat}}.v)Q_1Q_3,$$

*where*

$$P = \lambda g^{\,\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\,\mathsf{nat}}.\lambda z^{\,\mathsf{nat}}.\mathsf{if}\ z = 0\ \mathsf{then}\ gx\ \mathsf{else}\ f(\lambda u^{\mathsf{nat}}.3)\mathsf{d}_1^{\mathsf{nat}}(z-1) \quad .$$

**Example 8.28** *Take the term* $N = \mathsf{it}(\mathsf{n}, M, F)$ *of type* $\rho = (\mathsf{nat} \to \mathsf{nat}) \to \mathsf{nat} \to \mathsf{nat}$*, where*

$$M = \lambda h^{\mathsf{nat}\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.hx \quad ,$$

*and*

$$F = \lambda m^{\rho}.\lambda g^{\mathsf{nat}\to\mathsf{nat}}.\lambda y^{\mathsf{nat}}.(m(\lambda u^{\mathsf{nat}}.u)5 + m(\lambda z^{\mathsf{nat}}.3)y) + mgy \quad .$$

*Let*

- $\phi_1 = (\delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}) \to \delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$*, and*
- $\phi_2 = (\top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}) \to \top^{\mathsf{nat}} \to \delta^{\mathsf{nat}}$*.*

*It is easy to check that* $\emptyset \vdash_{\mathrm{cd}} \mathsf{it}(\mathsf{n}^{\delta}, \{\!|M'^{\phi_1}; M''^{\phi_2}|\!\}, \{\!|F'^{\phi_1\wedge\phi_2\to\phi_1}; F''^{\phi_1\wedge\phi_2\to\phi_2}|\!\})$*, where (writing, for short,* $\delta$ *and* $\top$ *instead of* $\delta^{\mathsf{nat}}$ *and* $\top^{\mathsf{nat}}$*):*

- $M'^{\phi_1} = (\lambda h^{\delta\to\delta}.(\lambda x^{\delta}.(h^{\delta\to\delta}x^{\delta})^{\delta})^{\delta\to\delta})^{\phi_1}$*,*
- $M''^{\phi_2} = (\lambda h^{\top\to\delta}.(\lambda x^{\top}.(h^{\top\to\delta}x^{\top})^{\delta})^{\top\to\delta})^{\phi_2}$*,*
- $F'^{\phi_1\wedge\phi_2\to\phi_1} = (\lambda m^{\phi_1\wedge\phi_2}.(\lambda g^{\delta\to\delta}.(\lambda y^{\delta}.((m^{\phi_1}(\lambda u^{\delta}.u^{\delta})^{\delta\to\delta})^{\delta\to\delta}5^{\delta})^{\delta} +^{\delta\times\delta\to\delta}$
  $(((m^{\phi_2}(\lambda z^{\top}.3^{\delta})^{\top\to\delta})^{\top\to\delta}y^{\top})^{\delta} +^{\delta\times\delta\to\delta} ((m^{\phi_1}g^{\delta\to\delta})^{\delta\to\delta}y^{\delta}))^{\delta})^{\delta\to\delta})^{\phi_1})^{\phi_1\wedge\phi_2\to\phi_1}$*, and*
- $F''^{\phi_1\wedge\phi_2\to\phi_2} = (\lambda m^{\phi_1\wedge\phi_2}.(\lambda g^{\top\to\delta}.(\lambda y^{\top}.((m^{\phi_1}(\lambda u^{\delta}.u^{\delta})^{\delta\to\delta})^{\delta\to\delta}5^{\delta})^{\delta} +^{\delta\times\delta\to\delta}$
  $(((m^{\phi_2}(\lambda z^{\top}.3^{\delta})^{\top\to\delta})^{\top\to\delta}y^{\top})^{\delta} +^{\delta\times\delta\to\delta} ((m^{\phi_2}g^{\top\to\delta})^{\top\to\delta}y^{\top}))^{\delta})^{\top\to\delta})^{\phi_2})^{\phi_1\wedge\phi_2\to\phi_2}$ *.*

*So, if we consider the term*

$$P = \mathsf{it}(\mathsf{n}, M, F)(\lambda v^{\mathsf{nat}}.7)Q,$$

*for some (for simplicity closed) term* $Q$*, we can derive the following faithful* $\vdash_{\mathrm{cd}}$*-typing of* $P$*:*

$$\emptyset \vdash_{\mathrm{cd}} ((\mathsf{it}(\mathsf{n}^{\delta}, \{\!|M'^{\phi_1}; M''^{\phi_2}|\!\}, \{\!|F'^{\phi_1\wedge\phi_2\to\phi_1}; F''^{\phi_1\wedge\phi_2\to\phi_2}|\!\})^{\phi_2}(\lambda v^{\top}.7^{\delta})^{\top\to\delta})^{\top\to\delta}Q^{\top})^{\delta}.$$

*Then by applying the simplification mapping* $\mathbf{O}^{\wedge}$ *and erasing all the d-type decorations we get the following simplified version of* $P$*:*

$$P' = \mathsf{it}(\mathsf{n}, M, F)(\lambda v^{\mathsf{nat}}.7)\mathsf{d}^{\mathsf{nat}}.$$

*Note that this dead-code cannot be found with the systems in Chapter 7. Moreover, in this case, using the simplification mapping* $\mathbf{O}_2$ *does not introduce further simplifications.*

## 8.5 Summary

In this chapter we have extended the dead-code inference system, $\vdash_1$, and the associated dead-code elimination mapping, $\mathbf{O}_1$, presented in Chapter 7 to conjunctive dead-code types.

At the moment we have no idea of how the inference algorithm for $\vdash_1$ could be extended to $\vdash_{\mathrm{cd}}$.

# Chapter 9

# An algorithm for rank 2 conjunctive dead-code analysis

In this chapter we define an algorithms for detecting and removing dead-code based on a restriction to rank 2 of the conjunctive dead-code analysis presented in Chapter 8.

The example introduced at the beginning of Chapter 8 can still be handled when restricting to the rank 2 conjunctive dead-code types assignment system introduced in this chapter (Definition 9.6) and, more in general, the restriction to rank 2 still captures a significant class of optimizations. Indeed the class of examples discussed in Section 8.3 was originally investigated to find applications for the rank 2 conjunctive dead-code type inference algorithm described in this chapter.

On the other hand, is is easy to find terms $N$ with dead-code that can be detected by using the system $\vdash_{\mathrm{cd}}$ of Chapter 8 but cannot be detected when restricting to rank 2 conjunction: simply replace a subterm $(\lambda f^{\rho}.M)F$ of type $\sigma$ in which, to detect some dead-code, is necessary to assign to $f$ two different and non-comparable dead-code types, by the ground observational equivalent term $(\lambda g^{\rho\to\sigma}.gF)(\lambda f^{\rho}.M)$ (for instance, apply this construction to the term $N$ of the example at the beginning of Chapter 8).

A preliminary version of the material presented in this chapter appeared in [Damiani and Prost, 1997], where the dead-code inference algorithm is not given.

## 9.1 Rank 2 conjunctive dead-code types

**Definition 9.1 (Rank 0 conjunctive dead-code types)** *The language* $\mathbf{L}^0$ *of* rank 0 conjunctive dead-code types *(*d-0-types *for short), ranged over by* $\phi$*, is defined by the following grammar:* $\phi ::= a^{\iota} \mid \phi \to \phi \mid \phi \times \phi$*, where* $a \in \{\delta, \top\}$ *and* $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$.

Note that $\mathbf{L}^0$ is exactly the language $\mathbf{L}^{\mathrm{d}}$ of dead-code types without conjunction of Chapter 7.

**Definition 9.2 (Rank 1 and rank 2 conjunctive dead-code types)** *1. The language* $\mathbf{L}^1$ *of* rank 1 conjunctive dead-code types *(*d-1-types *for short), ranged over by* $\xi$*, is defined by:*

- $\phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{L}^1$*, if* $n \geq 1$*,* $\phi_1, \ldots, \phi_n \in \mathbf{L}^0$*, and* $\epsilon(\phi_1) = \cdots = \epsilon(\phi_n)$*.*

*2. The language* $\mathbf{L}^2$ *of* rank 2 conjunctive dead-code types *(*d-2-types *for short), ranged over by* $\psi$*, is inductively defined by:*

- $\phi \in \mathbf{L}^2$*, if* $\phi \in \mathbf{L}^0$*.*
- $\xi \to \psi \in \mathbf{L}^2$*, if* $\xi \in \mathbf{L}^1$ *and and* $\psi \in \mathbf{L}^2$*.*
- $\psi_1 \times \psi_2 \in \mathbf{L}^2$*, if* $\psi_1, \psi_2 \in \mathbf{L}^2$*.*

Notice that $\mathbf{L}^0 \subseteq \mathbf{L}^1$, $\mathbf{L}^0 \subseteq \mathbf{L}^2$, and $\mathbf{L}^1 \cap \mathbf{L}^2 = \mathbf{L}^0$.

**Definition 9.3 ($\delta$-d-types and $\top$-d-types)** *1. For* $i \in \{0, 1, 2\}$*, the set* $\mathbf{L}^i_\delta$ *of* $\delta$*-d-*$i$*-types is the subset of* $\mathbf{L}^i$ *containing only the basic property constructor* $\delta$*.*

*2. The set* $\mathbf{L}^2_\top$ *of* $\top$*-d-2-types is inductively defined by:*

- $\top^\iota \in \mathbf{L}^2_\top$*, if* $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$*,*
- $\xi \to \psi \in \mathbf{L}^2_\top$*, if* $\xi \in \mathbf{L}^1$ *and* $\psi \in \mathbf{L}^2_\top$*,*
- $\psi_1 \times \psi_2 \in \mathbf{L}^2_\top$*, if* $\psi_1, \psi_2 \in \mathbf{L}^2_\top$*.*

*The sets* $\mathbf{L}^0_\top$ *of* $\top$*-d-0-types is defined by* $\mathbf{L}^0_\top = \mathbf{L}^2_\top \cap \mathbf{L}^0$*, and the set* $\mathbf{L}^1_\top$ *of* $\top$*-d-1-types is defined by:*

- $\phi_1 \wedge \cdots \wedge \phi_n \in \mathbf{L}^1_\top$*, if* $n \geq 1$*,* $\phi_1, \ldots, \phi_n \in \mathbf{L}^0_\top$*, and* $\epsilon(\phi_1) = \cdots = \epsilon(\phi_n)$*.*

## 9.2 Detecting and removing dead-code

### 9.2.1 Rank 2 conjunctive dead-code type entailment relation

In this section we introduce an entailment relation between d-2-types, denoted $\leq_{\text{r2cd}}$ which is the restriction to d-2-types of the entailment relation between cd-types $\leq_{\text{cd}}$ of Definition 8.5. The $\leq_{\text{r2cd}}$ entailment relation is defined on the top of the entailment relation for d-types $\leq_1$ of Definition 7.9.

**Definition 9.4 (Entailment relation $\leq_{\text{r2cd}}$)** *Let* $\psi_1, \psi_2 \in \mathbf{L}^2$*. We write* $\psi_1 \leq_{\text{r2cd}} \psi_2$ *to mean that* $\psi_1 \leq_{\text{r2cd}} \psi_2$ *is derivable by the rules in Fig. 9.1. By* $\cong_{\text{r2cd}}$ *we denote the equivalence relation induced by* $\leq_{\text{r2cd}}$*.*

It is immediate to show that $\leq_{\text{r2cd}}$ is reflexive and transitive, and that it behaves in the same way of $\leq_1$ on $\mathbf{L}^0$, and of $\leq_{\text{cd}}$ on $\mathbf{L}^2$, i.e.,

$$(\top)\ \frac{\psi_2 \in \mathbf{L}^2_\top \quad \epsilon(\psi_1) = \epsilon(\psi_2)}{\psi_1 \leq_{\text{r2cd}} \psi_2} \qquad (\text{Ref})\ \psi \leq_{\text{r2cd}} \psi$$

$$(\rightarrow)\ \frac{\psi \leq_{\text{r2cd}} \psi' \quad \forall i \in \{1,\ldots,m\}.\exists j \in \{1,\ldots,n\}.\phi'_j \leq_1 \phi_i}{\phi_1 \wedge \cdots \wedge \phi_m \rightarrow \psi \leq_{\text{r2cd}} \phi'_1 \wedge \cdots \wedge \phi'_n \rightarrow \psi'}$$

$$(\times)\ \frac{\psi_1 \leq_{\text{r2cd}} \psi'_1 \quad \psi_2 \leq_{\text{r2cd}} \psi'_2}{\psi_1 \times \psi_2 \leq_{\text{r2cd}} \psi'_1 \times \psi'_2}$$

Figure 9.1: Entailment rules for d-2-types (system $\leq_{\text{r2cd}}$)

1. for all $\phi, \phi' \in \mathbf{L}^0$, $\phi \leq_{\text{r2cd}} \phi'$ if and only if $\phi \leq_1 \phi'$, and
2. for all $\psi, \psi' \in \mathbf{L}^2$, $\psi \leq_{\text{r2cd}} \psi'$ if and only if $\psi \leq_{\text{cd}} \psi'$.

Obviously the $\leq_{\text{r2cd}}$ relation between dead-code types is sound w.r.t. the p.e.r. interpretation of Definition 8.2.

**Theorem 9.5 (Soundness of $\leq_{\text{r2cd}}$)** *$\psi_1 \leq_{\text{r2cd}} \psi_2$ implies $[\![\psi_1]\!] \subseteq [\![\psi_2]\!]$.*

**Proof.** Since $\psi_1 \leq_{\text{r2cd}} \psi_2$ implies $\psi_1 \leq_{\text{cd}} \psi_2$, the result follows immediately by Theorem 8.6 (soundness of the entailment $\leq_{\text{cd}}$). □

### 9.2.2 Rank 2 conjunctive dead-code type assignment

In this section we introduce a restriction of the assignment system $\vdash_{\text{cd}}$ (of Definition 8.8) which uses only rank 2 dead-code conjunctive types.

If $x^\rho$ is a term variable of type $\rho$, an assumption for $x^\rho$ is an expression of the shape $x^\rho : \xi$, or $x : \xi$ for short, where $\xi \in \mathbf{L}^1$, and $\epsilon(\xi) = \rho$. A basis is a set $\Sigma$ of d-1-types assumptions for term variables.

**Definition 9.6 (Rank 2 conjunctive d-type assignment system $\vdash_{\text{r2cd}}$)** *We write $\Sigma \vdash_{\text{r2cd}} M^\psi$ to mean that $\Sigma \vdash M^\psi$ can be derived by the rules in Fig. 9.2.*

If $\Sigma \vdash_{\text{r2cd}} M^\psi$ then $M^\psi$ has written in it the d-types assigned to its subterms. We say that $M^\psi$ is a *decorated* term. Note that, being $\vdash_{\text{r2cd}}$ an inference system, the same terms can have different decorations. Let $\Lambda_{\mathbf{T}}^{\vdash_{\text{r2cd}}}$ be the set of decorated terms, i.e.,

$$\Lambda_{\mathbf{T}}^{\vdash_{\text{r2cd}}} = \{M^\psi \mid \Sigma \vdash_{\text{r2cd}} M^\psi \text{ for some d-2-type } \psi \text{ and basis } \Sigma\}.$$

**Remark 9.7** *As explained in Remark 8.9, a $\{\!|\}$-sequence $\{\!|N^{\phi'_1}; \cdots; N^{\phi'_n}|\!\}$, is just a way of storing $n$ ($n \geq 1$) decorations of a subterm. These decorations correspond to different uses of the subterm. When $n = 1$ we will write, for simplicity, $N^{\phi'_1}$ instead of $\{\!|N^{\phi'_1}|\!\}$.*

*Indeed (see Section 9.4) this space-consuming code duplication can easily be avoided in the implementation of the d-type inference algorithm presented in Section 9.3.*

$$\text{(Var)}\ \frac{\phi_i \leq_0 \phi_i'}{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash x^{\phi_i'}}\ 1 \leq i \leq n \qquad \text{(Con)}\ \frac{\mathbf{L}(\mathsf{c}) \leq_0 \phi}{\Sigma \vdash \mathsf{c}^{\phi}}$$

$$(\to \text{I})\ \frac{\Sigma, x : \phi_1 \wedge \cdots \wedge \phi_n \vdash M^{\psi}}{\Sigma \vdash (\lambda x^{\phi_1 \wedge \cdots \wedge \phi_n}.M^{\psi})^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi}}$$

$$(\to \text{E})\ \frac{\begin{array}{c}\Sigma \vdash M^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi} \qquad \Sigma \vdash N^{\phi_1'} \ \cdots\ \Sigma \vdash N^{\phi_n'} \\ \epsilon(\phi_1' \wedge \cdots \wedge \phi_n') = \epsilon(\phi_1 \wedge \cdots \wedge \phi_n) \\ \psi \notin \mathbf{L}^2_\top \text{ implies } \forall i \in \{1, \ldots, n\}.\phi_i' \leq_0 \phi_i\end{array}}{\Sigma \vdash (M^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi} \{\!|N^{\phi_1'}; \ldots; N^{\phi_n'}|\!\})^{\psi}}$$

$$(\times \text{I})\ \frac{\Sigma \vdash M_1{}^{\psi_1} \qquad \Sigma \vdash M_2{}^{\psi_2}}{\Sigma \vdash \langle M_1{}^{\psi_1}, M_2{}^{\psi_2} \rangle^{\psi_1 \times \psi_2}} \qquad (\times \text{E}_i)\ \frac{\Sigma \vdash M^{\psi_1 \times \psi_2}}{\Sigma \vdash (\mathsf{proj}_i M^{\psi_1 \times \psi_2})^{\psi_i}}\ i \in \{1, 2\}$$

$$\text{(Fix)}\ \frac{\Sigma, x : \phi \vdash M^{\phi}}{\Sigma \vdash (\mathsf{fix}\, x^{\,\phi}.M^{\phi})^{\phi}}$$

$$\text{(If)}\ \frac{\begin{array}{c}\Sigma \vdash N^{a_0\,\mathsf{bool}} \qquad \Sigma \vdash M_1{}^{\psi_1} \qquad \Sigma \vdash M_2{}^{\psi_2} \\ \psi_1 \leq_{\text{r2cd}} \psi \qquad \psi_2 \leq_{\text{r2cd}} \psi \\ a_0 = \top \text{ implies } \psi \in \mathbf{L}^2_\top\end{array}}{\Sigma \vdash (\mathsf{if}\ N^{a_0\,\mathsf{bool}}\ \mathsf{then}\ M_1{}^{\psi_1}\ \mathsf{else}\ M_2{}^{\psi_2})^{\psi}}$$

$$\text{(Case)}\ \frac{\begin{array}{c}\Sigma \vdash N^{a_0\,\mathsf{nat}} \qquad \Sigma \vdash M^{\psi_1} \qquad \Sigma \vdash F^{a^{\mathsf{nat}} \to \psi_2} \\ \psi_1 \leq_{\text{r2cd}} \psi \qquad \psi_2 \leq_{\text{r2cd}} \psi \\ a_0 = \top \text{ implies } \psi \in \mathbf{L}^2_\top\end{array}}{\Sigma \vdash \mathsf{case}(N^{a_0\,\mathsf{nat}}, M^{\psi_1}, F^{a^{\mathsf{nat}} \to \psi_2})^{\psi}}$$

$$\text{(It)}\ \frac{\begin{array}{c}\Sigma \vdash N^{a_0\,\mathsf{nat}} \qquad \Sigma \vdash M^{\phi_1} \qquad \Sigma \vdash F^{\phi_2 \to \phi_3} \\ \phi_1 \leq_0 \phi \qquad \phi_2 \to \phi_3 \leq_0 \phi \to \phi \\ a_0 = \top \text{ implies } \phi \in \mathbf{L}^0_\top\end{array}}{\Sigma \vdash \mathsf{it}(N^{a_0\,\mathsf{nat}}, M^{\phi_1}, F^{\phi_2 \to \phi_3})^{\phi}}$$

$$\text{(Rec)}\ \frac{\begin{array}{c}\Sigma \vdash N^{a_0\,\mathsf{nat}} \qquad \Sigma \vdash M^{\phi_1} \qquad \Sigma \vdash F^{a^{\mathsf{nat}} \to \phi_2 \to \phi_3} \\ \phi_1 \leq_0 \phi \qquad a^{\mathsf{nat}} \to \phi_2 \to \phi_3 \leq_0 \delta^{\mathsf{nat}} \to \phi \to \phi \\ a_0 = \top \text{ implies } \phi \in \mathbf{L}^0_\top\end{array}}{\Sigma \vdash \mathsf{rec}(N^{a_0\,\mathsf{nat}}, M^{\phi_1}, F^{a^{\mathsf{nat}} \to \phi_2 \to \phi_3})^{\phi}}$$

Figure 9.2: Rules for d-2-type assignment (system $\vdash_{\text{r2cd}}$)

Now we can state the soundness of the $\vdash_{\text{r2cd}}$-type assignment w.r.t. the p.e.r. semantics of Definition 8.2.

**Theorem 9.8 (Soundness of $\vdash_{\text{r2cd}}$)** *Let $\Sigma \vdash_{\text{r2cd}} M^{\psi}$. Then $\epsilon(M^{\psi}) \sim^{\Sigma}_{\psi} \epsilon(M^{\psi})$.*

**Proof.** Note that $\Sigma \vdash_{\text{r2cd}} M^{\psi}$ implies $\Sigma \vdash_{\text{cd}} M^{\psi}$. Then the result follows immediately by soundness of the assignment system $\vdash_{\text{cd}}$ (Theorem 8.13). □

### 9.2.3 Dead-code elimination

In this section we introduce a dead-code elimination mapping $\mathbf{O}^{\wedge 2}$ that, given a $\vdash_{\text{r2cd}}$-decorated term $M^{\psi}$, returns an optimized version of $\epsilon(M^{\psi})$. It is easy to see that, for every $\vdash_{\text{r2cd}}$-decorated term $M$, the mapping $\mathbf{O}^{\wedge 2}$ behaves in the same way of the mapping $\mathbf{O}^{\wedge}$ of Definition 8.19. Indeed $\mathbf{O}^{\wedge 2}$ is simply the restriction of $\mathbf{O}^{\wedge}$ to $\vdash_{\text{r2cd}}$-decorated terms.

#### Dummy variables

Dummy variables and the pruning relation on terms, $\preceq_{\text{prune}}$, have been introduced in Section 8.2.3. Dummy variables are simply placeholders for some dead-code removed, therefore each occurrence of a dummy variable in a program is dead-code. So they can be handled in a special by the dead-code type assignment system $\vdash_{\text{cd}}$. In particular, given a term $M$ obtained by a correct application of the mapping $\mathbf{O}^{\wedge 2}$ (see Theorem 9.11), it is safe to allow that any d-0-type $\phi \in \mathbf{L}^0(\rho)$ can be assigned to a dummy variable $\mathsf{d}^{\rho}$ occurring in $M$. This can be ensured by:

- forbidding dummy variables to occur in bases, and
- adding to the rules in Fig. 9.2 the following rule

$$(\mathsf{d})\ \frac{\vdash_{\mathbf{T}} \mathsf{d}^{\rho} : \rho \qquad \phi \in \mathbf{L}^0(\rho)}{\vdash \mathsf{d}^{\phi}}\ .$$

For every term $M$, define $\text{FV}'(M)$ be the set of the free non dummy variables in $M$.

#### The simplification mapping

Let $\Lambda_{\mathbf{T}}^{\vdash_{\text{r2cd}}}$ be the set of all decorated terms which are defined according to Definition 9.6, i.e., $\Lambda_{\mathbf{T}}^{\vdash_{\text{r2cd}}} = \{M^{\psi} \mid \Sigma \vdash_{\text{r2cd}} M^{\psi}$ for some d-2-type $\psi$ and basis $\Sigma\}$.

**Definition 9.9 (Simplification mapping $\mathbf{O}^{\wedge 2}$ on terms)** *1. The function*

$$\mathbf{O}^{\wedge 2} : \Lambda_{\mathbf{T}}^{\vdash_{\text{r2cd}}} \to \Lambda_{\mathbf{T}}$$

*is defined by the clauses in Fig. 9.3.*

$\mathbf{O}^{\wedge 2}(M^{\psi}) = \mathsf{d}^{\epsilon(\psi)}, \quad \text{if } \psi \in \mathbf{L}^2_{\top}$

otherwise:

$\mathbf{O}^{\wedge 2}(\mathsf{c}^{\psi}) = \mathsf{c}^{\epsilon(\psi)}$

$\mathbf{O}^{\wedge 2}(x^{\psi}) = x^{\epsilon(\psi)}$

$\mathbf{O}^{\wedge 2}(\langle {M_1}^{\psi_1}, {M_2}^{\psi_2}\rangle^{\psi_1 \times \psi_2}) = \langle \mathbf{O}^{\wedge 2}({M_1}^{\psi_1}), \mathbf{O}^{\wedge 2}({M_2}^{\psi_2})\rangle$

$\mathbf{O}^{\wedge 2}((\mathsf{proj}_i M^{\psi_1 \times \psi_2})^{\psi_i}) = \mathsf{proj}_i \mathbf{O}^{\wedge 2}(M^{\psi_1 \times \psi_2}), \quad \text{where } i \in \{1, 2\}$

$\mathbf{O}^{\wedge 2}((M^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi} \{\!| N^{\phi'_1}; \cdots; N^{\phi'_n} |\!\})^{\psi}) = \mathbf{O}^{\wedge 2}(M^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi})\mathbf{sup}(\mathbf{O}^{\wedge 2}(N^{\phi'_1}), \ldots, \mathbf{O}^{\wedge 2}(N^{\phi'_n}))$

$\mathbf{O}^{\wedge 2}((\lambda x^{\phi_1 \wedge \cdots \wedge \phi_n}.M^{\psi})^{\phi_1 \wedge \cdots \wedge \phi_n \to \psi}) = \lambda x^{\epsilon(\phi_1)}.\mathbf{O}^{\wedge 2}(M^{\psi})$

$\mathbf{O}^{\wedge 2}((\mathsf{fix}\, x^{\phi}.M^{\phi})^{\phi}) = \mathsf{fix}\, x^{\epsilon(\phi)}.\mathbf{O}^{\wedge 2}(M^{\phi})$

$\mathbf{O}^{\wedge 2}((\mathsf{if}\ N^{\delta^{\mathsf{bool}}}\ \mathsf{then}\ {M_1}^{\psi_1}\ \mathsf{else}\ {M_2}^{\psi_2})^{\psi}) = \mathsf{if}\ \mathbf{O}^{\wedge 2}(N^{\delta^{\mathsf{bool}}})\ \mathsf{then}\ \mathbf{O}^{\wedge 2}({M_1}^{\psi_1})\ \mathsf{else}\ \mathbf{O}^{\wedge 2}({M_2}^{\psi_2})$

$\mathbf{O}^{\wedge 2}(\mathsf{case}(N^{\delta^{\mathsf{nat}}}, M^{\psi_1}, F^{a^{\mathsf{nat}} \to \psi_2})^{\psi}) = \mathsf{case}(\mathbf{O}^{\wedge 2}(N^{\delta^{\mathsf{nat}}}), \mathbf{O}^{\wedge 2}(M^{\psi_1}), \mathbf{O}^{\wedge 2}(F^{a^{\mathsf{nat}} \to \psi_2}))$

$\mathbf{O}^{\wedge 2}(\mathsf{it}(N^{\delta^{\mathsf{nat}}}, M^{\phi_1}, F^{\phi_2 \to \phi_3})^{\phi}) = \mathsf{it}(\mathbf{O}^{\wedge 2}(N^{\delta^{\mathsf{nat}}}), \mathbf{O}^{\wedge 2}(M^{\phi_1}), \mathbf{O}^{\wedge 2}(F^{\phi_2 \to \phi_3}))$

$\mathbf{O}^{\wedge 2}(\mathsf{rec}(N^{\delta^{\mathsf{nat}}}, M^{\phi_1}, F^{a^{\mathsf{nat}} \to \phi_2 \to \phi_3})^{\phi}) = \mathsf{rec}(\mathbf{O}^{\wedge 2}(N^{\delta^{\mathsf{nat}}}), \mathbf{O}^{\wedge 2}(M^{\phi_1}), \mathbf{O}^{\wedge 2}(F^{a^{\mathsf{nat}} \to \phi_2 \to \phi_3}))$

Figure 9.3: Mapping $\mathbf{O}^{\wedge 2}$ on terms

*2. If $\Sigma$ is a basis then*

$$\mathbf{O}^{\wedge 2}(\Sigma) = \{x^{\epsilon(\phi_1)} \mid x : \phi_1 \wedge \cdots \wedge \phi_n \in \Sigma,\ n \geq 1 \text{ and } \exists i \in \{1, \ldots, n\}.\phi_i \notin \mathbf{L}^0_\top\}.$$

To prove the correctness of the simplification mapping $\mathbf{O}^{\wedge 2}$ we identify a subset of $\vdash_{\text{r2cd}}$-typings for which the $\sim^{\Sigma}_{\psi}$ relation implies the $\simeq^{\text{gnd}}_{\text{obs}}$ relation.

**Definition 9.10 (Faithful d-2-type assignment)** $\Sigma \vdash_{\text{r2cd}} M^{\phi}$ *is a* faithful *d-2-type assignment statement if $\phi \in \mathbf{L}^0_\delta$, and for all $x : \phi_1 \wedge \cdots \wedge \phi_n \in \Sigma$, $n = 1$ and $\phi_1 \in \mathbf{L}^0_\top \cup \mathbf{L}^0_\delta$.*

**Theorem 9.11** *Let $\Sigma \vdash_{\text{r2cd}} M^{\psi}$ be a faithful typing. Then $\epsilon(M^{\psi}) \simeq^{\text{gnd}}_{\text{obs}} \mathbf{O}^{\wedge 2}(M^{\psi})$.*

**Proof.** $\Sigma \vdash_{\text{r2cd}} M^{\psi}$ implies $\Sigma \vdash_{\text{cd}} M^{\psi}$. So the result follows immediately by Theorem 8.24. □

**Example 9.12** *The $\vdash_{\text{cd}}$-typing $\Sigma \vdash_{\text{cd}} M'^{\delta^{\mathsf{nat}}}$ of Example 8.11 is also a faithful $\vdash_{\text{r2cd}}$-typing.*

*It is easy to see that by applying the simplification mapping $\mathbf{O}^{\wedge 2}$ we obtain the same simplification of Example 8.25.*

## 9.3 An algorithm for rank 2 dead-code type inference

In this section we deal with the problem of defining a complete inference algorithm for the dead-code type assignment system $\vdash_{\text{r2cd}}$. To this aim the main problem is to use the inference rules to detect a faithful decoration showing the maximum amount of dead code, i.e., assigning an $\top$-d-type to all the maximal subterms that can be proved to be dead code by the system. The application of the simplification mapping $\mathbf{O}^{\wedge 2}$ is then trivial.

### 9.3.1 A syntax-directed version of $\vdash_{\text{r2cd}}$

The dead-code type assignment system $\vdash_{\text{r2cd}}$ presented in Section 9.1 is not immediately suggestive of an algorithm. The main problems are the possibility of assuming conjunctions of arbitrary length for the free variables and the fact that there are rules ((If) and (Case)) that make an explicit use of the $\leq_{\text{r2cd}}$ entailment.

So the first step to design an inference algorithm is the definition of a syntax-directed version of the d-type assignment system $\vdash_{\text{r2cd}}$ which enforces some discipline in the use of the assumptions and uses only the $\leq_0$ entailment relation. To definition of the syntax-directed dead-code type assignment system require the introduction of a stratification on the set of the d-2-types.

**Definition 9.13 (Sets $\mathbf{L}[p]$)** *For every natural number $p$, let $\mathbf{L}[p]$ denote the set of the d-2-types inductively defined by:*

- $\phi \in \mathbf{L}[p]$, *if* $\phi \in \mathbf{L}^0$

- $\xi \to \psi \in \mathbf{L}[p+1]$, *if* $\xi \in \mathbf{L}^1$ *and* $\psi \in \mathbf{L}[p]$
- $\psi_1 \times \psi_2 \in \mathbf{L}[p]$, *if* $\psi_1, \psi_2 \in \mathbf{L}[p]$.

To have an intuition of the meaning of such a stratification on $\mathbf{L}^2$ observe that, for every natural number $p$, if $\psi \in \mathbf{L}[p]$ and $\psi$ does not contain occurrences of the constructor $\times$, then $\psi$ is of the shape

$$\xi_1 \to \cdots \to \xi_q \to \phi \ ,$$

where $0 \leq q \leq p$, $\xi_1, \ldots, \xi_q \in \mathbf{L}^1$, and $\phi \in \mathbf{L}^0$. I.e., at most the first $p$ antecedents of $\psi$ are allowed to be conjunctions. For instance: $\phi_1 \to (\phi_2 \wedge \phi_2') \to \phi_3 \in \mathbf{L}[2]$, but $\phi_1 \to (\phi_2 \wedge \phi_2') \to (\phi_3 \wedge \phi_3') \to \phi_4 \notin \mathbf{L}[2]$.

In the judgments of the syntax-directed d-type assignment system there are two basis: the first contains a set of variables for which it is allowed to assume only d-0-types (and not d-1-types), while the second contains exactly the free variables of the term that does not occur in the first one. To enforce the invariant on the second basis of a judgement we will use (in the the assignment rules for application, $\mathsf{if}$, $\mathsf{case}$, $\mathsf{it}$ and $\mathsf{rec}$, see Fig. 9.5) an operation ($\uplus$ in the following definition) which allows to combine two basis by "appending" the d-1-types in two different assumptions for a same program variable.

**Definition 9.14 (Operation $\uplus$)** *Let $\Sigma$, $\Sigma'$ be two basis, then $\Sigma \uplus \Sigma'$ denotes the basis*

$$\begin{array}{l}\{x : \xi \wedge \xi' \mid x : \xi \in \Sigma \text{ and } x : \xi' \in \Sigma'\} \\ \cup\{x : \xi \mid x : \xi \in \Sigma \text{ and } x \notin \mathrm{FV}(\Sigma')\} \\ \cup\{x : \xi' \mid x : \xi' \in \Sigma' \text{ and } x \notin \mathrm{FV}(\Sigma)\} \ .\end{array}$$

Moreover each judgment is parameterized by a natural number $p$. The idea is that, if the judgment $\Upsilon; \Sigma \vdash^{[\mathbf{p}]} M^{\psi}$ holds, then

- $\psi \in \mathbf{L}[p]$, and
- for every $x^{\rho} : \phi_1 \wedge \cdots \wedge \phi_n \in \Sigma$, we have that $n \geq 1$ and each $\phi_i$ $(1 \leq i \leq n)$ corresponds to a free occurrence of $x^{\rho}$ in the decorated term $M^{\psi}$.

Finally, to avoid the use of the $\leq_{\mathrm{r2cd}}$ entailment in the rules for $\mathsf{if}$ and $\mathsf{it}$ we introduce a "Syntax Directed Condition" (suitable for the definition of the inference algorithm in Section 9.3.3), formalized by the predicate SDC (see Definition 9.15 below), such that for every natural number $p$ and $\psi_1, \psi_2 \in \mathbf{L}^{\mathrm{st}}[p]$, if $\mathrm{SDC}(p, \psi_1, \psi_2, \psi)$ holds, then $\psi \in \mathbf{L}^{\mathrm{st}}[p]$, $\psi \leq_{\mathrm{r2cd}} \psi_1$, and $\psi \leq_{\mathrm{r2cd}} \psi_2$.

**Definition 9.15 (Predicate** SDC**)** *Let $p$ be a natural number and $\psi_1, \psi_2, \psi \in \mathbf{L}^{\mathrm{st}}[p]$. The predicate $\mathrm{SDC}(p, \psi_1, \psi_2, \psi)$ is inductively defined by the clauses in Fig. 9.4.*

**Definition 9.16 (Syntax-directed d-2-type assignment system)** *We write $\Upsilon; \Sigma \vdash^{[\mathbf{p}]} M^{\phi}$ if $\Upsilon; \Sigma \vdash^{[\mathbf{p}]} M^{\psi}$ can be derived by the rules in Fig. 9.5.*

$\mathrm{SDC}(0, \phi_1, \phi_2, \phi) = \phi_1 \leq_0 \phi$ and $\phi_2 \leq_0 \phi$

$\mathrm{SDC}(p+1, \xi_1 \to \psi_1, \xi_2 \to \psi_2, \xi \to \psi) = (\xi = \xi_1 \wedge \xi_2)$ and $\mathrm{SDC}(p, \psi_1, \psi_2, \psi)$

$\mathrm{SDC}(p+1, \psi_1 \times \psi'_1, \psi_2 \times \psi'_2, \psi \times \psi') = \mathrm{SDC}(p+1, \psi_1, \psi_2, \psi)$ and $\mathrm{SDC}(p+1, \psi'_1, \psi'_2, \psi')$

Figure 9.4: Predicate SDC

It is easy to see that, for every $p \geq 0$, all the $\vdash^{[\mathbf{p}]}$-decorated terms are also $\vdash_{\mathrm{r2cd}}$-decorated terms. I.e., the following theorem holds.

**Theorem 9.17** *$\Upsilon; \Sigma \vdash^{[\mathbf{p}]} M^{\psi}$ implies $\Upsilon \cup \Sigma \vdash_{\mathrm{r2cd}} M^{\psi}$.* □

We also have that all the dead-code that can be detected by using the d-2-type assignment system $\vdash_{\mathrm{r2cd}}$ can also be detected by using its syntax-directed version. More precisely the following theorem holds.

**Theorem 9.18** *Let $\vdash_{\mathbf{T}} M : \rho$. Then, for each faithful $\vdash_{\mathrm{r2cd}}$-typing of $M$, $\Upsilon \vdash_{\mathrm{r2cd}} M'^{\phi}$, there is a $\vdash^{[\mathbf{0}]}$-typing of $M$, $\Upsilon; \emptyset \vdash^{[\mathbf{0}]} M''^{\phi}$, such that*

$$\mathbf{O}^{\wedge 2}(M'^{\phi}) = \mathbf{O}^{\wedge 2}(M''^{\phi}).$$

□

In the rest of the chapter we will call *faithful $\vdash^{[\mathbf{0}]}$-typing* any $\vdash^{[\mathbf{0}]}$ typing $\Upsilon; \emptyset \vdash^{[\mathbf{0}]} M''^{\phi}$ such that $\Upsilon \vdash_{\mathrm{r2cd}} M''^{\phi}$ is a faithful $\vdash_{\mathrm{r2cd}}$-typing (i.e., such that $\phi \in \mathbf{L}^0_{\delta}$ and, for all $x : \phi' \in \Upsilon$, $\phi' \in \mathbf{L}^0_{\top} \cup \mathbf{L}^0_{\delta}$).

In the following sections it is developed (using the technique presented in Section 7.6) an algorithm that, given a $\mathrm{PCFP}^{\mathcal{T}}$ term, returns a decoration of the term containing constructor variables and a set of d-constraints involving constructor variables. The output of the algorithm characterizes all the possible faithful $\vdash^{[\mathbf{0}]}$-typings of the term, i.e., any solution of the set of constraints corresponds to a faithful $\vdash^{[\mathbf{0}]}$-typing, and vice versa. Moreover, the set of constraints has a maximal solution, i.e., a solution corresponding to a $\vdash^{[\mathbf{0}]}$-typing showing all the dead-code that can be proved using the type assignment system $\vdash^{[\mathbf{0}]}$. The language of constraints is the same of Section 7.6.1, so the maximal solution can be found in an effective way by using the constraints solving algorithm $\rightsquigarrow$ of Definition 7.67.

### 9.3.2 Rank 2 dead-code type schemes

**Definition 9.19 (Dead-code type patterns)** *Let $\mathcal{A}$ be the set of* constructor variables*, ranged by $\alpha$, $\beta$, $\gamma$, ....*

$$(\mathrm{Var}_1^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\phi \leq_0 \phi'}{\Upsilon, x:\phi; \emptyset \vdash^{[\mathbf{p}]} x^{\phi'}} \qquad (\mathrm{Var}_2^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\phi \leq_0 \phi'}{\Upsilon; \{x:\phi\} \vdash^{[\mathbf{p}]} x^{\phi'}}\ x \notin \mathrm{FV}(\Upsilon)$$

$$(\mathrm{Con}^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\mathbf{L}(\mathsf{c}) \leq_0 \phi}{\Upsilon; \emptyset \vdash^{[\mathbf{p}]} \mathsf{c}^{\phi}} \qquad (\to \mathrm{I}_1^{[\mathbf{p}=\mathbf{0}]})\ \frac{\Upsilon, x:\phi; \Sigma \vdash^{[\mathbf{0}]} M^{\phi'}}{\Upsilon; \Sigma \vdash^{[\mathbf{0}]} (\lambda x^{\phi}.M^{\phi'})^{\phi\to\phi'}}$$

$$(\to \mathrm{I}_2^{[\mathbf{p}\geq\mathbf{1}]})\ \frac{\Upsilon; \Sigma, x:\phi_1\wedge\cdots\wedge\phi_n \vdash^{[\mathbf{p}-\mathbf{1}]} M^{\psi}}{\Upsilon; \Sigma \vdash^{[\mathbf{p}]} (\lambda x^{\phi_1\wedge\cdots\wedge\phi_n}.M^{\psi})^{\phi_1\wedge\cdots\wedge\phi_n\to\psi}}$$

$$(\to \mathrm{I}_3^{[\mathbf{p}\geq\mathbf{1}]})\ \frac{\Upsilon; \Sigma \vdash^{[\mathbf{p}-\mathbf{1}]} M^{\psi}}{\Upsilon; \Sigma \vdash^{[\mathbf{p}]} (\lambda x^{\phi}.M^{\psi})^{\phi\to\psi}}\ x \notin \epsilon(\Upsilon)\cup\epsilon(\Sigma)$$

$$(\to \mathrm{E}^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\begin{array}{c}\Upsilon; \Sigma \vdash^{[\mathbf{p}+\mathbf{1}]} M^{\phi_1\wedge\cdots\wedge\phi_n\to\psi} \quad \Upsilon; \Sigma_1 \vdash^{[\mathbf{0}]} N^{\phi'_1} \ \cdots\ \Upsilon; \Sigma_n \vdash^{[\mathbf{0}]} N^{\phi'_n} \\ \psi \notin \mathbf{L}^2_\top \text{ implies } \forall i \in \{1,\ldots,n\}.\phi'_i \leq_0 \phi_i\end{array}}{\Upsilon; \Sigma \uplus \Sigma_1 \uplus \cdots \uplus \Sigma_n \vdash^{[\mathbf{p}]} (M^{\phi_1\wedge\cdots\wedge\phi_n\to\psi}\{\!|N^{\phi'_1};\cdots;N^{\phi'_n}|\!\})^{\psi}}$$

$$(\times \mathrm{I}^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\Upsilon; \Sigma_1 \vdash^{[\mathbf{p}]} {M_1}^{\psi_1} \quad \Upsilon; \Sigma_1 \vdash^{[\mathbf{p}]} {M_2}^{\psi_2}}{\Upsilon; \Sigma_1 \uplus \Sigma_2 \vdash^{[\mathbf{p}]} \langle {M_1}^{\psi_1}, {M_2}^{\psi_2}\rangle^{\psi_1\times\psi_2}}$$

$$(\times \mathrm{E}_i^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\Upsilon; \Sigma \vdash^{[\mathbf{p}]} M^{\psi_1\times\psi_2}}{\Upsilon; \Sigma \vdash^{[\mathbf{p}]} (\mathsf{proj}_i M^{\psi_1\times\psi_2})^{\psi_i}}\ i \in \{1,2\}$$

$$(\mathrm{Fix}^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\Sigma; \Upsilon, x:\phi \vdash^{[\mathbf{0}]} M^{\phi}}{\Upsilon; \Sigma \vdash^{[\mathbf{p}]} (\mathsf{fix}\, x^{\phi}.M^{\phi})^{\phi}}$$

$$(\mathrm{If}^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\begin{array}{c}\Upsilon; \Sigma \vdash^{[\mathbf{0}]} N^{{a_0}^{\mathsf{bool}}} \quad \Upsilon; \Sigma_1 \vdash^{[\mathbf{p}]} {M_1}^{\psi_1} \quad \Upsilon; \Sigma_2 \vdash^{[\mathbf{p}]} {M_2}^{\psi_2} \\ a_0 = \top \text{ implies } \psi \in \mathbf{L}^2_\top\end{array}}{\Upsilon; \Sigma \uplus \Sigma_1 \uplus \Sigma_2 \vdash^{[\mathbf{p}]} (\mathsf{if}\ N^{{a_0}^{\mathsf{bool}}}\ \mathsf{then}\ {M_1}^{\psi_1}\ \mathsf{else}\ {M_2}^{\psi_2})^{\psi}}\ \mathrm{SDC}(p,\psi_1,\psi_2,\psi)$$

$$(\mathrm{Case}^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\begin{array}{c}\Upsilon; \Sigma_1 \vdash^{[\mathbf{p}]} M^{\psi_1} \quad \Upsilon; \Sigma_2 \vdash^{[\mathbf{p}]} F^{{a_1}^{\mathsf{nat}}\wedge\cdots\wedge{a_n}^{\mathsf{nat}}\to\psi_2} \\ \Upsilon; \Sigma \vdash^{[\mathbf{0}]} N^{{a_0}^{\mathsf{nat}}} \quad a^{\mathsf{nat}} \leq_0 {a_1}^{\mathsf{nat}} \ \cdots\ a^{\mathsf{nat}} \leq_0 {a_n}^{\mathsf{nat}} \\ a_0 = \top \text{ implies } \psi \in \mathbf{L}^2_\top\end{array}}{\Upsilon; \Sigma \uplus \Sigma_1 \uplus \Sigma_2 \vdash^{[\mathbf{p}]} \mathsf{case}(N^{{a_0}^{\mathsf{nat}}}, M^{\psi_1}, F^{a^{\mathsf{nat}}\to\psi_2})^{\psi}}\ \mathrm{SDC}(p,\psi_1,\psi_2,\psi)$$

$$(\mathrm{It}^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\begin{array}{c}\Upsilon; \Sigma \vdash^{[\mathbf{0}]} N^{{a_0}^{\mathsf{nat}}} \quad \Upsilon; \Sigma_1 \vdash^{[\mathbf{0}]} M^{\phi_1} \quad \Upsilon; \Sigma_2 \vdash^{[\mathbf{0}]} F^{\phi_2\to\phi_3} \\ \phi_1 \leq_0 \phi \quad \phi_2 \to \phi_3 \leq_0 \phi \to \phi \\ a_0 = \top \text{ implies } \psi \in \mathbf{L}^2_\top\end{array}}{\Upsilon; \Sigma \uplus \Sigma_1 \uplus \Sigma_2 \vdash^{[\mathbf{p}]} \mathsf{it}(N^{{a_0}^{\mathsf{nat}}}, M^{\phi_1}, F^{\phi_2\to\phi_3})^{\phi}}$$

$$(\mathrm{Rec}^{[\mathbf{p}\geq\mathbf{0}]})\ \frac{\begin{array}{c}\Upsilon; \Sigma \vdash^{[\mathbf{0}]} N^{{a_0}^{\mathsf{nat}}} \quad \Upsilon; \Sigma_1 \vdash^{[\mathbf{0}]} M^{\phi_1} \quad \Upsilon; \Sigma_2 \vdash^{[\mathbf{0}]} F^{a^{\mathsf{nat}}\to\phi_2\to\phi_3} \\ \phi_1 \leq_0 \phi \quad a^{\mathsf{nat}} \to \phi_2 \to \phi_3 \leq_0 \delta^{\mathsf{nat}} \to \phi \to \phi \\ a_0 = \top \text{ implies } \psi \in \mathbf{L}^2_\top\end{array}}{\Upsilon; \Sigma \uplus \Sigma_1 \uplus \Sigma_2 \vdash^{[\mathbf{p}]} \mathsf{rec}(N^{{a_0}^{\mathsf{nat}}}, M^{\phi_1}, F^{a^{\mathsf{nat}}\to\phi_2\to\phi_3})^{\phi}}$$

Figure 9.5: Rules for syntax-directed d-2-type assignment (system $\vdash^{[\mathbf{p}]}$)

1. *The language* $\mathbf{P}^0$ *of* d-0-type patterns *(*d-0-patterns *for short), ranged over by* $\theta$*, is defined from the grammar of Definition 9.1 by replacing* $a \in \{\delta, \top\}$ *by* $\alpha \in \mathcal{A}$*, i.e.* $\theta ::= \alpha^\iota \mid \theta \to \theta \mid \theta \times \theta$*, where* $\alpha \in \mathcal{A}$ *and* $\iota \in \{\mathsf{nat}, \mathsf{bool}\}$*.*

2. *The language* $\mathbf{P}^1$ *of* d-1-type patterns *(*d-1-patterns *for short) ranged over by* $\nu$*, is defined according to the clauses of Definition 9.2.1. and*

3. *The languages* $\mathbf{P}^2$ *of* d-2-type patterns *(*d-2-patterns *for short) ranged over by* $\eta$*, is defined according to the clauses of Definition 9.2.2.*

The function $\epsilon : \mathbf{L}^0 \cup \mathbf{L}^1 \cup \mathbf{L}^2 \to \mathbf{T}$ is extended in the obvious way to d-patterns.

D-constraints, renaming and instantiations are defined as in Section 7.6.1.

**Definition 9.20 (Rank 2 d-type schemes)** *An* d-2-type scheme *is a pair* $\langle \eta, \mathcal{E} \rangle$ *where* $\eta$ *is an d-2-pattern and* $\mathcal{E}$ *is a finite set of constraints.*

### 9.3.3 An algorithm for inferring $\vdash^{[\mathbf{0}]}$-typings

To define the algorithm we need some preliminary notations. By **newa**() we denote a 0-ary function that, whenever called, returns a fresh constructor variable. The functions **fresh**$(\cdot)$, **vars**$(\cdot)$, **tail**$(\cdot)$, $\mathbf{cs}_{=}(\cdot,\cdot)$, $\mathbf{cs}_{\leq_1}(\cdot,\cdot)$, $\mathbf{ucs}_{\leq_1}(\cdot,\cdot)$, and **sts**$(\cdot)$, are defined as in Section 7.6.2.

**Definition 9.21 (Sets $\mathbf{P}[p]$)** *For every natural number* $p$*, let* $\mathbf{P}[p]$ *denote the set of the d-2-patterns inductively defined by:*

- $\theta \in \mathbf{P}[p]$*, if* $\theta \in \mathbf{P}^0$
- $\nu \to \eta \in \mathbf{P}[p+1]$*, if* $\nu \in \mathbf{P}^1$ *and* $\eta \in \mathbf{P}[p]$
- $\eta_1 \times \eta_2 \in \mathbf{P}[p]$*, if* $\eta_1, \eta_2 \in \mathbf{P}[p]$*.*

Consider the rules (If$^{[\mathbf{p}\geq\mathbf{0}]}$) and (Case$^{[\mathbf{p}\geq\mathbf{0}]}$) in Fig. 9.5, and the d-2-types $\psi_1$, $\psi_2$, and $\psi$ that occur in these rules. Let $\eta_1, \eta_2 \in \mathbf{P}[p]$ be d-2-patterns corresponding respectively to $\psi_1, \psi_2 \in \mathbf{L}^{\mathrm{st}}[p]$. Then $\mathcal{J}(p, \eta_1, \eta_2)$, where $\mathcal{J}$ is the algorithm in Fig. 9.6, returns a d-2-pattern $\eta \in \mathbf{P}[p]$ and a set of constraints $\mathcal{E}$ that characterize all the d-2-types $\psi$ such that $\mathrm{SDC}(p, \psi_1, \psi_2, \psi)$ holds. More precisely, we have the following proposition.

**Proposition 9.22** *Let* $\eta_1, \eta_2 \in \mathbf{P}[p]$*,* $\epsilon(\eta_1) = \epsilon(\eta_2) = \rho$*, and* $\langle \eta, \mathcal{E} \rangle = \mathcal{J}(p, \eta_1, \eta_2)$*. Then*

1. $\eta \in \mathbf{P}[p]$*, and*
2. *for every instantiation* $\mathbf{i} \in \mathbf{sat}(\mathcal{E})$*,* $\mathrm{SDC}(p, \mathbf{i}(\eta_1), \mathbf{i}(\eta_2), \mathbf{i}(\eta))$*, and*
3. *for every instantiation* $\mathbf{i}$ *and d-2-type* $\psi$ *such that* $\mathrm{SDC}(p, \mathbf{i}(\eta_1), \mathbf{i}(\eta_2), \psi)$*, there is an instantiation* $\mathbf{i}' \in \mathbf{sat}(\mathcal{E})$ *such that* $\mathbf{i}'(\eta_1) = \mathbf{i}(\eta_1)$*,* $\mathbf{i}'(\eta_2) = \mathbf{i}(\eta_2)$*, and* $\mathbf{i}'(\eta) \leq_{\mathrm{r2cd}} \psi$*.* □

$$
\begin{array}{lll}
\mathcal{J}(0,\theta,\theta') = & \text{let} & \theta'' = \mathbf{fresh}(\epsilon(\theta)) \\
& \text{in} & \langle \theta'', \mathbf{cs}_{\leq_1}(\theta,\theta'') \cup \mathbf{cs}_{\leq_1}(\theta',\theta'') \rangle \text{ end} \\
\mathcal{J}(p,\eta,\eta') = & \text{case } \langle \eta, \eta' \rangle \text{ of} & \\
& \langle {\alpha_1}^\iota, {\alpha_2}^\iota \rangle : \text{let } \beta = \mathbf{newa}() & \\
& \qquad \text{in } \langle \beta, \{\alpha_1 \sqsubseteq \beta,\ \alpha_2 \sqsubseteq \beta\} \rangle \text{ end} & \\
& \langle \nu \to \eta, \nu' \to \eta' \rangle : \text{let } \langle \eta'', \mathcal{E} \rangle = \mathcal{J}(p-1,\eta,\eta') & \\
& \qquad \text{in } \langle \nu \wedge \nu' \to \eta'', \mathcal{E} \rangle \text{ end} & \\
& \langle \eta_1 \times \eta_2, \eta_1' \times \eta_2' \rangle : \text{let } \langle \eta_1'', \mathcal{E}_1 \rangle = \mathcal{J}(p,\eta_1,\eta_1') \text{ and } \langle \eta_2'', \mathcal{E}_2 \rangle = \mathcal{J}(p,\eta_2,\eta_2') & \\
& \qquad \text{in } \langle \eta_1'' \times \eta_2'', \mathcal{E}_1 \cup \mathcal{E}_2 \rangle \text{ end} &
\end{array}
$$

Figure 9.6: Algorithm $\mathcal{J}$

$$
\begin{array}{lll}
\mathcal{W}(P) = & \text{let} & \Theta = \mathbf{fresh}(\mathrm{FV}(P)) \\
& & \text{and } \langle \emptyset, P', \mathcal{E} \rangle = \mathcal{W}(0,\Theta,P) \\
& \text{in} & \langle \Theta, P', \mathcal{E} \rangle \text{ end}
\end{array}
$$

Figure 9.7: Algorithm $\mathcal{W}$

We can now proceed to define the inference algorithm $\mathcal{W}$. This algorithm is presented in Fig. 9.7, 9.8 and 9.9. Let $\vdash_{\mathbf{T}} M : \rho$, if $\mathcal{W}(M) = \langle \Theta, {M'}^\theta, \mathcal{E} \rangle$ then $\Theta$ is a basis that associates to each term variable in $\mathrm{FV}(M)$ an d-0-pattern, ${M'}^\theta$ is a pattern-decorated term, and $\mathcal{E}$ is a finite set of constraints. We will prove that the triple $\langle \Theta, M^\theta, \mathcal{E} \rangle$ represents the $\vdash_{\mathrm{r2cd}}$-typings of $M$. More precisely, for any $\Upsilon$ and ${M''}^\phi$ such that $\epsilon(\Upsilon) = \mathrm{FV}(M)$ and $\epsilon({M''}^\phi) = M$, we have that $\Upsilon; \emptyset \vdash^{[\mathbf{0}]} {M''}^\phi$ implies $\Upsilon = \mathbf{i}(\Theta)$ and ${M''}^\phi = \mathbf{i}({M'}^\theta)$, for some $\mathbf{i}$ that satisfies $\mathcal{E}$.

Correctness and completeness of the inference (w.r.t. $\vdash^{[\mathbf{p}]}$-typings containing as many $\top$ constructors as possible, i.e., showing a maximum amount of dead-code) is expressed by the following lemma.

**Lemma 9.23** *$\vdash_{\mathbf{T}} M : \rho$, $\Gamma \subseteq \mathrm{FV}(M)$, $\Theta = \mathbf{fresh}(\Gamma)$, and $\mathcal{W}(p,\Theta,M) = \langle \Xi, {M'}^\eta, \mathcal{E} \rangle$ imply*

1. *if $\mathbf{i}$ is the maximum of $\mathbf{sat}(\mathcal{E})$, then $\mathbf{i}(\Theta); \mathbf{i}(\Xi) \vdash^{[\mathbf{p}]} \mathbf{i}({M'}^\eta)$, and*
2. *for all $\Upsilon$, $\Sigma$ and ${M''}^\psi$ such that $\epsilon(\Upsilon) = \Gamma$, $\epsilon(\Sigma) = \epsilon(\Xi)$ and $\epsilon({M''}^\psi) = M$, if $\Upsilon; \Sigma \vdash^{[\mathbf{p}]} {M''}^\psi$ then exists $\mathbf{i} \in \mathbf{sat}(\mathcal{E})$ such that $\mathbf{i}(\Theta) = \Upsilon$, $\mathbf{i}(\Xi) = \Sigma$ and $\mathbf{i}({M'}^\eta) = {M''}^\psi$.* □

We are interested in faithful $\vdash^{[\mathbf{0}]}$-typings, so we want to restrict the set of solutions of the constraints generated by the algorithm to those that correspond to faithful d-typings. This can be done as shown by the following theorem.

$\mathcal{W}(p, \Theta, \mathsf{c}) =$
  let $\langle \theta, \mathcal{E} \rangle = \mathbf{sts}(\mathsf{c})$ in $\langle \emptyset, \mathsf{c}^{\theta}, \mathcal{E} \rangle$ end
$\mathcal{W}(p, \Theta, x^{\rho}) =$
  let $\theta = \mathbf{fresh}(\rho)$ and $\theta' = \mathbf{fresh}(\rho)$
  in if $x \in \epsilon(\Theta)$
    then $\langle \emptyset, x^{\theta'}, \mathbf{cs}_{\leq_1}(\Theta(x), \theta') \rangle$
    else $\langle \{x : \theta\}, x^{\overline{\theta'}}, \mathbf{cs}_{\leq_1}(\theta, \theta') \rangle$ end
$\mathcal{W}(p, \Theta, \lambda\, x^{\rho}.M) =$
  let $\theta = \mathbf{fresh}(\rho)$ in if $p = 0$
    then let $\langle \Xi, M'^{\theta'}, \mathcal{E} \rangle = \mathcal{W}(0, \Theta \cup \{x : \theta\}, M)$
      in $\langle \Xi, (\lambda\, x^{\theta}.M'^{\theta'})^{\theta \to \theta'}, \mathcal{E} \rangle$ end
    else let $\langle \Xi, M'^{\eta}, \mathcal{E} \rangle = \mathcal{W}(p-1, \Theta, M)$
      in case $\Xi$ of
        $\Xi', x : \nu$: $\langle \Xi', (\lambda\, x^{\nu}.M'^{\eta})^{\nu \to \eta}, \mathcal{E} \rangle$
        _: $\langle \Xi, (\lambda\, x^{\theta}.M'^{\eta})^{\theta \to \eta}, \mathcal{E} \rangle$ end end
$\mathcal{W}(p, \Theta, MN) =$
  let $\langle \Xi_0, M'^{\theta_1 \wedge \cdots \wedge \theta_n \to \eta}, \mathcal{E}_0 \rangle = \mathcal{W}(p+1, \Theta, M)$
    and $\langle \Xi, N'^{\theta}, \mathcal{E} \rangle = \mathcal{W}(0, \Theta, N)$
    and, for each $l \in \{1, \ldots, n\}$, $\langle \Xi_l, N_l^{\theta'_l}, \mathcal{E}_l \rangle = \mathbf{r}_l(\langle \Xi, N'^{\theta}, \mathcal{E} \rangle)$,
      where $\mathbf{r}_l$ is a fresh renaming of all the constructor variables not in $\Theta$
  in $\langle \Xi_0 \uplus \Xi_1 \uplus \cdots \uplus \Xi_n,$
   $(M'^{\theta_1 \wedge \cdots \wedge \theta_n \to \eta} \{\!|N'^{\theta'_1}; \ldots; N'^{\theta'_n}|\!\})^{\eta},$
   $\{(\delta \text{ in } \mathbf{tail}(\eta)) \Rightarrow (\mathcal{E}_0 \cup \bigcup_{1 \leq l \leq n} (\mathcal{E}_l \cup \mathbf{cs}_{\leq_1}(\theta'_l, \theta_l)))\} \rangle$ end
$\mathcal{W}(p, \Theta, \langle M_1, M_2 \rangle) =$
  let $\langle \Xi_1, M_1'^{\eta_1}, \mathcal{E}_1 \rangle = \mathcal{W}(p, \Theta, M_1)$
    and $\langle \Xi_2, M_2'^{\eta_2}, \mathcal{E}_2 \rangle = \mathcal{W}(p, \Theta, M_2)$
  in $\langle \Xi_1 \uplus \Xi_2, (\langle M_1'^{\eta_1}, M_2'^{\eta_2} \rangle)^{\eta_1 \times \eta_2}, \mathcal{E}_1 \cup \mathcal{E}_2 \rangle$ end
$\mathcal{W}(p, \Theta, \mathsf{proj}_i M) =$
  let $\langle \Xi, M'^{\eta_1 \times \eta_2}, \mathcal{E} \rangle = \mathcal{W}(p, \Theta, M)$
  in $\langle \Xi, (\mathsf{proj}_i M'^{\eta_1 \times \eta_2})^{\eta_i}, \mathcal{E} \rangle$ end
$\mathcal{W}(p, \Theta, \mathsf{fix}\, x^{\rho}.M) =$
  let $\theta_1 = \mathbf{fresh}(\rho)$
    and $\langle \Xi, M'^{\theta_2}, \mathcal{E} \rangle = \mathcal{W}(0, \Theta \cup \{x : \theta_1\}, M)$
  in $\langle \Xi, (\mathsf{fix}\, x^{\theta}.M'^{\theta})^{\theta}, \mathcal{E} \cup \mathbf{cs}_{=}(\theta_1, \theta_2) \rangle$ end
$\mathcal{W}(p, \Theta, \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2) =$
  let $\langle \Xi_0, N'^{\alpha^{\mathsf{bool}}}, \mathcal{E}_0 \rangle = \mathcal{W}(p, \Theta, N)$
    and $\langle \Xi_1, M_1'^{\eta_1}, \mathcal{E}_1 \rangle = \mathcal{W}(p, \Theta, M_1)$
    and $\langle \Xi_2, M_2'^{\eta_2}, \mathcal{E}_2 \rangle = \mathcal{W}(p, \Theta, M_2)$
    and $\langle \eta, \mathcal{E} \rangle = \mathcal{J}(p, \eta_1, \eta_2)$
  in $\langle \Xi_0 \uplus \Xi_1 \uplus \Xi_2,$
   $(\mathsf{if}\ N'^{\alpha^{\mathsf{bool}}}\ \mathsf{then}\ M_1'^{\theta_1}\ \mathsf{else}\ M_2'^{\theta_2})^{\eta},$
   $\{(\delta \text{ in } \mathbf{tail}(\eta)) \Rightarrow (\{\alpha \equiv \delta\} \cup \mathcal{E}_0 \cup \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E})\} \rangle$ end

Figure 9.8: Algorithm $\mathcal{W}$ (continue in Fig. 9.9)

$$
\begin{array}{l}
\mathcal{W}(p,\Theta,\mathsf{case}(N,M,F)) = \\
\quad \text{let} \quad \langle \Xi_0, {N'}^{\alpha_0{}^{\mathsf{nat}}}, \mathcal{E}_0\rangle = \mathcal{W}(p,\Theta,N) \\
\qquad \text{and } \langle \Xi_1, {M'}^{\eta_1}, \mathcal{E}_1\rangle = \mathcal{W}(p,\Theta,M) \\
\qquad \text{and } \langle \Xi_2, {F'}^{\alpha_1{}^{\mathsf{nat}}\wedge\cdots\wedge\alpha_n{}^{\mathsf{nat}}\to\eta_2}, \mathcal{E}_2\rangle = \mathcal{W}(p,\Theta,F) \\
\qquad \text{and } \alpha = \mathbf{newa}() \\
\qquad \text{and } \langle \eta, \mathcal{E}\rangle = \mathcal{J}(p,\eta_1,\eta_2) \\
\quad \text{in} \quad \langle \Xi_0 \uplus \Xi_1 \uplus \Xi_2, \\
\qquad (\mathsf{case}({N'}^{\alpha_0{}^{\mathsf{nat}}}, {M'}^{\theta_1}, {F'}^{\alpha^{\mathsf{nat}}\to\theta_2})^{\theta}, \\
\qquad \{(\delta \text{ in } \mathbf{tail}(\eta)) \Rightarrow (\{\alpha_0 \equiv \delta, \alpha \sqsubseteq \alpha_1, \ldots, \alpha \sqsubseteq \alpha_n\} \cup \mathcal{E}_0 \cup \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E})\}\rangle \ \text{ end} \\
\mathcal{W}(p,\Theta,\mathsf{it}(N,M,F)) = \\
\quad \text{let} \quad \langle \Xi_0, {N'}^{\alpha_0{}^{\mathsf{nat}}}, \mathcal{E}_0\rangle = \mathcal{W}(0,\Theta,N) \\
\qquad \text{and } \langle \Xi_1, {M'}^{\theta_1}, \mathcal{E}_1\rangle = \mathcal{W}(0,\Xi,M) \\
\qquad \text{and } \langle \Xi_3, {F'}^{\theta_2\to\theta_3}, \mathcal{E}_2\rangle = \mathcal{W}(0,\Theta,F) \\
\qquad \text{and } \theta = \mathbf{fresh}(\epsilon(\theta_1)) \\
\quad \text{in} \quad \langle \Xi_0 \uplus \Xi_1 \uplus \Xi_2, \\
\qquad (\mathsf{it}({N'}^{\alpha_0{}^{\mathsf{nat}}}, {M'}^{\theta_1}, {F'}^{\theta_2\to\theta_3})^{\theta}, \\
\qquad \{(\delta \text{ in } \mathbf{tail}(\theta)) \Rightarrow (\{\alpha_0 \equiv \delta\} \cup \mathcal{E}_0 \cup \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathbf{ucs}_{\leq_1}(\theta_1,\theta) \cup \\
\qquad\qquad\qquad \mathbf{ucs}_{\leq_1}(\theta_2 \to \theta_3, \theta \to \theta))\}\rangle \ \text{ end} \\
\mathcal{W}(p,\Theta,\mathsf{rec}(N,M,F)) = \\
\quad \text{let} \quad \langle \Xi_0, {N'}^{\alpha_0{}^{\mathsf{nat}}}, \mathcal{E}_0\rangle = \mathcal{W}(0,\Theta,N) \\
\qquad \text{and } \langle \Xi_1, {M'}^{\theta_1}, \mathcal{E}_1\rangle = \mathcal{W}(0,\Theta,M) \\
\qquad \text{and } \langle \Xi_3, {F'}^{\alpha_1{}^{\mathsf{nat}}\to\theta_2\to\theta_3}, \mathcal{E}_2\rangle = \mathcal{W}(0,\Theta,F) \\
\qquad \text{and } \theta = \mathbf{fresh}(\epsilon(\theta_1)) \\
\quad \text{in} \quad \langle \Xi_0 \uplus \Xi_1 \uplus \Xi_2, \\
\qquad (\mathsf{rec}({N'}^{\alpha_0{}^{\mathsf{nat}}}, {M'}^{\theta_1}, {F'}^{\alpha_1{}^{\mathsf{nat}}\to\theta_2\to\theta_3})^{\theta}, \\
\qquad \{(\delta \text{ in } \mathbf{tail}(\theta)) \Rightarrow (\{\alpha_0 \equiv \delta\} \cup \mathcal{E}_0 \cup \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathbf{ucs}_{\leq_1}(\theta_1,\theta) \cup \\
\qquad\qquad\qquad \mathbf{ucs}_{\leq_1}(\alpha_1{}^{\mathsf{nat}} \to \theta_2 \to \theta_3, \delta^{\mathsf{nat}} \to \theta \to \theta))\}\rangle \ \text{ end}
\end{array}
$$

Figure 9.9: Algorithm $\mathcal{W}$ (continue from Fig. 9.8)

**Theorem 9.24** *Let* $\vdash_{\mathbf{T}} M : \rho$ *and* $\mathcal{W}(M) = \langle \Theta, M'^{\theta}, \mathcal{E} \rangle$. *If* $\mathbf{i}$ *is the maximum of* $\mathbf{sat}(\mathcal{E} \cup \mathbf{faithful}(\Theta, \theta))$ *then* $\mathbf{i}(\Theta); \emptyset \vdash^{[\mathbf{0}]} \mathbf{i}(M'^{\theta})$ *is a faithful assignment showing the maximum amount of dead code, where*
$\mathbf{faithful}(\Theta, \theta) = \bigcup_{x:\theta' \in \Theta} \{(\delta \text{ in } \mathbf{tail}(\theta')) \Rightarrow \{\gamma \equiv \delta | \gamma \in \mathbf{vars}(\theta')\}\ \cup\ \{\alpha \equiv \delta | \alpha \in \mathbf{vars}(\theta)\}$.
□

The constraint $(\delta \text{ in } \mathbf{tail}(\theta')) \Rightarrow \{\gamma \equiv \delta | \gamma \in \mathbf{vars}(\theta')\}$ means that $\theta$ must be instantiated either to an $\top$-d-0-type or to an $\delta$-d-0-type.

**Example 9.25** *Let* $M$ *be the PCFP*$^{\mathcal{T}}$ *term of Example 8.11. Let* $\theta_1 = (\alpha_1 \to \alpha_2) \to \alpha_3 \to \alpha_4$, $\theta'_1 = (\alpha'_1 \to \alpha'_2) \to \alpha'_3 \to \alpha'_4$, $\theta_2 = (\beta_1 \to \beta_2) \to \beta_3 \to \beta_4$, *and* $\theta'_2 = (\beta'_1 \to \beta'_2) \to \beta'_3 \to \beta'_4$. *We have* $\mathcal{W}(M) = \langle \Theta, M'^{\gamma_5^{\mathsf{nat}}}, \mathcal{E} \rangle$ *where (writing, for short,* $\zeta$ *instead of* $\zeta^{\mathsf{nat}}$*):* $\Theta = \{u : \gamma_1, v : \gamma_2\}$, $M'^{\theta} =$

$$
\begin{array}{l}
((\lambda f^{\theta_1 \wedge \theta_2}. \\
\quad (((f^{\theta'_1}(\lambda x^{\alpha_5}.3^{\alpha_6})^{\alpha_5 \to \alpha_6})^{\alpha'_3 \to \alpha'_4} u^{\gamma'_1})^{\alpha'_4} \\
\qquad\qquad +^{\gamma_3 \times \gamma_4 \to \gamma_5}((f^{\theta'_2}(\lambda y^{\beta_5}.y^{\beta_6})^{\beta_5 \to \beta_6})^{\beta'_3 \to \beta'_4} v^{\gamma'_2})^{\beta'_4})^{\gamma_5})^{(\theta_1 \wedge \theta_2) \to \gamma_5} \\
\qquad\qquad\quad \{\!|(\lambda h^{\alpha_7 \to \alpha_8}.(\lambda z^{\alpha_9}.(h^{\alpha'_7 \to \alpha'_8} z^{\alpha'_9})^{\alpha'_8})^{\alpha_9 \to \alpha'_8})^{(\alpha_7 \to \alpha_8) \to \alpha_9 \to \alpha'_8}; \\
\qquad\qquad\qquad (\lambda h^{\beta_7 \to \beta_8}.(\lambda z^{\beta_9}.(h^{\beta'_7 \to \beta'_8} z^{\beta'_9})^{\beta'_8})^{\beta_9 \to \beta'_8})^{(\beta_7 \to \beta_8) \to \beta_9 \to \beta'_8} |\!\})^{\gamma_5} \quad ,
\end{array}
$$

*and* $\mathcal{E} =$

$$
\begin{array}{lll}
\{ & (\delta \mathsf{in} \gamma_5) \Rightarrow ( & \mathcal{E}_0 \cup \mathcal{E}_1 \cup \mathcal{E}_2 \cup \\
& & \{(\delta \mathsf{in} \{\alpha_4\}) \Rightarrow \{ \quad \alpha'_8 \sqsubseteq \alpha_4, \\
& & \qquad\qquad\qquad\quad \alpha_3 \sqsubseteq \alpha_9, \\
& & \qquad\qquad\qquad\quad (\delta \mathsf{in} \{\alpha_8\}) \Rightarrow \{\alpha_2 \sqsubseteq \alpha_8, \alpha_7 \sqsubseteq \alpha_1\}\}, \\
& & (\delta \mathsf{in} \{\beta_4\}) \Rightarrow \{ \quad \beta'_8 \sqsubseteq \beta_4, \\
& & \qquad\qquad\qquad\quad \beta_3 \sqsubseteq \beta_9, \\
& & \qquad\qquad\qquad\quad (\delta \mathsf{in} \{\beta_8\}) \Rightarrow \{\beta_2 \sqsubseteq \beta_8, \beta_7 \sqsubseteq \beta_1\}\} \quad \} \quad ) \quad \},
\end{array}
$$

*where the sets* $\mathcal{E}_0$, $\mathcal{E}_1$, *and* $\mathcal{E}_2$ *are as in Fig. 9.10.*

*The set* $\mathbf{faithful}(\Theta, \gamma_5)$ *is*

$$
\{ \quad (\delta \mathsf{in} \{\gamma_1\}) \Rightarrow \{\gamma_1 \equiv \delta\}, \quad (\delta \mathsf{in} \{\gamma_2\}) \Rightarrow \{\gamma_2 \equiv \delta\}, \quad \gamma_5 \equiv \delta \quad \} \ ,
$$

*and* $\mathcal{E} \cup \mathbf{faithful}(\Theta, \gamma_5) \rightsquigarrow \mathcal{I}$, *where*

$$
\begin{array}{rcl}
\mathcal{I} & = & \{\alpha_2, \alpha'_2, \alpha_4, \alpha'_4, \alpha_6, \alpha_8, \alpha'_8, \\
& & \beta_1, \beta'_1, \beta_2, \beta'_2, \beta_3, \beta'_3, \beta_4, \beta'_4, \beta_5, \beta_6, \beta_7, \beta'_7, \beta_8, \beta'_8, \beta_9, \beta'_9, \\
& & \gamma_2, \gamma'_2, \gamma_3, \gamma_4, \gamma_5 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \} \ .
\end{array}
$$

*Let* $\mathbf{i}$ *be defined by:* $\mathbf{i}(\alpha) = \delta$ *if* $\alpha \in \mathcal{I}$ *and* $\mathbf{i}(\alpha) = \top$ *otherwise. Then* $\mathbf{i}(\Theta); \emptyset \vdash^{[\mathbf{0}]} \mathbf{i}(M'^{\theta})$ *is the faithful* $\vdash^{[\mathbf{0}]}$*-typing that shows all the dead code that can be detected by using the d-2-type assignment system* $\vdash_{\mathrm{r2cd}}$. *In particular* $\mathbf{i}(\Theta) \vdash_{\mathrm{r2cd}} \mathbf{i}(M'^{\theta})$ *is the faithful* $\vdash_{\mathrm{r2cd}}$*-typing of Example 9.12.* □

$$
\begin{array}{l}
\mathcal{E}_0 = \{(\delta \text{ in } \gamma_5) \Rightarrow \\
\quad \{ \quad (\delta \text{ in } \{\gamma_5\}) \Rightarrow \{ \quad \gamma_3 \sqsubseteq \delta, \\
\qquad\qquad\qquad\qquad\quad \gamma_4 \sqsubseteq \delta \ \}, \\
\qquad (\delta \text{ in } \{\alpha'_4\}) \Rightarrow \{ \quad \gamma'_1 \sqsubseteq \alpha'_3, \\
\qquad\qquad\qquad\qquad\quad (\delta \text{ in } \{\alpha'_4\}) \Rightarrow \{ \quad (\delta \text{ in } \{\alpha'_4\}) \Rightarrow \{ \quad \alpha_4 \sqsubseteq \alpha'_4, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \alpha'_3 \sqsubseteq \alpha_3, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad (\delta \text{ in } \{\alpha_2\}) \Rightarrow \{ \quad \alpha_2 \sqsubseteq \alpha'_2, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \alpha'_1 \sqsubseteq \alpha_1\}\} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad (\delta \text{ in } \{\alpha_2\}) \Rightarrow \{ \quad \alpha_6 \sqsubseteq \alpha'_2, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \alpha'_1 \sqsubseteq \alpha_5\}\}\} \\
\qquad (\delta \text{ in } \{\beta'_4\}) \Rightarrow \{ \quad \gamma'_2 \sqsubseteq \beta'_3, \\
\qquad\qquad\qquad\qquad\quad (\delta \text{ in } \{\beta'_4\}) \Rightarrow \{ \quad (\delta \text{ in } \{\beta'_4\}) \Rightarrow \{ \quad \beta_4 \sqsubseteq \beta'_4, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \beta'_3 \sqsubseteq \beta_3, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad (\delta \text{ in } \{\beta_2\}) \Rightarrow \{ \quad \beta_2 \sqsubseteq \beta'_2, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \beta'_1 \sqsubseteq \beta_1\}\} \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \beta_5 \sqsubseteq \beta_6, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad (\delta \text{ in } \{\beta_2\}) \Rightarrow \{ \quad \beta_6 \sqsubseteq \beta'_2, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \beta'_1 \sqsubseteq \beta_5\}\}\} \\
\qquad \alpha'_4 \sqsubseteq \gamma_3, \\
\qquad \beta'_4 \sqsubseteq \gamma_4 \\
\quad \} \quad \}
\end{array}
$$

$$\mathcal{E}_1 \quad = \quad \{ \quad (\delta \text{ in } \alpha_{8'}) \Rightarrow (\mathbf{cs}_{\leq_1}(\alpha_7 \to \alpha_8,\ \alpha'_7 \to \alpha'_8) \cup \{\alpha'_9 \sqsubseteq \alpha'_7,\ \alpha_9 \sqsubseteq \alpha'_9\}) \quad \}$$

$$\mathcal{E}_2 \quad = \quad \{ \quad (\delta \text{ in } \beta_{8'}) \Rightarrow (\mathbf{cs}_{\leq_1}(\beta_7 \to \beta_8,\ \beta'_7 \to \beta'_8) \cup \{\beta'_9 \sqsubseteq \beta'_7,\ \beta_9 \sqsubseteq \beta'_9\}) \quad \}$$

Figure 9.10: Sets $\mathcal{E}_0$, $\mathcal{E}_1$, and $\mathcal{E}_2$

## 9.4 Final remarks

The algorithm $\mathcal{W}$ is presented in this form to make it as close to the $\vdash^{[\mathbf{0}]}$-type assignment system as possible. Indeed it generates some constraints that can be avoided in a real implementation. Moreover, an efficient implementation of the algorithm should avoid the use of $\{\!|\}$-sequences by recording just the constructor variables containing the relevant information w.r.t. the dead code elimination. In fact (see Section 7.7.4) for every d-2-pattern $\eta$ associated to a subterm, it suffices to keep just the constructor variables in $\mathbf{tail}(\eta)$. This can be done by decorating the terms with sets of constructor variables. For instance, the decorated term $M'^{\theta}$ of Example 9.25 could be replaced by the following:

$$
\begin{array}{l}
((\lambda f^{\{\alpha_4,\beta_4\}}.\\
\quad(((f^{\{\alpha'_4\}}(\lambda x^{\{\alpha_5\}}.3^{\{\alpha_6\}})^{\{\alpha_6\}})^{\{\alpha'_4\}}u^{\{\gamma'_1\}})^{\{\alpha'_4\}}\\
\qquad\quad +^{\{\gamma_5\}}((f^{\{\beta'_4\}}(\lambda y^{\{\beta_5\}}.y^{\{\beta_6\}})^{\{\beta_6\}})^{\{\beta'_4\}}v^{\{\gamma'_2\}})^{\{\beta'_4\}})^{\{\gamma_5\}})^{\{\gamma_5\}}\\
\qquad\qquad\qquad\quad ((\lambda h^{\{\alpha_8,\beta_8\}}.(\lambda z^{\{\alpha_9,\beta_9\}}.(h^{\{\alpha'_8,\beta'_8\}}z^{\{\alpha'_9,\beta'_9\}})^{\{\alpha'_8,\beta'_8\}})^{\{\alpha'_8,\beta'_8\}}))^{\{\gamma_5\}}\ .
\end{array}
$$

In this way it should be possible to embed the inference algorithm $\mathcal{W}$ and the simplification mapping $\mathbf{O}^{\wedge 2}$ in the dead-code detection and elimination procedure described in Section 7.7.

# Part IV

# Related topics and further work

# Chapter 10

# Related topics and further work

In this chapter we give a brief account of some topics related to what we have presented in the thesis together with some suggestions for further work.

We first discuss some possible extensions of our work to include:

1. richer languages of properties (Section 10.1),

2. others program analyses (Section 10.2), and

3. other programming language features (Section 10.3).

Moreover we mention the important issue of relating our non-standard type inference framework with abstract interpretation (Section 10.4), and conclude with a discussion about how to use the results of the static analyses presented in the thesis (Section 10.5).

## 10.1 Disjunctive properties

In [Jensen, 1992] Chapter 4 a *disjunctive and conjunctive strictness logic* is presented. This analysis can be easily described in our framework. In fact:

- the set of properties considered is $\mathbf{L}^{\mathrm{cds}} = \cup_{\rho \in \mathbf{T}} \mathbf{L}^{\mathrm{cds}}(\rho)$, where the sets $\mathbf{L}^{\mathrm{cds}}(\rho)$ are defined according to clauses Fig. 10.1, and

- a semantics for these *conjunctive and disjunctive strictness types* is specified by the clauses of Definition 5.2 and by the following clause for the interpretation of the disjunction operator
$$[\![\nu_1 \vee \nu_2]\!]^{\mathcal{M}} = [\![\nu_1]\!]^{\mathcal{M}} \cup [\![\nu_2]\!]^{\mathcal{M}}.$$

The main problem about this conjunctive and disjunctive strictness analysis is that it does not have realistic checking or inference algorithms. So a possible direction for further research could be the investigation of tractable restrictions of this powerful system.

$$(a)\ \frac{a \in \{\bot, \top\} \quad \rho \in \mathbf{T}}{a^{\rho} \in \mathbf{L}^{\mathrm{cds}}(\rho)} \qquad (\to)\ \frac{\nu_1 \in \mathbf{L}^{\mathrm{cds}}(\rho_1) \quad \nu_2 \in \mathbf{L}^{\mathrm{cds}}(\rho_2)}{\nu_1 \to \nu_2 \in \mathbf{L}^{\mathrm{cds}}(\rho_1 \to \rho_2)}$$

$$(\wedge)\ \frac{\nu_1 \in \mathbf{L}^{\mathrm{cds}}(\rho) \quad \nu_2 \in \mathbf{L}^{\mathrm{cds}}(\rho)}{\nu_1 \wedge \nu_2 \in \mathbf{L}^{\mathrm{cds}}(\rho)} \qquad (\vee)\ \frac{\nu_1 \in \mathbf{L}^{\mathrm{cds}}(\rho) \quad \nu_2 \in \mathbf{L}^{\mathrm{cds}}(\rho)}{\nu_1 \vee \nu_2 \in \mathbf{L}^{\mathrm{cds}}(\rho)}$$

Figure 10.1: Conjunctive and disjunctive strictness types

## 10.2 Binding-time analysis

In [Jensen, 1992] Section 3.4 it is showed how the conjunctive strictness logic can be modified to give a binding-time analysis corresponding to the abstract interpretation for binding-time analysis presented in [Hunt and Sands, 1991]. Binding-time analysis, given a term and a description of the parameters that will be known at partial evaluation time, aims to determine which parts of the programs depends solely on this known parts (and therefore also known at partial evaluation time).

As noted in [Hunt and Sands, 1991] there is a connection between dead-code and binding-time analyses. Take for instance the dead-code analyses developed in Part III of the thesis:

- a $\delta$-d-type, $\phi$ (interpreted as the diagonal p.e.r. over $\mathbf{I}(\epsilon(\phi))$) characterizes terms whose values have a precise identity, so it can be used to express the fact that the value of an expression can be computed at compile-time,
- while a $\top$-d-type $\phi$ (interpreted as the trivial p.e.r. over $\mathbf{I}(\epsilon(\phi))$) characterizes terms whose values do not have a precise identity, so it can be used to express the fact that the value of an expression must be computed at run-time.

In the binding-time literature, expressions to which it is assigned a property of the first kind are called *static*, while expressions having a property of the second kind are called *dynamic*.

So we may think to use the dead-code type assignment systems in Part III of the thesis for binding-time analysis. In this perspective, given a set of constraints describing all the possible decorations of a term, we have to look for a decoration of the term that respect the static and dynamic information of the parameters and that maximizes the number of $\delta$'s in the term. In fact a subterm decorated by a $\delta$-d-type is such that its value does not depend on the values of the variables with dynamic type. So it is a natural candidate for evaluation when all static values are known. Consider the following example: take the term

$$M = \lambda g^{\mathsf{nat}\to(\mathsf{nat}\to\mathsf{nat})\to\mathsf{nat}}.\lambda x^{\mathsf{nat}}.(\lambda f^{\mathsf{nat}\to\mathsf{nat}}.g(fx)f)(\lambda y^{\mathsf{nat}}.y)$$

of type $\rho = (\mathsf{nat} \to (\mathsf{nat} \to \mathsf{nat}) \to \mathsf{nat}) \to \mathsf{nat} \to \mathsf{nat}$. If we assume that that $x$ is known and $g$ is not, we have that $M$ must be assigned a property of the form

$$\theta = (\alpha_1{}^{\mathsf{nat}} \to (\alpha_2{}^{\mathsf{nat}} \to \alpha_3{}^{\mathsf{nat}}) \to \top^{\mathsf{nat}}) \to \delta^{\mathsf{nat}} \to \alpha_4{}^{\mathsf{nat}},$$

where the $\alpha_i$ ($i \in \{1, \ldots, 4\}$) may be either $\top$ or $\delta$. A decoration of the term consistent with the requirements above and maximizing the number of $\delta$'s is the following (writing, for short, $\delta$ and $\top$ instead of $\delta^{\mathsf{nat}}$ and $\top^{\mathsf{nat}}$):

$$(\lambda g^{\delta\to(\delta\to\delta)\to\top}.(\lambda x^{\delta}.((\lambda f^{\delta\to\delta}.((g^{\delta\to(\delta\to\delta)\to\top}(f^{\delta\to\delta}x^{\delta})^{\delta})^{(\delta\to\delta)\to\top}f^{\delta\to\delta})^{\top})^{(\delta\to\delta)\to\top}(\lambda y^{\delta}.y^{\delta})^{\delta\to\delta})^{\top})^{\delta\to\top})^{\phi},$$

where

$$\phi = (\delta^{\mathsf{nat}} \to (\delta^{\mathsf{nat}} \to \delta^{\mathsf{nat}}) \to \top^{\mathsf{nat}}) \to \delta^{\mathsf{nat}} \to \top^{\mathsf{nat}}.$$

The previous decorations of $M$ shows that, given a static expression (e.g. a natural number) $N$ and a dynamic one $G$, the term $(M\,G\,N)$ is dynamic, since it is decorated by an $\top$-d-type (according to the fact that its value depends on that of the unknown value of $G$). However,

- the substitution of the formal parameter $x$ with the actual parameter $N$,
- the substitution of the formal parameter $f$ with the actual parameter $\lambda y.y$, and
- the evaluation of the subterm $(fx)[x := N,\ f := \lambda\, y.y]$,

which involve only parts of the term decorated by $\delta$-types, can be performed at partial evaluation time.

## 10.3 Programming language extensions

The language PCFP lacks important features of real functional programming languages. In particular

- it has no support for *recursive datatypes*,
- its simple type system does not incorporate any form of *polymorphism*,
- it has no support for *Input/Output*, and
- it does not include an *exception* mechanism.

Other features not present in PCFP are

- a notion of state, which is present in *functional languages with state* (like ML and Caml), and
- concurrent primitives, which are characteristic of *concurrent functional languages* (like Concurrent ML [Reppy, 1992]).

To apply the techniques developed through the thesis to a given programming language we need an operational theory of the language. Operational theories for functional programming languages including some of the features listed above have been investigated and are actually investigated by a number of authors (see for instance [Gordon, 1994; Pitts, 1997]).

In the rest of this section we briefly discuss static analyses of functional languages with recursive datatypes and polymorphism.

### 10.3.1 Recursive datatypes

A recursive datatype definition can be described by an equality like

$$\mathrm{id} = \tau[\mathrm{id}],$$

where "id" is the name of the type being defined and $\tau[\mathrm{id}]$ is an arbitrary type expression involving the name "id".

For a simple example (using ML syntax) consider a datatype describing integer expressions:

$$\begin{array}{rcl} \mathsf{datatype}\ \mathrm{exp} & = & \mathrm{Num}\ \mathsf{of}\ \mathsf{nat} \mid \mathrm{Uop}\ \mathsf{of}\ \mathrm{exp} \to \mathrm{esp} \mid \\ & & \mathrm{Bop}\ \mathsf{of}\ \mathrm{exp} \times \mathrm{exp} \to \mathrm{exp}, \end{array}$$

where the symbol "|" is a *labeled sum* type constructor, specifying that the canonical elements of the type being defined are build in different ways (each of them is identified by a label), and the identifiers "Num", "Uop", "Bop" (called *constructors of the datatype* "exp") are the labels of the labeled sum.

*Algebraic datatypes* (also known as *sum-of-products* or *polynomial* datatypes) are the subclass of recursive datatypes characterized by the fact that the right part of the datatype definition consists of a labeled sum of products of

- simple types,
- previously defined datatypes, and
- occurrences of the name of the type being defined.

This class includes lists, $n$-ary trees, and so on. For a simple example take the type of lists of functions from natural to booleans:

$$\mathsf{datatype}\ \mathrm{flist} = \mathrm{Nil} \mid \mathrm{Cons}\ \mathsf{of}\ (\mathsf{nat} \to \mathsf{bool}) \times \mathrm{flist}.$$

Note that "exp" is not algebraic, since (in the last two components of the labeled sum, "Uop" and "Bop") the identifier name of the type being defined occurs in the scope of an arrow type constructor.

#### Strictness and totality analyses

Both [Jensen, 1992] and [Benton, 1992a] consider the problem of defining a strictness analysis for a functional language with algebraic datatypes. Moreover [Hankin and Le Métayer, 1994c; Hankin and Le Métayer, 1995] present a strictness logic (based on Jensen's work) which corresponds to Wadler's 4-point domain [Wadler, 1987] for lists of ground type and then generalize it to domains of any depth. The analysis of [Hankin and Le Métayer, 1994c; Hankin and Le Métayer, 1995] can be easily formalized in our framework. Interesting direction for further work are

- formalizing the general case of strictness analyses of algebraic datatypes in our framework, and

- considering totality analysis of algebraic datatypes.

#### Dead-code analysis

[Boerio, 1995] extends the dead-code analysis of [Berardi, 1993b; Berardi, 1996] (which, as explained in Section 7.9, is essentially the analysis presented Section 7.3 of this thesis) to a language with algebraic datatypes. This analysis can be easily formalized in our framework. We are planning to extend the dead-code detection and elimination procedure of Section 7.7 to a version of PCFP$^{\mathcal{T}}$ supporting algebraic datatypes.

### 10.3.2 Polymorphism

The type systems of most functional languages are extension of the Hindley/Milner type system [Hindley, 1969; Milner, 1978; Damas and Milner, 1982]. Such systems usually support two kinds of polymorphism

- *ad hoc polymorphism* (or *overloading*), in which a same expression denotes different operations at different types (for instance, in many programming languages the symbol "+" is used to denote both integer and real addition), and

- *parametric polymorphism*, in which a same expression behaves in a "uniform way" at different types. For instance the function

  $$\text{swap} = \lambda p.\langle \mathsf{proj}_2 p, \mathsf{proj}_1 p\rangle,$$

  works in the same way on pair of every type, and this uniformity can be expressed by a type like

  $$\sigma = \forall\alpha.\forall\beta.(\alpha \times \beta) \to (\beta \times \alpha).$$

The more interesting notion is parametric polymorphism, and in the rest of this section we focus on it. Results making explicit the meaning of the phrase "uniform way" above (i.e. showing in which sense a polymorphic term works in the same way at different types) are called *parametricity results*.

[Wadler, 1989] observes that parametricity provides a source of theorems useful for program analysis and transformation. Given a polymorphic type it is possible to derive a theorem which is satisfied by every term of that type. For example, from the polymorphic type of the functions mapping every list of a given type to a list of the same type:

$$\tau = \forall\alpha.\text{list}(\alpha) \to \text{list}(\alpha),$$

it is possible to derive the following theorem: for all functions $F$ of type $\tau$, for all types $\rho_1$ and $\rho_2$, and for all total functions $G$ of type $\rho_1 \to \rho_2$, we have

$$(\mathrm{map}\, G) \circ F_{\rho_1} = F_{\rho_2} \circ (\mathrm{map}\, G),$$

where the function "map $G$" applies $G$ elementwise to a list of $\rho_1$ returning a list of $\rho_2$, and $F_\rho : \mathrm{list}(\rho) \to \mathrm{list}(\rho)$ is the instance of $F$ at the type $\rho$. The intuition behind this result is that $F$ can only rearrange the elements of the list in input independently on the values (and the type) of its elements, since (having type $\tau$) it must work on lists of $\rho$ for every type $\rho$.

The system considered in [Wadler, 1989] is the Girard/Reynolds system [Girard, 1972; Girard, 1986; Reynolds, 1974; Reynolds, 1983] (also known as polymorphic $\lambda$-calculus, second order $\lambda$-calculus, and system F), see [Benton, 1992a] Chapter 5 for a formulation of Wadler's results for a language with algebraic datatypes supporting an explicitly typed variant of Hindley/Milner polymorphism.

[Abramsky, 1986] (see also [Abramsky and Jensen, 1991]) introduce *polymorphic invariance* to exploit the strong regularities inherent in the parametric polymorphism for deducing properties valid for all instances from properties valid from single instances. There are two kinds of polymorphic invariance.

- Polymorphic invariance of a property, which means that one instance of a function has a given property if and only if all instances of the function have that property.

- Polymorphic invariance of an analysis, which means that a particular analysis technique can find a given property at one instance of a function if and only if it can find that property at all instances of the function.

[Abramsky, 1986; Abramsky and Jensen, 1991] focus on the second notion and show polymorphic invariance for a strictness analyses by abstract interpretation. [Benton, 1992b] (see also [Benton, 1992a] Chapter 5) improves this kind of polymorphic invariance by considering polymorphic invariance results at subsets of all instances, starting from the fact that certain properties of polymorphic functions (like *"maps strict functions to strict functions"*) are polymorphic invariant only *"at all instances for which that properties make sense"*.

[Boerio, 1995] extends the dead-code analysis of [Berardi, 1993b; Berardi, 1996] (see Section 7.9 of this thesis) to the Girard/Reynolds system.

## 10.4 Relating our framework with abstract interpretation

In this thesis we have not addressed the important issue of relating our non-standard type inference framework with the abstract interpretation approach[1].

The relations between type inference and abstract interpretation have been investigated, for instance, in [Jensen, 1992; Benton, 1992a; Coppo and Ferrari, 1993; Cousot, 1997].

We believe that a careful study of the relations with abstract interpretation could give a better understanding of both limits and potentialities of our framework.

## 10.5 How to use the results of the analyses

As anticipated in Section 3.8

- in Part II of the thesis we have not considered optimizing program transformations based on the results of the various strictness and totality analyses presented,
- while, in Part III we have introduced, for each dead-code analysis presented, a program transformation based on the results of the analysis.

In general to specify how the results of a program analysis can be used as the basis for optimizing transformation may be a quite complex task. This is indeed a large area of further work. For a short survey (and a proposal) about how to use the results of strictness analysis see, for instance, [Benton, 1996]. Quoting from [Benton, 1996] Section 1: *"There has, however, been surprisingly little serious work on just* how *the results of strictness analysis can be used as the basis for optimizing transformations. This is probably because it turns out to be rather more difficult to express and justify these optimizations than one might at first imagine. Roughly speaking, it seems we have to decide*

1. *What optimizations we wish to perform.*
2. *How to express these optimizations in some formal framework.*
3. *Exactly what information has to be gathered to enable each optimization.*
4. *How to prove the correctness of the optimizations."*

For dead-code analysis we have considered all the four points above. In particular, in Section 7.7, we have described an incremental dead-code detection and elimination procedure.

[1] Although, for the conjunctive strictness analysis of Section 5.2.3, we can rely (see Section 5.2.4) on the results of [Jensen, 1992] and [Benton, 1992a].

# Bibliography


Abelson, H. and Susman, G. J. (1985). *Structure and Interpretation of Computer Programs*. MIT Press.

Abramsky, S. (1986). Strictness analysis and polymorphic invariance. In *Programs as Data Objects*, LNCS 217, pages 1–32. Springer–Verlag.

Abramsky, S. (1990a). Abstract interpretation, logical relations and Kan extensions. *Journal of Logic and Computation*, 1(1):5–39.

Abramsky, S. (1990b). The Lazy $\lambda$-calculus. In Turner, D. A., editor, *Reseach Topics in Functional Programming*, pages 1–77. Addison Wesley.

Abramsky, S. and Jensen, T. P. (1991). A relational approach to strictness analysis of higher order polymorphic functions. In *POPL'91*. ACM.

Barendregt, H. P. (1984). *The Lambda Calculus, its Syntax and Semantics*. North-Holland, Amsterdam.

Barendregt, H. P. (1992). Lambda calculi with types. In S. Abramsky *et al*, editor, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Claredon Press, UK.

Barendregt, H. P., Coppo, M., and Dezani-Ciancaglini, M. (1983). A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic*, 48:931–940.

Barendsen, E. and Smetsers, S. (1995a). A Derivation System for Uniqueness Typing . In *SEGRAGRA'95*. Elsevier, Electronic Notes in Theoretical Computer Science.

Barendsen, E. and Smetsers, S. (1995b). Uniqueness Type Inference. In *PLILP'95*, LNCS 982, pages 189–207. Springer–Verlag.

Barras, B., Boutin, S., Cornes, C., Courant, J., Filliâtre, J.-C., Herbelin, H., Huet, G., Manoury, P., Muñoz, C., Murthy, C., Parent, C., Paulin-Mohring, C., Saïbi, A., and Werner, B. (1996). *The Coq Proof Assistant Reference Manual Version 6.1*. INRIA-Rocquencourt-CNRS-ENS Lyon.

Benton, P. N. (1992a). *Strictness Analysis of Lazy Functional Programs*. PhD thesis, University of Cambridge, Pembroke College.

Benton, P. N. (1992b). Strictness Logic and Polymorphic Invariance. In *LFCS'92*, LNCS 620, pages 33–44. Springer–Verlag.

Benton, P. N. (1996). A Unified Approach to Strictness Analyses and Optimizing Transformations. Technical Report 388, University of Cambridge Computer Laboratory.

Berardi, S. (1993a). Extensional Equality for Simply Typed Lambda Terms. Dipartimento di Informatica, Università di Torino, Internal report.

Berardi, S. (1993b). Pruning Simply Typed Lambda Terms. Dipartimento di Informatica, Università di Torino, Internal report.

Berardi, S. (1996). Pruning Simply Typed Lambda Terms. *Journal of Logic and Computation*, 6(5):663–681.

Berardi, S. and Boerio, L. (1995). Using Subtyping in Program Optimization. In *TLCA'95*, LNCS 902. Springer–Verlag.

Boerio, L. (1995). *Optimizing Programs Extracted from Proofs*. PhD thesis, Università di Torino.

Burn, G. L., Hankin, C., and Abramsky, S. (1986). Strictness Analysis for Higher-Order Functions. *Science of Computer Programming*, 7:249–278.

Coppo, M., Damiani, F., and Giannini, P. (1996). Refinement Types for Program Analysis. In *SAS'96*, LNCS 1145, pages 143–158. Springer–Verlag.

Coppo, M., Damiani, F., and Giannini, P. (1997). On Strictness and Totality. In *TACS'97*, LNCS 1281, pages 138–164. Springer–Verlag.

Coppo, M. and Dezani-Ciancaglini, M. (1980). An extension of basic functional theory for lambda-calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693.

Coppo, M., Dezani-Ciancaglini, M., and Venneri, B. (1981). Functional Characters of Solvable Terms. *Zeith. Math. Logik Und Grund. Math.*, 27:45–58.

Coppo, M. and Ferrari, A. (1993). Type inference, abstract interpretation and strictness analysis. *Theoretical Computer Science*, 121:113–145.

Cousot, P. (1997). Types as Abstract Interpretations. Invited paper. In *POPL'97*, pages 316–331. ACM.

Cousot, P. and Cousot, R. (1977). Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction of Approximations of Fixedpoints . In *POPL'77*, pages 238–252. ACM.

Cousot, P. and Cousot, R. (1979). Systematic design of program analysis frameworks. In *POPL'79*. ACM.

Cousot, P. and Cousot, R. (1994). Higher-Order Abstract Interpretation (and Application to Comportment Analysis Generalizing Strictness, Termination, Projection and PER Analysis of Functional Languages). Invited paper. In *ICCL'94*, pages 95–112. IEEE.

Damas, L. M. M. and Milner, R. (1982). Principal type schemas for functional programs. In *POPL'82*, pages 207–212. ACM.

Damiani, F. and Giannini, P. (1997). An Inference Algorithm for Strictness. In *TLCA'97*, LNCS 1210, pages 129–146. Springer–Verlag.

Damiani, F. and Prost, F. (1997). Detecting and Removing Dead Code using Rank 2 Intersection. In *International Workshop TYPES'96, Selected Papers*, LNCS, to appear. Springer–Verlag.

de Mast, P., Jansen, J.-M., Bruin, D., Fokker, J., Koopman, P., Smetsers, S., van Eekelen, M., and Plasmeijer, R. (1997). Functional Programming in Clean - draft. Available from `htpp://www.cs.kun.nl/~clean`.

de Roever, W. P. (1978). On backtracking and greater fixed points. In *Formal Description of Programming Concepts*, LNCS 104, pages 621–639. North Holland.

Dussart, D., Henglein, F., and Mossin, C. (1995). Polymorphic Recursion and Subtype Qualifications: Polymorphic Binding-Time Analysis in Polynomial Time. In *SAS'95*, LNCS 983, pages 118–135. Springer–Verlag.

Dybjer, P. and Sander, H. P. (1989). A functional programming approach to the specification and verification of cuncurrent systems. *Formal Aspects of Computing*, 1:303–319.

Egidi, L., Honsell, F., and Ronchi della Rocca, S. (1992). Operational, denotational and logical descriptions: a case study. *Fundamenta Informaticae*, 16(2):149–169.

Girard, J. Y. (1972). *Interpretation fonctionelle et elimination des coupures dans l'aritmetique d'ordre superieur*. PhD thesis, Université Paris VII.

Girard, J.-Y. (1986). The system $f$ of variable types, fifteen years later. *Theoretical Computer Science*, 45(2):159–192.

Gordon, A. D. (1994). *Functional Programming and Input/Output*. Distinguished Dissertations in Computer Science. Cambridge University Press.

Hankin, C. and Le Métayer, D. (1994a). A Type-Based Framework for Program Analysis. In *SAS'94*, LNCS 864, pages 380–394. Springer–Verlag.

Hankin, C. and Le Métayer, D. (1994b). Deriving algorithms for type inference systems: Applications to strictness analysis. In *POPL'94*, pages 202–212. ACM.

Hankin, C. and Le Métayer, D. (1994c). Lazy Types Inference for the Strictness Analysis of List. In *ESOP'94*, LNCS 788, pages 257–271. Springer–Verlag.

Hankin, C. and Le Métayer, D. (1995). Lazy type inference and program analysis. *Science of Computer Programming*, 25:219–249.

Hindley, R. (1969). The principal types schemes for an object in combinatory logic. *Transactions of American Mathematical Society*, 146:29–60.

Hindley, R. (1997). *Basic Simple Type Theory*. Number 42 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, London.

Howe, D. J. (1989). Equality in lazy computation systems. In *LICS'89*, pages 198–203. IEEE.

Howe, D. J. (1996). Proving congruence of bisimulations in functional programming languages. *Information and Computation*, 2(124):103–112.

Hudak, P. (1989). Conception, Evolution, and Application of Functional Programming Languages. *ACM Computing Surveys*, 21(3):359–411.

Hughes, J. (1989). Why Functional Programming Matters. *Computer Journal*, 32(2):98–107.

Hunt, L. and Sands, D. (1991). Binding Time Analysis: A New PERspective. In *Partial Evaluation and Semantics-based Program Manipulation*. ACM.

Jensen, T. P. (1992). *Abstract Interpretation in Logical Form*. PhD thesis, University of London, Imperial College.

Jensen, T. P. (1995). Conjunctive Type Systems and Abstract Interpretation of Higher-order Functional Programs. *Journal of Logic and Computation*, 5(4):397–421.

Kahn, G. (1988). Natural semantics. In Fuchi, K. and Nivat, M., editors, *Programming Of Future Generation Computer*. Elsevier Sciences B.V. (North-Holland).

Kuo, T. M. and Mishra, P. (1989). Strictness analysis: a new perspective based on type inference. In *Functional Programming Languages and Computer Architecture*, pages 260–272. ACM.

Leroy, X. (1997). The Caml Light System, documentation and users's guide. Available from `htpp://paulliac.inria.fr/caml`.

Luo, Z. and Pollack, R. (1992). Lego proof development system : User's manual. Technical Report ECS-LFCS-92-211, University of Edinburgh.

Milner, R. (1978). A Theory of Type Polymorphism in Programming. *Journal of Computer and System Science*, 17:348–375.

Milner, R. (1989). *Communication and Concurrency*. Prentice Hall.

Milner, R., Tofte, M., and Harper, R. (1990). *The Definition of Standard ML*. MIT press.

Milner, R., Tofte, M., and Harper, R. (1991). *Commentarey on Standard ML*. MIT press.

Mossin, C. (1997). *Flow analysis of Typed Higher-Order Programs*. PhD thesis, DIKU, University of Copenhagen. Revised version.

Mycroft, A. (1980). The theory and practice of transforming call-by-need into call-by-value. In *International Symposium on Programming*, LNCS 83, pages 269–281. Springer–Verlag.

Mycroft, A. (1981). *Abstract Interpretation and Optimizing Transforming for Applicative programs*. PhD thesis, University of Edinburgh, Scotland.

Mycroft, A. and Solberg, K. L. (1995). Uniform PERs and comportment analysis. In *PLILP'95*, LNCS 982, pages 169–187. Springer–Verlag.

Nielson, F. (1988). Strictness Analysis and Denotational Abstract Interpretation. *Information and Computation*, 76(1):29–92.

Nielson, F. (1989). Two-level semantics and abstract interpretation. *Theoretical Computer Science*, 69(2):117–242.

Park, D. (1981). Concurrency and automata in infinite sequences. In *GI-Conference on Theoretical Computer Science*, LNCS 104, pages 167–183. Springer–Verlag.

Paulin-Mohring, C. (1989a). Extracting $F_\omega$'s Programs from Proofs in the Calculus of Constructions. In *POPL'89*. ACM.

Paulin-Mohring, C. (1989b). *Extraction de Programme dans le Calcul des Constructions*. PhD thesis, Université Paris VII.

Paulson, L. C. (1996). *ML for the working programmer*. Cambridge University Press.

Peterson, J., Hammond, K., Agustsson, L., Boutel, B., Burton, W., Fasel, J., Gordon, A. D., Hughes, J., Hudak, P., Johnsson, T., Jones, M., Meijer, E., Pejton Jones, S., Reis, A., and Wadler, P. (1997). Report on the Programming Language Haskell (version 1.4) - draft. Available from `htpp://www-i2.informatik.rwth-aaken.de/Forshung/FP/Haskell`.

Pfenning, F. (1996). The practice of logical frameworks. In Kirchner, H., editor, *Proceedings of the Colloquium on Trees in Algebra and Programming*, pages 119–134, Linköping, Sweden. Springer-Verlag LNCS 1059. Invited talk.

Pierce, B. C. (1991). *Programming with Intersection Types and Bounded Polymorphism*. PhD thesis, Carnegie Mellon University. Available as School of Computer Science technical report CMU-CS-91-205.

Pitts, A. M. (1997). Operationally-based theories of program equivalence. In Pitts, A. M. and Dybjer, P., editors, *Semantics and Logics of Computation*, pages 241–298. Cambridge University Press.

Plasmeijer, R. and van Eekelen, M. (1997). The Cuncurrent Clean Language Report (version 1.2) - draft. Available from `htpp://www.cs.kun.nl/~clean`.

Plotkin, G. D. (1977). LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223–255.

Plotkin, G. D. (1981a). A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University.

Plotkin, G. D. (1981b). Post-graduate lecture notes in advanced domain theory (incorporating the "Pisa Notes"). Technical report, Dept. of Computer Science, Univ. of Edinburgh.

Prost, F. (1995). Marking techniques for extraction. Technical Report RR95-47, Ecole Normale Supérieure de Lyon, Lyon.

Prost, F. (1997). Personal communication. September 1997.

Reppy, J. H. (1992). Cuncurrent ML: Design, appllication and semantics. In *Functional programming, Concurrency, Simulation and Automated Reasoning*, LNCS 693, pages 165–198. Springer–Verlag.

Reynolds, J. C. (1974). Towards a Theory of Type Structure. In *Colloque sur la Programmation*, LNCS 19. Springer–Verlag.

Reynolds, J. C. (1983). Types, abstraction and parametric polymorphism. In *Information Processing 83*, pages 513–523, Amsterdam. Elsevier Science Publishers B. V. (North-Holland).

Scott, D. S. (1982). Domains for denotational semantics. In *Automata, Languages and Programming*, LNCS 140. Springer–Verlag.

Solberg, K. L. (1995a). *Annotated Type Systems for Program Analysis*. PhD thesis, Aarhus University, Denmark. Revised version.

Solberg, K. L. (1995b). Strictness and Totality Analysis with Conjunction. In *TAPSOFT'95*, LNCS 915, pages 501–515. Springer–Verlag.

Solberg, K. L. (1996). Personal communication. December 1996.

Solberg, K. L., Nielson, H. R., and Nielson, F. Strictness and Totality Analysis. *Science of Computer Programming,* to appear.

Solberg, K. L., Nielson, H. R., and Nielson, F. (1994). Strictness and Totality Analysis. In *SAS'94*, LNCS 864, pages 408–422. Springer–Verlag.

Takayama, Y. (1991). Extraction of Redundancy-free Programs from Constructive Natural Deduction Proofs. *Journal of Symbolic Computation*, 12:29–69.

Turner, D. A. (1985). Miranda: a non-strict functional language with polymorphic types. In *Proceedings of the IFIP International Conference on Functional Programming Languages and Computer Architecture*, LNCS 201, pages 1–16. Springer–Verlag.

Ullman, J. D. (1993). *Elements of ML programming.* Prentice Hall.

van Bakel, S. (1992). Complete restrictions of intersection type discipline. *Theoretical Computer Science*, 102:135–163.

van Bakel, S. (1993). *Intersection Type Disciplines in Lambda Calculus and Applicative Term Rewriting Systems.* PhD thesis, Katholieke Universiteit Nijmegen.

Wadler, P. (1987). Strictness analysis on non-flat domains. In Abramsky, S. and Hankin, C. L., editors, *Abstract Interpetation of Declarative Languages.* Ellis Horwood, Chichester, UK.

Wadler, P. (1989). Theorems for free. In *Functional Programming Languages and Computer Architecture*, pages 347–359. ACM.

Wadler, P. and Hughes, J. (1987). Projections for strictness analysis. In *Proceedings of FPCA'87*, LNCS 274, pages 385–407. Springer–Verlag.

Wright, D. A. (1991). A New Technique for Strictness Analysis. In *Proceedings of TAPSOFT'91*, LNCS 494, pages 260–272. Springer–Verlag.

Wright, D. A. (1992). *Reduction Types and Intensionality in the Lambda-Calculus.* PhD thesis, University of Tasmania.