A Finitary Version of the Calculus of Partial Inductive Definitions

SICS Research Report R92:08*)

Lars-Henrik Eriksson
Swedish Institute of Computer Science (SICS)
Box 1263
S-164 28 KISTA, SWEDEN
e-mail: lhe@sics.se

ABSTRACT

The theory of partial inductive definitions is a mathematical formalism which has proved to be useful in a number of different applications. The fundamentals of the theory is shortly described. Partial inductive definitions and their associated calculi are essentially infinitary. To implement them on a computer, they must be given a formal finitary representation. We present such a finitary representation, and prove its soundness. The finitary representation is given in a form with and without variables. Without variables, derivations are unchanging entities. With variables, derivations can contain logical variables that can become bound by a binding environment that is extended as the derivation is constructed. The variant with variables is essentially a generalization of the pure GCLA programming language.

ACKNOWLEDGEMENTS

The author wishes to thank his colleagues at the Swedish Institute of Computer Science for their suggestions and for providing a stimulating research environment. In particular, I want to thank the people of the GCLA project for provoking me to develop many of the results presented here, as well as Philipp Hanschke (of DFKI, Kaiserslautern) who read a draft version of the article and suggested several improvements.

*) Also published in: Lars-Henrik Eriksson, Lars Hallnäs and Peter Schroeder-Heister (eds.), *ELP '91, Proceedings of the Second Workshop on Extensions of Logic Programming held at SICS, Stockholm, Sweden, January 1991,* Lecture Notes in Artificial Intelligence, Springer-Verlag, 1992.

1. INTRODUCTION

The theory of partial inductive definitions is a mathematical formalism developed by Lars Hallnäs [7,11]. It has proved to be useful in a number of different applications, both theoretical and practical. So far, the theory has been used for such diverse things as knowledge representation and reasoning [2,10,15,24], logic programming [4,5,13,14,18], functional programming [3,12], general logic [6,7], program verification [7], recursion theory [8] and typing non-normalizable terms of the lambda calculus [9]. Some of these application areas have been subject to extensive investigations, while in other only exploratory work has been done.

A "partial inductive definition" is similar in form and concept to an ordinary inductive definition [1], but is extended by permitting definitions depending on hypotheses, e.g.

$X \rightarrow Y$ is a true proposition if Y is a true proposition under the assumption that X is one

While any set that can be expressed using a partial inductive definition can also be expressed by an ordinary one (trivially, by listing all members), the expressive power of a partial inductive definition is much greater.

Every partial inductive definition defines a particular sequent calculus used to infer whether an object belongs to the defined set. The exact form of the inference rules depend in a natural way of the particular partial inductive definitions.

A problem with the partial inductive definitions and their associated calculi, as defined by Hallnäs, is that they are purely mathematical objects. In particular, they are in general infinite. To use a mathematical formalism on a computer, or even for a human to deal with it in an efficient way, it must be possible to capture the infinities in a finite representation. In [11] and other original work on the subject there is no mention of how this could be done. Also, the meaning of the concept of "proof" in infinitary deductive systems is unclear.

Hallnäs and Schroeder-Heister [13,14] developed a formal base for logic programming, based on partial inductive definitions. This work was the foundation for the programming language GCLA [4,5], and thus a kind of finitary representation. However, the connection between the system of Hallnäs and Schroeder-Heister and pure partial inductive definitions was never made explicit. Also, their system could not handle partial inductive definitions with infinite conditions. In [7] this author, and in [15] Hanschke, present partial ad-hoc solutions to these problems. The work presented in the present article is a further development and formalisation of the approach from [7].

We will present a finitary representation and its corresponding finitary sequent calculi, show the correspondence with the infinitary partial inductive definition and prove that the finitary calculus is sound. By soundness we mean that any derivation in the finitary calculus faithfully represents some derivation of the infinitary calculus.

With a finitary representation we can, of course, only represent countably many partial inductive definitions and derivations. Since there are uncountably many of both kinds, it will be impossible to represent every definition or derivation. We will not attempt a formal

characterisation of precisely which objects we can represent. Intuitively, we can represent those definitions and derivations which can be generated by uniform replacement of quantified variables by elements of some set.

To permit our finitary system to be used as a suitable base for the construction of proofs in the calculi of partial inductive definitions, we further introduce logical variables in proofs. As in a logic programming language, logical variables can be bound to terms as the proof construction progresses. Indeed, our finitary calculus with variables is an extension of the system of Hallnäs and Schroeder-Heister used as the foundation for the GCLA language. We will give a natural interpretation to the occurrence of variables in derivations, and show that binding a variable is a meaningful operation, given our interpretation.

2. PARTIAL INDUCTIVE DEFINITIONS

The theory of partial inductive definitions is given a short presentation. The intention of this presentation is to provide a reference for the rest of the paper. The motivation and intuition behind partial inductive definitions will not be described, the interested reader is referred to [7] and [11].

The precise definition of partial inductive definitions and the notation used varies considerably between different articles on the subject, although all are essentially equivalent. This is partly due to the evolving of the theory, partly for technical reasons in computer implementations. The concepts and notation used here for the pure theory will be the ones currently used at the University of Gothenburg. The concepts and notation used for the finitary version will be the more computer-oriented ones used at SICS.

Assume that there is a universe, U, of objects which we call **atoms**. We define the set of **conditions** (over U), denoted Cond(U), to be all of the following:

$\wedge$	(The universal condition)
$\bot$	(The empty condition)
a	for every atom $a \in U$
$(C_i)_{i \in I}$	where every C_i is a condition (over U)
$C \rightarrow C'$	where C and C′ are conditions (over U)

The notation $(C_i)_{i \in I}$ should be understood as the set of all C_i such that $i \in I$. Note that the $(C_i)_{i \in I}$ -conditions can be infinite collections, if the index set I is infinite.

If a is an atom in U and A is a condition (over U), then a **(definitional) clause** (over U) has the form

$$a = A$$

A **partial inductive definition** (over U) is a set of clauses (over U). Definitions will be denoted by D (possibly indexed or annotated). Note that a partial inductive definition can comprise an infinite number of clauses.

The **domain** of a definition, denoted $\mathcal{D}om(D)$, is defined as $\mathcal{D}om(D) = \{a \mid (a=A) \in D\}$.

The **definiens** of an atom, denoted $\mathcal{D}(a)$ is defined as

$\mathcal{D}(a) = \{A \mid (a = A) \in D\}$ when $a \in \mathcal{D}om(D)$

$\mathcal{D}(a) = \{\bot\}$ otherwise

The interpretation of a definition is given by the "D-consequence" relation $\vdash_D$. $\vdash_D$ is a relation between finite sets of conditions and conditions. When it is clear from the context which definition is intended, the index will be dropped. Expressions of the form $\Gamma \vdash C$ will be called **sequents**, as usual. In every case, the assumption Γ is an unordered finite set of conditions. A sequent such as $\Gamma,C \vdash C'$ has the set $\Gamma \cup \{C\}$ as its assumptions set. Note that in this case C may occur in Γ, i.e. the union is not disjoint.

The set $\mathcal{D}ef(D)$ **defined by** a partial inductive inductive D, is the set of atoms $\{a \in U \mid \vdash_D a\}$.

$\vdash$ is defined to be the smallest relation satisfying all the following properties:

$$\Gamma,a \vdash a \qquad \text{(axiom)}$$

$$\Gamma,\bot \vdash C \qquad (\bot)$$

$$\Gamma \vdash \wedge \qquad (\wedge)$$

$$\frac{\{\Gamma \vdash C_i \mid i \in I\}}{\Gamma \vdash (C_i)_{i \in I}} \qquad (\vdash())$$

$$\frac{\Gamma,C_i \vdash C}{\Gamma,(C_i)_{i \in I} \vdash C}\ i \in I \qquad (() \vdash)$$

$$\frac{\Gamma,C \vdash C'}{\Gamma \vdash C \to C'} \qquad (\vdash \to)$$

$$\frac{\Gamma \vdash C' \quad \Gamma,C'' \vdash C}{\Gamma,C' \to C'' \vdash C} \qquad (\to \vdash)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash a}\ A \in \mathcal{D}(a) \qquad (\vdash D)$$

$$\frac{\{\Gamma,A \vdash C \mid A \in \mathcal{D}(a)\}}{\Gamma,a \vdash C} \qquad (D \vdash)$$

The use of a set expression as premise should be taken to mean that the premises of the rule are the elements of the set expression.

EXAMPLE 2.1
Let D be the definition comprising the clauses a=b, a=c, b=^ and c=d. Then we have $\mathcal{D}ef(D)=\{a,b\}$. ®

The properties of - can be taken as the inference rules of a formal system, called the **calculus** of the particular partial inductive definition. In the calculus, it will be convenient to regard the assumptions as unordered sequences instead of sets. We will then have to include the contraction rule

$$\frac{\Gamma,C,C \text{ - } C'}{\Gamma,C \text{ - } C'} \qquad \text{(contraction)}$$

It is sometimes convenient to also include the weakening rule. We can see from the rules above that adding extra assumptions to all sequents in a subderivation never invalidates that derivation. The extra assumptions are simply added to Γ and play no part in the rules. Thus the weakening rule does not add any more power to the system, but can be included or excluded at will.

$$\frac{\Gamma \text{ - } C'}{\Gamma,C \text{ - } C'} \qquad \text{(weakening)}$$

The infinitary nature of this calculus can be seen in rules -() and D-, which can both take an infinite number of premises, provided the set I is infinite. In earlier formulations of partial inductive definitions, the set of assumptions could also be infinite, as the right-hand side of a clause could be an infinite set of conditions. The D--rule would then bring the infinite number of conditions into the assumption of a sequent. In the present version, all sets of conditions are encapsulated in the () construction, so it is sufficient to have finite sets as assumptions.

From the formulation of the rules, we can see that ^ has the same properties as $(C_i)_{i\in\varnothing}$, for any C_i. Also ⊥ has the same properties as any undefined atom. Thus, we do not actually need ^ and ⊥, they could be removed without changing the properties or expressiveness of the system.

3. FINITARY PARTIAL INDUCTIVE DEFINITIONS

As we could see in the previous section, there are two main reasons for the infinitary nature of partial inductive definitions - that a definition could comprise an infinite number of clauses and that the $(C_i)_{i\in I}$ -conditions could be infinite collections. To avoid this, we will define finite representations of the concepts from section 2 and define a finitary calculus that will let us derive that a particular sequent Γ - C holds, only in those cases where the corresponding represented sequent holds according to the rules in section 2. Any represented concept can be mapped to the corresponding pure concept. We will use [ ] to denote this mapping.

We assume there is some set of **terms**. To avoid restricting the applicability of the finitary calculus, we will make very few assumptions about terms. Essentially, we only assume that terms can contain variables that can be replaced by other terms, that equality is decidable, and

that axiom 3.3 below holds. These very general requirements permit different kinds of terms, such as first-order terms, higher-order terms, various equality theories etc.

Unless we say otherwise, the examples of this article will use ordinary first-order terms or higher-order terms of the simply typed lambda calculus, with two terms being equal if they, or their $\alpha\beta\eta$-normal forms, are identical. Actually, in applications of the finitary system, we are only interested in these kinds of terms. However, as the presentation could be extended with little complication to include much more general term theories, it seemed reasonable to do this.

To represent the universe U, we will take the set of variable-free terms. We assume that the mapping [ ] from variable-free terms to atoms is given. The mapping will not be defined for terms with variables. It must hold that $[x]=[y] \rightarrow x=y$, i.e. [ ] must be injective.

We divide the set of variables into two classes, the class of **parameters**, and the class of ordinary variables. Ordinary variables will be denoted by X, Y, Z,… Parameters will be denoted by X^*, Y^*, Z^*,... As usual, the particular name of a variable or parameter should not matter.

The special significance of parameters will be formally explained below. Intuitively, parameters occurring in sequents can be seen as variables that are implicitly universally quantified on the meta level - a dual to ordinary variables that can be seen as being implicitly existentially quantified on the meta level. For historical reasons, parameters are sometimes called "starred variables". Parameters having a special role in induction schemas will be called **induction parameters**, see section 4.

Any bound variables will be part of the internal structure of terms and are not considered here, except that it is assumed that the usual precautions are taken to avoid capture of free variables when a variable is replaced with a term.

In the sequel, the term "**variable**" will always refer to ordinary variables. All definitions and concepts regarding variables should be assumed to be defined in the analogous way for parameters also.

Terms not containing any variables will be called **ground**. A term not containing parameters is called **simple**. Note that a ground term may contain parameters and that a simple term may contain variables.

With any variable, we associate a **range**, which is the set of ground simple terms that are intended to potentially be substituted for it. The range will be denoted $ran(X)$. The **extended range** is the largest set of terms which has the range as the set of its ground and simple instances, i.e. $\mathcal{RAN}(X) = \{t \mid t\sigma \in ran(X), t \text{ is ground and simple}\}$. We can see that the extended range is closed under substitution, i.e. for any substitution τ, if $t \in \mathcal{RAN}(x)$ then $t\tau \in \mathcal{RAN}(x)$. In a typed system, the extended range of a variable could be a set of terms with the same type. Parameters are given ranges in the analogous way. In our examples, we will generally let it be clear from the context what the ranges of variables and parameters are.

Apart from terms, we will use a few special expressions, finitary conditions and finitary sequents. These will be defined below. Finitary conditions, finitary sequents, terms, or sets and sequences of these will together be called just **expressions**.

The notation E(X) is used to make explicit the occurrence of the variable X in the expression E. E(t) will denote the expression obtained by replacing all occurrences of X in E by t. Note that it is permissible to write E(X), even if E does not contain X.

A **substitution** is a set of variable-term pairs, as usual, where every term must be in the range of the corresponding variable. Substitutions will be denoted σ, τ, … A particular substitution is written as $\{X_1/t_1,\ldots,X_n/t_n\}$. We will assume that a substitution never contains an identical pair, X/X, for some X. In the same way, a **parameter substitution** is a set of parameter-term pairs. Parameter substitutions will be denoted σ^*, τ^*,… We will not use any particular symbol to denote a substitution that can substitute both variables and parameters. Any such substitution must be written as an explicit composition of variable and parameter substitutions, e.g. $\sigma\sigma^*$.

In the usual way, applying a substitution to a term involves simultaneously replacing the variables of every variable-term pair with the corresponding term. The result of applying a substitution to an special expression is the same expression with the substitution applied to all component terms. Composition of substitutions will be done in the usual way.

We call the set of all variables occurring in the variable part of the variable-term pairs of a substitution the **domain** of that substitution. Formally, $\mathcal{DOM}(\sigma)=\{X \mid (X/t)\in\sigma\}$, is the domain of the substitution σ. Informally, the domain can be said to be the set of variables "affected" by the substitution.

We will denote the set of variables **occurring in** (the normal form of) an expression E by $\mathcal{VAR}$(E). Formally, $\mathcal{VAR}(E) = \{X \mid \exists t\ E\{X/t\}\neq E\}$. The set of variables occurring in a variable (parameter) substitution σ, are the variables than can be introduced by an application of the substitution. Formally, $\mathcal{VAR}(\sigma) = \cup\{\mathcal{VAR}(t) \mid (X/t)\in\sigma\}$. $\mathcal{PARM}$(E) and $\mathcal{PARM}(\sigma)$ are defined similarly.

We may also speak of the restriction of a composite variable-parameter substitution $\sigma\sigma^*$ to variables (or parameters). By this we mean $(\sigma\sigma^*)\backslash\mathcal{V}$ where $\mathcal{V}$ is the set of all variables (parameters).

EXAMPLE 3.1
Let $E=p(X^*,\lambda x.x(Y))$ and $\sigma=\{Y/f(Z)\}$. Then $\mathcal{VAR}(E)=\{Y\}$, $\mathcal{PARM}(E)=\{X^*\}$, $\mathcal{DOM}(\sigma)=\{Y\}$, $\mathcal{VAR}(\sigma)=\{Z\}$ and $\mathcal{PARM}(\sigma)=\varnothing$. ®

A **ground (simple) substitution** will be a substitution where all terms are ground (simple). σ is a **grounding (simplifying) substitution** for an expression E, if $E\sigma$ is a ground (simple) expression. The **restriction** of a substitution σ to an expression E, written $\sigma\backslash E$, is a substitution, such that $E\sigma=E(\sigma\backslash E)$, but $X=X(\sigma\backslash E)$, for every variable $X\notin\mathcal{VAR}(E)$. A substitution σ is **nonredundant** for E, if $\sigma=\sigma\backslash E$, i.e. if $X=X\tau$, for every variable $X\notin\mathcal{VAR}(E)$. These concepts are also defined for parameter substitutions in the obvious way.

We have the following important lemma:

LEMMA 3.2 Substitution equality lemma
$\sigma=\tau$ iff $X\sigma=X\tau$ for all variables X.
PROOF
Follows immediately from the definition of substitutions. ®

AXIOM 3.3 Substitution equality axiom
$a\sigma=a\tau$ iff for all variables $X\in \mathcal{VAR}(a)$, $X\sigma=X\tau$.
NOTE
This axiom is a property that must hold for any term theory used with the finitary system. Given the definition of substitutions, this property is clearly true for the usual term theories such as first order terms or lambda terms. As an alternative to this axiom, it would possible to take as axioms more primitive properties, namely as the existence of a computable normal form of every term and that structural induction can be done over terms. From these more primitive properties, the statement of the present axiom could be obtained as a lemma. ®

Substitutions may be composed in the usual way. As usual, a substitution σ is **less general** than a substitution τ, written $\sigma\leq\tau$ iff $\exists\upsilon$: $\sigma=\tau\upsilon$. The same holds for parameter substitutions.

EXAMPLE 3.4
Let $\sigma=\{X/f(A)\}$, $\tau=\{X/f(3)\}$ and $\upsilon=\{X/f(A), Y/a\}$. Then $\tau\leq\sigma$, $\upsilon\leq\sigma$, but neither $\tau\leq\upsilon$, nor $\upsilon\leq\tau$.

We will need a way to compare sets of instances of expressions. We say that a *set* S is **less general** than the set T, written $S\leq T$, iff $\forall s\in S\ \exists t\in T\ \exists\upsilon\ s=t\upsilon$. Intuitively, this means that the set of variable instances of S is a subset of the set of (variable) instances of T. Without loss of generality, we can assume that υ is nonredundant for t. The sets S and T are **equally general**, written $S\cong T$, iff $S\leq T$ and $T\leq S$. Note that if S and T are ground, these concepts are exactly subset and equality, respectively. We define $\leq^*$ and $\cong^*$ for parameter instances in the analogous way.

EXAMPLE 3.5
Let $R=\{p(2,2), p(X,X)\}$, $S=\{p(X,Y)\}$ and $T=\{p(1,1), p(X,X)\}$. Then we have $R\leq S$ and $R\cong T$. ®

It is easy to show the following lemma.

LEMMA 3.6
1) If $S\cong T$ and $T\cong V$, then $S\cong V$, likewise $S\cong^* T$ and $T\cong^* V$, then $S\cong^* V$.
2) If $S\cong T$, then $S\theta\cong T\theta$, provided that $(\mathcal{DOM}(\theta)\cup\mathcal{VAR}(\theta))\cap(\mathcal{VAR}(S)\cup\mathcal{VAR}(T))=\emptyset$, likewise if $S\cong^* T$, then $S\theta^*\cong^* T\theta^*$, provided that $(\mathcal{DOM}(\theta^*)\cup\mathcal{PARM}(\theta^*))\cap(\mathcal{PARM}(S)\cup\mathcal{PARM}(T))=\emptyset$
3) If $S\cong T$, then $S\theta^*\cong T\theta^*$, provided that $\mathcal{VAR}(\theta^*)\cap(\mathcal{VAR}(S)\cup\mathcal{VAR}(T))=\emptyset$, likewise if $S\cong^* T$, then $S\theta\cong^* T\theta$, provided that $\mathcal{PARM}(\theta)\cap(\mathcal{PARM}(S)\cup\mathcal{PARM}(T))=\emptyset$

PROOF
The first part follows easily from the definitions of $\cong$ and $\leq$.

To prove the second part, we show that if $S \leq T$, then $S\theta \leq T\theta$. By analogy, this implies that if $T \leq S$, then $T\theta \leq S\theta$. Together the we establish that if $S \cong T$, then $S\theta \cong T\theta$. The proof for $\cong^*$ is done in the same way.

Assume that $S \leq T$, and that $(\mathcal{DOM}(\theta) \cup \mathcal{VAR}(\theta)) \cap \mathcal{VAR}(T) = \emptyset$. By the definition of $\leq$, $\forall s \in S\ \exists t \in T\ \exists \upsilon\ s = t\upsilon$, so for a particular s, we have $s = t\upsilon$, for some t and υ. From this we get $s\theta = t\upsilon\theta$. Let $\tau = (\upsilon\theta) \setminus \mathcal{VAR}(t)$. For each $X \in \mathcal{VAR}(t)$, $X\upsilon\theta = X\tau = X\theta\tau$, since $X \in \mathcal{VAR}(t)$ implies $X \notin \mathcal{DOM}(\theta)$. For each $X^* \in \mathcal{PARM}(t)$, $X^*\upsilon\theta = X^* = X^*\theta\tau$. By the substitution equality axiom, we get $t\upsilon\theta = t\theta\tau$. Together with $s\theta = t\upsilon\theta$, we get $s\theta = t\theta\tau$. By the definition of $\leq$, this implies that $S\theta \leq T\theta$.

The proof of the third part is very similar to that of the second part. Again we show that if $S \leq T$, then $S\theta^* \leq T\theta^*$ with the result for $\geq$ and $\cong^*$ being obtained through analogy.

Assume that $S \leq T$, and that $\mathcal{VAR}(\theta^*) \cap \mathcal{VAR}(T) = \emptyset$. By the definition of $\leq$, $\forall s \in S\ \exists t \in T\ \exists \upsilon\ s = t\upsilon$, so for a particular s, we have $s = t\upsilon$, for some t and υ. From this we get $s\theta^* = t\upsilon\theta^*$. Let $\tau = (\upsilon\theta^*) \setminus \mathcal{VAR}(t)$. Clearly, for each $X \in \mathcal{VAR}(t)$, $X\upsilon\theta^* = X\tau = X\theta^*\tau$, and for each $X^* \in \mathcal{PARM}(t)$, $X^*\upsilon\theta^* = X^*\theta^* = X^*\theta^*\tau$. The last step of the equality is true since θ^*, and thus $X^*\theta^*$, has no variables in common with T, and thus also not with t, since $t \in T$. By the substitution equality axiom, we get $t\upsilon\theta^* = t\theta^*\tau$. Together with $s\theta^* = t\upsilon\theta^*$, we get $s\theta^* = t\theta^*\tau$. By the definition of $\leq$, this implies that $S\theta^* \leq T\theta^*$. ®

We will define a **complete set of unifiers (CSU)** [16] of a and b to be a set of substitutions, Σ, such that

- $\forall \sigma \in \Sigma\ a\sigma = b\sigma$ (every member of Σ is a unifier)
- $\forall \sigma\ (a\sigma = b\sigma \rightarrow \exists \tau \in \Sigma\ \sigma \leq \tau)$ (any unifier is less general than some member of a CSU)

CSU´s are not unique. In the sequel we will usually write a CSU of a and b as $\Sigma(a,b)$. Here Σ should be understood as a choice function, selecting an arbitrary CSU of a and b. However, for every particular context (e.g. inference rule instance) each $\Sigma(a,b)$ must denotes the same arbitrary CSU.

We will also impose some minimality conditions on all CSUs in this paper:

- $\forall \sigma \in \Sigma\ \sigma = \sigma\sigma$ (every member of Σ is idempotent)
- $\forall \sigma \in \Sigma\ \sigma$ is nonredundant for a and b

It would be desirable to have additional minimality properties of CSUs, e.g. that for different $\sigma, \tau \in \Sigma$, neither $\sigma \leq \tau$ nor $\tau \leq \sigma$. As noted by Huet [17], this is not always possible.

The idempotency condition is not strictly a minimality condition, and it is not always required of unifiers in the literature. However, a strong case can be made for this requirement [19,23], and in the present context it simplifies several proofs. It it easy to show that idempotent unifiers always exist. Clearly, if a unifier σ of a and b is not idempotent, it must be the case that $\mathcal{DOM}(\sigma) \cap \mathcal{VAR}(\sigma) \neq \emptyset$. Let $V = \mathcal{DOM}(\sigma) \cap \mathcal{VAR}(\sigma)$. Define $\tau = \{\{v/x_v\} \mid v \in V\}$, where the

different x_v are distinct variables not occurring in $\mathcal{DOM}(\sigma)\cup\mathcal{VAR}(\sigma)$. Then $(\sigma\tau)\backslash\mathcal{DOM}(\sigma)$ will be a unifier of a and b, and $\mathcal{DOM}((\sigma\tau)\backslash\mathcal{DOM}(\sigma))\cap\mathcal{VAR}((\sigma\tau)\backslash\mathcal{DOM}(\sigma))=\emptyset$, so it is idempotent.

NOTE 3.7
Since the names of variables and parameters have no significance, variables and parameters that are in σ, but not in a or b, can be renamed arbitrarily. In particular, for any variable X, such that $X\notin\mathcal{VAR}(a)$ and $X\notin\mathcal{VAR}(b)$, we can always assume that $X\notin\mathcal{VAR}(\sigma)$. Likewise for parameters.

It is interesting to note that if Σ and Φ are two CSU´s of a and b and we take ≅ to be a formal relation (i.e. we care only about the formula defining ≅, not about its intended meaning), then Σ ≅ Φ. This fact could be used as the basis for an alternative definition of CSU´s.

We define a **parameter-CSU (CSU*)** in the analogue manner, usually denoting a CSU* of a and b by $\Sigma^*(a,b)$.

EXAMPLE 3.8
Let s=p(X) and t=p(f(Y)), then {{X/f(Y)}} is a CSU of s and t, while {{X/f(3), Y/3}}, {{X/f(X), Y/X}}, and {{X/f(Y), Z/1}} are not.

Let $s=F^*(X^*)$ and t=f(a). A CSU* of s and t is
$\{\{F^*/\lambda x.f(a)\}, \{F^*/f, X^*/a\}, \{F^*/\lambda x.x, X^*/f(a)\}\}$. ®

We define the **(finitary) conditions** to be all of the following:

t	where t is any term
$C_1,C_2,\ldots,C_n$	where every C_i is a condition. $n\geq 1$.
$C\Rightarrow C'$	where C and C′ are conditions
$\Pi x\ C(x)$	where C is a condition containing the variable x. (x is bound by Π)

Parentheses will be used around the second form when necessary. $C_1,C_2,\ldots,C_n\Rightarrow C$ will be taken as $(C_1,C_2,\ldots,C_n)\Rightarrow C$.

The Π-construction is intended to represent the collection of all conditions obtained by substitutions for x. The intentions of the other conditions are obvious. Formally, the mapping [ ] is defined on conditions other than terms as:

$[C_1,C_2,\ldots,C_n] = ([C_i])_{i\in\{1,\ldots,n\}}$
$[C\Rightarrow C'] = [C]\rightarrow[C']$
$[\Pi x\ C(x)] = ([C(t)])_{t\in ran(x)}$

Note that we have no finitary conditions that represent the two conditions ^ and ⊥ of the infinitary calculus. As we remarked at the end of the previous section, these conditions are superfluous so we do not lose in expressiveness by not being able to represent them.

A **clause** is an expression of the form

$H \Leftarrow B$

where the **head** H is a term and the **body** B is a condition. Neither of these may contain parameters. The body may also be empty, which should be interpreted as the condition (), i.e. the empty sequence of conditions.

Intuitively, free variables in a clause are universally quantified. Each clause represents the union of its ground instances. Formally the mapping [ ] is defined on clauses as:

$$[H \Leftarrow B] = \{[H\sigma] = [B\sigma] \mid \sigma \text{ is a grounding simple substitution of } H\}$$

Without loss of generality, we can assume that σ above is nonredundant for H and B.

NOTE 3.9
From this definition, we can see that variable renamings in $H \Leftarrow B$, do not affect the value of $[H \Leftarrow B]$. In the sequel, we will assume that each time a clause is used, the variables in the clause can be renamed as required to avoid conflicts with other variables, i.e. whenever a definition P (or its parameter transform P^* - see below) is used, the variables (parameters) can be renamed arbitrarily to fulfil any necessary conditions. ®

A **finitary partial inductive definition** is a finite set of clauses. The mapping [ ], on a definition D, is defined simply as the union of the mappings of the clauses, i.e.

$$[D] = \bigcup_{C \in D} [C]$$

The **parameter transform** of a clause is obtained by replacing each free variable in the clause with a unique parameter. The **parameter transform** P^*, of a finitary definition P, is obtained by taking the parameter transform of every clause in P.

EXAMPLE 3.10
In many of the examples in this article, we will use the following finitary partial inductive definition, FOL, intended to model the semantics of the logical connectives of first order logic with equality. The construction $\forall x\ p(x)$ should be seen as syntactic sugaring for the term $\forall(\lambda x.p(x))$, similarly for $\exists$. The variables of the definition range over formulae, represented terms (not terms of the finitary theory of partial inductive definitions) or functions from represented terms to formulae, as appropriate.

$A=A \Leftarrow$
$A \wedge B \Leftarrow A,B$
$A \vee B \Leftarrow A$
$A \vee B \Leftarrow B$
$A \rightarrow B \Leftarrow (A \Rightarrow B)$
$\neg A \Leftarrow (A \Rightarrow \text{false})$
$\exists x\ A(x) \Leftarrow A(X)$
$\forall x\ A(x) \Leftarrow \Pi x\ A(x)$

The term "false" must not be defined by any clause.
The parameter transform of the last clause is $\forall x\ A^*(x) \Leftarrow \Pi x\ A^*(x)$. The mapping of the same clause, $[\forall x\ A(x) \Leftarrow \Pi x\ A(x)]$ is a set of clauses:

$allf_1=\{f_1(t_1), f_1(t_2), f_1(t_3),\ldots\}$
$allf_2=\{f_2(t_1), f_2(t_2), f_2(t_3),\ldots\}$
:

where the f_i form an enumeration of the mappings of all functions in the restricted range of A, the t_i form an enumeration of the mappings of the represented terms, and each $allf_i$ is the mapping of $\forall x\ A(x)$, for each function in the restricted range of A. ®

The **definiens** of a term, denoted $\mathcal{D}(a)$ is defined as
$\mathcal{D}(a) = \{B\sigma \mid a=H\sigma, (H \Leftarrow B) \in P\}$

The **defining clauses** of a term, denoted $\mathcal{C}(a)$ is defined as
$\mathcal{C}(a) = \{(H\sigma \Leftarrow B\sigma) \mid \sigma \in \Sigma(a,H), (H \Leftarrow B) \in P\}$

The **parameter definiens** ($\mathcal{D}^*$) and **defining parameter clauses** ($\mathcal{C}^*$) are defined analogously, using parameter substitutions and the parameter transform of the program.

EXAMPLE 3.11
With the definition FOL above, we have $\mathcal{D}(a \wedge b)=\{(a,b)\}$, $\mathcal{D}(a \vee b)=\{a,b\}$ and $\mathcal{C}^*(a \vee b)=\{a \vee b \Leftarrow a,\ a \vee b \Leftarrow b\}$.

If we instead take the following definition:

$p(A, A) \Leftarrow q$
$p(a, b) \Leftarrow q$

we will have $\mathcal{D}^*(p(X,Y^*))=\{q\}$ and $\mathcal{C}^*(p(X,Y^*))=\{p(X, X) \Leftarrow q,\ p(a, b) \Leftarrow q\}$. ®

The set $\mathcal{D}ef(D)$ **defined by** a finitary partial inductive inductive definition D, is the set of ground simple terms $\{t \mid \vdash_D t\}$, where $\vdash_D$ is derivability in the variable-free finitary calculus described in the next section.

4. A FINITARY CALCULUS

How do we avoid the infinitary character of the calculus of section 2? As we remarked at the end, the rules $\vdash()$ and $D \vdash$ could have an infinite number of premises. This would happen in the first case if the index set I of $(C_i)_{i \in I}$ was infinite, in the second case if there was an infinite number of clauses defining a. In the finitary representation of clauses given in the previous section, infinite index sets could arise only in connection with the Π construct, infinite sets of clauses could arise only if a finitary clause contained free variables. In each case the infinite set is generated uniformly by substituting ground simple terms for variables. Since the different premises differ only in the substituted term, the infinite set of premises can be represented by a premise where the infinite set can be generated uniformly by some substitution. In our finitary calculus, parameters are used for this purpose. A sequent containing parameters, actually represent the set of its simple instances.

Formally, [ ] is defined on sequents as:

$$[\Gamma \text{ - } C] = \{[\Gamma\sigma^*] \text{ - } [C\sigma^*] \mid \sigma^* \text{ is a ground simplifying substitution of } \Gamma \text{ and } C\}$$

There will actually be two versions of the finitary calculus, a variable-free calculus where no variables are permitted in a derivation, and a calculus with variables. The two calculi will use the same inference rules, but will be used for slightly different purposes. The variable-free calculus will be the "proper" finitary calculus where every derivation is the representation of a derivation in the infinitary calculus. Note that the mapping of finitary sequents to infinitary sequents is only defined in the variable-free case. For this reason, a derivation in the calculus with variables does not, in general, represent an infinitary derivation. The advantage with having variables in a derivation is that the variables can be used as placeholders for undetermined terms. By substituting ground terms for every variables, a derivation with variables can be instantiated into a "proper" derivation.

Soundness properties with respect to the infinitary calculus will only be shown for the variable-free finitary calculus. The soundness of the calculus with variables will be shown relative to the variable-free calculus.

We will now present a set of inference rules for the two finitary calculi of partial inductive definitions. In the case of the variable-free calculus, it should be assumed that all expressions (except clauses) are ground, that parameter substitutions are ground and that variable substitutions are grounding for all expressions they are applied to. Again, the use of a set expression as premise should be taken to mean that the premises of the rule are the elements of the set expression.

$$\Gamma,a \text{ - } a \qquad \text{(axiom)}$$

$$\frac{\Gamma \text{ - } C(X^*)}{\Gamma \text{ -}\Pi x\ C(x)} \qquad (\text{-}\Pi)$$

where $X^* \notin \mathcal{PARM}(\Gamma)$, $X^* \notin \mathcal{PARM}(C)$,
and $\mathcal{RAN}(x) = \mathcal{RAN}(X^*)$. X^* must not be an induction parameter.

$$\frac{\Gamma,C(t) \text{ - } C'}{\Gamma,\Pi x\ C(x) \text{ - } C'} \qquad (\Pi \text{ -})$$

where t is some arbitrary term in $\mathcal{RAN}(x)$.

In section 8, we will return to the question of how to choose the term t.

$$\frac{\Gamma \text{ - } C_1 \quad \Gamma \text{ - } C_2 \quad \cdots \quad \Gamma \text{ - } C_n}{\Gamma \text{ -}(C_1,C_2,\ldots,C_n)} \qquad (\text{-}())$$

$$\frac{\Gamma, C_i \text{ - } C}{\Gamma, (C_1, C_2, \ldots, C_n) \text{ - } C} \quad i \leq n \qquad (() \text{ -})$$

$$\frac{\Gamma, C \text{ - } C'}{\Gamma \text{ - } C \Rightarrow C'} \qquad (\text{-} \Rightarrow)$$

$$\frac{\Gamma \text{ - } C' \quad \Gamma, C'' \text{ - } C}{\Gamma, C' \Rightarrow C'' \text{ - } C} \qquad (\Rightarrow \text{-})$$

$$\frac{\Gamma \text{ - } B\sigma}{\Gamma \text{ - } a} \quad (H \Leftarrow B) \in P,\ a = H\sigma \qquad (\text{-D})$$

$$\frac{\left\{ \Gamma\sigma^*, B\sigma^* \text{ - } C\sigma^* \mid \sigma^* \in \Sigma^*(a,H),\ (H \Leftarrow B) \in P^* \right\}}{\Gamma, a \text{ - } C} \qquad (\text{D -})$$

where each $\Sigma^*(a,H)$ is *finite*.
$\mathcal{C}^*(a)\varphi \cong^* \mathcal{C}^*(a\varphi)$, for all φ.
$\mathcal{PARM}(P^*) \cap \mathcal{PARM}(\Gamma, a \text{ - } C) = \varnothing$
$\mathcal{PARM}(P^*)$ must not contain any induction parameters.

Recall that P^* denotes the parameter transform of the definition P. In the variable-free calculus, the condition that $\mathcal{C}^*(a)\varphi \cong^* \mathcal{C}^*(a\varphi)$, for all φ, will be trivially satisfied. In section 8 we will return to the question of verifying this condition. For an example of the necessity of the condition, see example 7.2.

$$\frac{\Gamma, C, C \text{ - } C'}{\Gamma, C \text{ - } C'} \qquad \text{(contraction)}$$

$$\frac{\Gamma \text{ - } C'}{\Gamma, C \text{ - } C'} \qquad \text{(weakening)}$$

In the infinitary calculus, weakening was an optional rule. In the presence of the induction schema below, that is not the case here.

The - Π and D - rules have side conditions to prevent key parameters (i.e. $\mathcal{PARM}(P^*)$ in the D - rule and X^* in the - Π rule) from occurring both in the premises and conclusion of the same rule instance. It might seem that this condition is too permissive and that stronger requirements should be made to ensure the uniqueness of these parameters over larger parts of the derivation. However, the definition of the mapping of a finitary sequent to a set of infinitary sequents shows that the significance of a particular parameter is local to the sequent in which it occurs. Thus there is no problem with the same parameter reoccurring at arbitrary places in the derivation as long as the present side conditions are fulfilled.

The use of unifiers in the D - rule needs some motivation. Suppose a definition includes the clause A=A⇐. Consider the sequent $X^*=Y^*$, $Y^*=Z^*$ - $X^*=Z^*$. What premises are needed if we want to derive this sequent using the D - rule on $X^*=Y^*$? According to the interpretation of finitary sequents, this sequent maps to an infinite set of infinitary sequents, generated by all possible ground simple instantiations of the three parameters. Call this set of infinitary sequents S. Let x be the mapping of some instantiation of X^* and y the mapping of some instantiation of Y^*. We can see that all the infinitary sequents in S with $x \neq y$ will be false by absurdity, since the "=" relation only holds between equal terms. In other words, to derive the sequent above, it suffices with a premise that maps to that subset of S where x=y. Only those cases have to be derived, the other ones hold by absurdity. We can see that $X^*=Z^*$ - $X^*=Z^*$ is a possible sequent.

Now, consider the finitary rule for D -. Since there is only one clause defining "=", there should be one premise for each element of some CSU of $X^*=Y^*$ and $A^*=A^*$ – the latter being the head of the parameter transform of the clause. Clearly $\{A^*/X^*, Y^*/Z^*\}$ is such a CSU. Using this CSU we arrive at exactly the premise $X^*=Z^*$ - $X^*=Z^*$. Example 4.2 presents a derivation that includes precisely this step.

To simplify the presentation of a derivation, we will frequently apply the () - or the -() rules without explicit mention whenever a sequence condition appears. E.g. if the definition contains a clause

$$a \Leftarrow b,c$$

we will write:

$$\frac{\Gamma \text{ - } b \quad \Gamma \text{ - } c}{\Gamma \text{ - } a} \qquad (\text{-D})$$

rather than:

$$\frac{\dfrac{\Gamma \text{ - } b \quad \Gamma \text{ - } c}{\Gamma \text{ - } (b,c)}\ (\text{-()})}{\Gamma \text{ - } a}\ (\text{-D})$$

We sometimes want a derivation of a sequent with a sequence $(C_1,\ldots,C_n)$ in the antecedent, from a sequent with the individual conditions $C_1,\ldots,C_n$ in the antecedent. Such a derivation can be constructed simply from a series of () - rule instances, and a series of contractions, e.g.

$$\frac{\dfrac{\dfrac{\Gamma,a,b \text{ - } C}{\Gamma,(a,b),b \text{ - } C}\ (() \text{ -})}{\Gamma,(a,b),(a,b) \text{ - } C}\ (() \text{ -})}{\Gamma,(a,b) \text{ - } C}\ (\text{contraction})$$

Again to simplify the presentation of a derivation, we will abuse the calculus by writing a single () - step, with all of $C_1,\ldots,C_n$ in the antecedent of the premise, e.g.

$$\frac{\Gamma,a,b \text{ - } C}{\Gamma,(a,b) \text{ - } C} \qquad (() \text{ -})$$

EXAMPLE 4.1
A sample derivation, given the definition FOL of example 3.10:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{q(X^*),\ p(X^*)\text{ - }q(X^*)}\text{(Axiom)}\quad\dfrac{}{p(X^*)\text{ - }p(X^*)}\text{(Axiom)}}{p(X^*),\ p(X^*)\Rightarrow q(X^*)\text{ - }q(X^*)}(\Rightarrow\text{ - })}{p(X^*),\ p(X^*)\rightarrow q(X^*)\text{ - }q(X^*)}(\text{P - })}{p(X^*),\ \Pi x\ p(x)\rightarrow q(x)\text{ - }q(X^*)}(\Pi\text{ - })}{p(X^*),\ \forall x(p(x)\rightarrow q(x))\text{ - }q(X^*)}(\text{P - })}{\Pi x\ p(x),\ \forall x(p(x)\rightarrow q(x))\text{ - }q(X^*)}(\Pi\text{ - })}{\forall x\ p(x),\ \forall x(p(x)\rightarrow q(x))\text{ - }q(X^*)}(\text{P - })}{\forall x\ p(x),\ \forall x(p(x)\rightarrow q(x))\text{ - }\Pi x\ q(x)}(\text{- }\Pi)}{\forall x\ p(x),\ \forall x(p(x)\rightarrow q(x))\text{ - }\forall x\ q(x)}(\text{- P})}{\forall x\ p(x)\wedge\forall x(p(x)\rightarrow q(x))\text{ - }\forall x\ q(x)}(\text{P - })}{\text{- }\forall x\ p(x)\wedge\forall x(p(x)\rightarrow q(x))\Rightarrow\forall x\ q(x)}(\text{- }\Rightarrow)}{\text{- }\forall x\ p(x)\wedge\forall x(p(x)\rightarrow q(x))\rightarrow\forall x\ q(x)}(\text{- P})$$

®

EXAMPLE 4.2
Given the definition FOL, the following derivation shows that equality so defined is transitive.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Y^*=Z^*\text{ -}Y^*=Z^*}\text{(Axiom)}}{X^*=Y^*,\ Y^*=Z^*\text{ -}X^*=Z^*}(\text{P -})}{X^*=Y^*\wedge Y^*=Z^*\text{ -}X^*=Z^*}(\text{P -})}{\text{-}X^*=Y^*\wedge Y^*=Z^*\Rightarrow X^*=Z^*}(\text{-}\Rightarrow)}{\text{-}X^*=Y^*\wedge Y^*=Z^*\rightarrow X^*=Z^*}(\text{-P})}{\text{-}\Pi z\ X^*=Y^*\wedge Y^*=z\rightarrow X^*=z}(\text{-}\Pi)}{\text{-}\forall z(X^*=Y^*\wedge Y^*=z\rightarrow X^*=z)}(\text{-P})}{\text{-}\Pi y\forall z(X^*=y\wedge y=z\rightarrow X^*=z)}(\text{-}\Pi)}{\text{-}\forall y\forall z(X^*=y\wedge y=z\rightarrow X^*=z)}(\text{-P})}{\text{-}\Pi x\forall y\forall z(x=y\wedge y=z\rightarrow x=z)}(\text{-}\Pi)}{\text{-}\forall x\forall y\forall z(x=y\wedge y=z\rightarrow x=z)}(\text{-P})$$

The following derivation step illustrates clearly how the premises of the D - rule corresponds to the elements of a CSU (see example 3.8). Since the assumption is $F^*(A^*)=f(a)$, there will be a premise for each unifier of $F^*(A^*)$ and $f(a)$. By keeping F^* and A^* in the conclusion, we can see what effect the various unifiers have on these parameters.

$$\frac{\Gamma - (\lambda x.f(a),X^*) \quad \Delta - (f,a) \quad \Theta - (\lambda x.x,f(a))}{\Gamma,\Delta,\Theta,F^*(X^*)=f(a) - (F^*,X^*)} \qquad (D\text{-})$$

®

It should be noted that the decidability of the D - rule (and consequently the finitaryness of the calculus) depends on the possibility of finding finite CSU´s of two terms or showing that none exist. This is not possible in general, e.g. for higher order terms [16]. However, this is a problem mainly when a given derivation is to be checked for correctness. If we check each inference step when the derivation is constructed and only admit steps that have been positively shown to be correct, the calculus is useful even if unification is not decidable. Conversely, there can be an undecidability problem in showing that the -D rule is *not* applicable.

In the infinitary calculus, there was a clear duality between the -D and D - rules. This duality appears lost here. However, in the variable-free calculus, the -D rule can be expressed in a more complicated, but equivalent, way that uncovers the duality.

First note that since in this case we deal with the variable-free calculus, Γ and a are ground so $\Gamma=\Gamma\sigma$ and $a=a\sigma$, for every σ. Thus $a=H\sigma$ is equivalent to $a\sigma=H\sigma$, which means that σ should be a unifier of a and H. Then there is a CSU, $\Sigma(a,H)$, of a and H that contains σ. To construct such a CSU, take any CSU of a and H and add σ to it. Since σ will be grounding, $\sigma=\sigma\sigma$, so σ is a valid member of a CSU, i.e. the resulting set will be a possible $\Sigma(a,H)$. In other words, the conditions $a=H\sigma$ and $\sigma\in\Sigma(a,H)$ are equivalent. We can now give the -D rule an alternative formulation that clearly shows the duality .

$$\frac{\Gamma\sigma - B\sigma}{\Gamma - a} \quad \sigma\in\Sigma(a,H),\ (H\Leftarrow B)\in P$$

where every element in $\Sigma(a,H)$ is grounding for B.

The finitary calculus also has two inference rules that have no counterparts in the infinitary calculus: specialisation and induction. When manipulating derivations, and in connection with induction, we will need the specialisation rule:

$$\frac{\Gamma - C}{\Gamma\sigma^* - C\sigma^*} \qquad \text{(specialisation)}$$

where $\mathcal{DOM}(\sigma^*)$ does not contain any induction parameters.

Without loss of generality, we can assume that σ^* is nonredundant for Γ, C and C′. The intuition behind this rule is that if we have shown that some sequent is derivable, then any (parameter) instance of it is also derivable.

The introduction of parameters enabled us to give a finitary formulation of inferences with infinite number of premises (infinite width). It did not, however, solve the problem of arbitrary long inference chains (infinite height).

To address this issue, we include an induction schema. Induction is expressed as an "improper" inference rule, taking entire derivations as premises instead of sequents. The premise derivations are hypothetical derivations, in the same sense as hypothetical derivations in natural deduction. Each such derivation has one or more hypothetical sequents. Those sequents may be used in the derivation without being proved. Every hypothetical derivation corresponds to a induction step. The base case(s) of an induction are degenerate hypothetical derivations without any hypotheses.

The induction schema expresses induction over a set of ground simple terms. This set must be defined by a finitary definition that maps to an infinitary definition that is an ordinary inductive definition, i.e. the arrow condition must not be used. We call this definition T, i.e. the set is $\mathcal{D}ef(T)$. Strictly speaking, the Π-condition should not be permitted in T either, as it generally maps to an infinite condition in the infinitary calculus, which is not permitted in an ordinary inductive definition. For our purposes, however, this is still permissible as long as induction is not done over the ranges of Π-bound variables of T.

The induction schema will have the following form:

$$\frac{\left\langle \left\{ \begin{array}{c} S(C) \mid C \in \langle B \rangle \\ \vdots \\ S(H) \end{array} \right\} \;\middle|\; (H \Leftarrow B) \in T^* \right\rangle}{S(X^*)} \qquad \text{(induction)}$$

where for every H and $\langle B \rangle$, $ran(X^*)=\mathcal{D}ef(T)$, $\mathcal{PARM}(H)\cap\mathcal{PARM}(S)=\varnothing$, and $\mathcal{PARM}(H)\cap\mathcal{PARM}(\langle B\rangle)=\varnothing$.

Each premise is a hypothetical derivation. The inner set expression gives the set of hypothetical sequents for each derivation. The expression $\langle B \rangle$ denotes some set of **hypothetical terms** of the clause body B. This is the set of constituent terms of B, with each variable bound by a Π-operator replaced by some parameter not occurring elsewhere in B. Different $\langle B \rangle$ differ only in the names of parameters. Note that $\langle B \rangle$ will always be a finite set. The parameters of each clause will be the **induction parameters** of the hypothetical derivation corresponding to that clause and all of its subderivations. Those parameters will not be considered as induction parameters in the derivation *containing* the induction schema. Note that the parameters introduced by replacing Π-bound variables of B are also *not* considered as induction parameters.

Formally, $\langle B \rangle$ is defined as

$\langle t\rangle=t$	where t is a term.
$\langle C_1,C_2,\ldots,C_n\rangle=\cup\langle C_i\rangle$	
$\langle \Pi x\ C(x)\rangle=\langle C(X^*)\rangle$	where X^* is a unique parameter.

EXAMPLE 4.3
Let the set of lists be defined by the inductive definition LIST:

nil ⇐
U.L ⇐ L

Add to the definition FOL the definition of the append relation for lists:

append(nil,X,X) ⇐
append(X.A,B,X.C) ⇐ append(A,B,C)

where the variables ranges over lists, i.e. *ran*(X)=*ran*(A)=*ran*(B)=*ran*(C)=*def*(LIST).

The following derivation shows that for all x, append(x,nil,x) holds:

$$\dfrac{\dfrac{\dfrac{}{\text{-append(nil,nil,nil)}}\,(\text{-P}) \qquad \dfrac{\dfrac{}{\text{-append}(L^*,\text{nil},L^*)}\,(\text{Hypothesis})}{\text{-append}(U^*.L^*,\text{nil},U^*.L^*)}\,(\text{-P})}{\text{-append}(X^*,\text{nil},X^*)}\,(\text{Induction})}{\text{-}\Pi x\ \text{append}(x,\text{nil},x)}\,(\text{-}\Pi)}{\text{-}\forall x\ \text{append}(x,\text{nil},x)}\,(\text{-P})$$

®

In this derivation U^* and L^* are induction parameters of the hypothetical derivation of the second premise of the induction rule.

The induction schema can also be used for case analysis, as the following example shows:

EXAMPLE 4.4
Given the definition FOL, we show that a list, defined as above, must be either empty or not empty. *ran*(x)=*def*(LIST).

$$\dfrac{\dfrac{\dfrac{}{\text{-nil=nil}}\,(\text{-P})}{\text{-nil=nil}\vee\neg\text{nil=nil}}\,(\text{-P}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{U^*.L^*=\text{nil}\ \text{-}\bot}\,(\text{P -})}{\text{-}U^*.L^*=\text{nil}\Rightarrow\bot}\,(\text{-}\Rightarrow)}{\text{-}\neg U^*.L^*=\text{nil}}\,(\text{-P})}{\text{-}U^*.L^*=\text{nil}\vee\neg U^*.L^*=\text{nil}}\,(\text{-P})}{\text{-}X^*=\text{nil}\vee\neg X^*=\text{nil}}\,(\text{Induction})}{\text{-}\Pi x\ x=\text{nil}\vee\neg x=\text{nil}}\,(\text{-}\Pi)}{\text{-}\forall x(x=\text{nil}\vee\neg x=\text{nil})}\,(\text{-P})$$

®

5. SOUNDNESS OF THE VARIABLE-FREE FINITARY CALCULUS

In this section we will show the soundness of the variable-free finitary calculus with respect to the infinitary calculus. Given a derivation in the variable-free finitary calculus of the form

$$\begin{array}{c} S_1 \quad \cdots \quad S_n \\ \vdots \\ S \end{array}$$

soundness amounts to showing the existence of a derivation in the infinitary calculus

$$\begin{array}{c} [S_1] \cdots [S_n] \\ \vdots \\ S' \end{array}$$

for every $S' \in [S]$. This is shown by induction over the structure of the finitary derivation.

An application of an inference rule in the finitary calculus has the following general form: ($n \geq 0$)

$$\frac{S_1 \cdots S_n}{S}$$

The conclusion S maps to a set of infinitary conclusions, [S], and the premises map to a set of infinitary premises, $\cup[S_k]$. To prove the soundness of the particular inference rule, we must show that for every application of a finitary inference rule, each sequent in [S] can be derived from a subset of the premises in $\cup[S_k]$, by an application of some rules from the infinitary calculus.

A particular application of an inference rule with n=0 will give a base case in the induction. With n>0 it will give an induction step.

The reason for taking a subset of $\cup[S_k]$ is that although all premises should be needed to derive all conclusions in [S], the derivation of a particular conclusion will not in general need every available premise. Ideally, every premise in $\cup[S_k]$, should be necessary for the derivation of some sequent in [S], otherwise the finitary rule will have premises that are, in a sense, redundant. To ascertain this, however, additional side conditions will have to be introduced in the formulation of the finitary rules. We have chosen to give a simpler formulation of the rules and accept that redundancies may occur.

EXAMPLE 5.1
Consider the following step of the finitary calculus:

$$\frac{p(X^*) \vdash q}{\Pi x\, p(x) \vdash q} \qquad (\Pi \vdash)$$

The conclusion maps to the infinitary sequent $([p(i)])_{i \in ran(x)} \vdash q$. According to the rules of the infinitary calculus, the $\Pi \vdash$ rule requires one premise, $[p(i)] \vdash q$ for some $i \in ran(x)$ to get the conclusion. However, the mapping of the premise of the finitary step above is the set of infinitary sequents $\{ ([p(i)]) \vdash q \mid i \in ran(x)\}$. Thus the premise of the finitary step is redundant. The redundancy could have been avoided by introducing side condition in $\Pi \vdash$ rule. Since that rule is still sound, we have chosen to accept the redundancy. ®

We will not attempt to extend the mapping [ ] to provide a translation from finitary to infinitary derivations. Although possible in principle, in the case of the D $\vdash$ rule, this will be quite complicated. Instead we will be satisfied with the soundness condition, which guarantees the existence of such a translated derivation.

Consider again the finitary rule application above. Suppose that the corresponding infinitary inference rule application has the form

$$\frac{\{S'_i \mid i \in I\}}{S'}$$

for some I. For each rule, we must show that for every $S' \in [S]$, the infinitary rule instance is correct, and $\cup S'_i \subseteq \cup [S_k]$. Recall that for a sequent S, $[S] = \{[S\sigma^*] \mid \sigma^*$ is a ground simplifying substitution for S$\}$. In other words, $S' = [S\sigma^*]$, for some such σ^*. Without loss of generality, we can assume that σ^* is nonredundant for S. For brevity, we will not explicitly mention the conditions on σ^* below.

We will give individual proofs for each rule below. For each rule, the form of the rule and the corresponding instance of an infinitary rule for an arbitrary σ^*, will be shown. In most cases it will be clear that the infinitary rule instance has a proper form, otherwise this will be proven. It will also be shown that the set of infinitary rule premises is indeed a subset of the set of infinitary sequents corresponding to the finitary rule premises.

Axiom

$$\Gamma, a \vdash a \qquad [\Gamma\sigma^*], [a\sigma^*] \vdash [a\sigma^*]$$

as there are no premises, correctness is immediate.

$\vdash$ ( )

$$\frac{\Gamma \vdash C_1 \quad \cdots \quad \Gamma \vdash C_n}{\Gamma \vdash (C_1,\ldots,C_n)} \qquad \frac{\{[\Gamma\sigma^*] \vdash [C_i\sigma^*] \mid i \in \{1,\ldots,n\}\}}{[\Gamma\sigma^*] \vdash ([C_i\sigma^*])_{i \in \{1,\ldots,n\}}}$$

From the definition of [ ], it is clear that $([\Gamma\sigma^*] \vdash [C_i\sigma^*]) \in [\Gamma \vdash C_i]$, for every $i \in \{1,\ldots,n\}$. Thus $\{[\Gamma\sigma^*] \vdash [C_i\sigma^*] \mid i \in \{1,\ldots,n\}\} \subseteq \cup [\Gamma \vdash C_i]$.

( ) $\vdash$

$$\frac{\Gamma, C_i \vdash C}{\Gamma, (C_1,\ldots,C_n) \vdash C} \qquad \frac{[\Gamma\sigma^*], [C_i\sigma^*] \vdash [C\sigma^*]}{[\Gamma\sigma^*], ([C_i\sigma^*])_{i \in \{1,\ldots,n\}} \vdash [C\sigma^*]}$$

for some i. From the definition of [ ], it is clear that $([\Gamma\sigma^*], [C_i\sigma^*] \vdash [C\sigma^*]) \in [\Gamma, C_i \vdash C]$, for every $i \in \{1,\ldots,n\}$.

$\vdash \Rightarrow$

$$\frac{\Gamma, C \vdash C'}{\Gamma \vdash C \Rightarrow C'} \qquad \frac{[\Gamma\sigma^*], [C\sigma^*] \vdash [C'\sigma^*]}{[\Gamma\sigma^*] \vdash [C\sigma^*] \rightarrow [C'\sigma^*]}$$

From the definition of [ ], it is clear that $([\Gamma\sigma^*], [C\sigma^*] \vdash [C'\sigma^*]) \in [\Gamma, C \vdash C']$.

$\Rightarrow \vdash$

$$\frac{\Gamma \text{ - } C' \quad \Gamma, C'' \text{ - } C}{\Gamma, C' \Rightarrow C'' \text{ - } C} \qquad \frac{[\Gamma\sigma^*] \text{ - } [C'\sigma^*] \quad [\Gamma\sigma^*], [C''\sigma^*] \text{ - } [C\sigma^*]}{[\Gamma\sigma^*], [C'\sigma^*] \rightarrow [C''\sigma^*] \text{ - } [C\sigma^*]}$$

From the definition of [ ], it is clear that $([\Gamma\sigma^*] \text{ - } [C'\sigma^*]) \in [\Gamma \text{ - } C']$ and that $([\Gamma\sigma^*], [C''\sigma^*] \text{ - } [C\sigma^*]) \in [\Gamma, C'' \text{ - } C]$.

-Π

$$\frac{\Gamma \text{ - } C(X^*)}{\Gamma \text{ -} \Pi x\ C(x)} \qquad \frac{\{[\Gamma\sigma^*] \text{ - } [C(i)\sigma^*] \mid i \in ran(x)\}}{[\Gamma\sigma^*] \text{ -} ([C(i)\sigma^*])_{i \in ran(x)}}$$

For each of the infinitary premises, $[\Gamma\sigma^*] \text{ - } [C(i)\sigma^*]$, let $\tau_i^* = \{X^*/i\}$. Since X^* does not occur in Γ or C, $[\Gamma\sigma^*] \text{ - } [C(i)\sigma^*] = [\Gamma\tau_i^*\sigma^*] \text{ - } [C(X^*)\tau_i^*\sigma^*]$. Clearly, $\tau_i^*\sigma^*$ is a ground simplifying substitution for $\Gamma \text{ - } C(X^*)$, since σ^* is a ground simplifying substitution for both Γ and C, and τ_i^* is simple. By the definition of [ ], $([\Gamma\tau_i^*\sigma^*] \text{ - } [C(X^*)\tau_i^*\sigma^*]) \in [\Gamma \text{ - } C(X^*)]$. Set inclusion follows.

Π -

$$\frac{\Gamma, C(t) \text{ - } C'}{\Gamma, \Pi x\ C(x) \text{ - } C'} \qquad \frac{[\Gamma\tau^*\sigma^*], [C(t)\tau^*\sigma^*] \text{ - } [C'\tau^*\sigma^*]}{[\Gamma\sigma^*], ([C(i)\sigma^*])_{i \in ran(x)} \text{ - } [C'\sigma^*]}$$

for some $t \in \mathcal{RAN}(x)$. Let τ^* be an arbitrary parameter substitution simplifying and nonredundant for all parameters in t that do not occur in Γ, C or C′. Then $C(t)\tau^*\sigma^*$ is an ground simple term, $\Gamma = \Gamma\tau^*$ and $C' = C'\tau^*$, so the premises of the infinitary rule have the proper form. Clearly, $([\Gamma\tau^*\sigma^*], [C(t)\tau^*\sigma^*] \text{ - } [C'\tau^*\sigma^*]) \in [\Gamma, C(t) \text{ - } C']$. Set inclusion follows.

-D

$$\frac{\Gamma \text{ - } B\sigma}{\Gamma \text{ - } a} \qquad \frac{[\Gamma\sigma^*] \text{ - } [B\sigma\sigma^*]}{[\Gamma\sigma^*] \text{ - } [a\sigma^*]}$$

where $(H \Leftarrow B) \in P$, σ grounds B, and $a = H\sigma$, thus $[a\sigma^*] = [H\sigma\sigma^*]$. By the definition of [ ], $([H\tau] = [B\tau]) \in [P]$, for all grounding and simple τ. Clearly $\sigma\sigma^*$ is a grounding and simple substitution for H and B, thus $([H\sigma\sigma^*] = [B\sigma\sigma^*]) \in [P]$, so $([a\sigma^*] = [B\sigma\sigma^*]) \in [P]$ and $[B\sigma\sigma^*] \in D(a\sigma^*)$, showing that the infinitary rule has the proper form. Clearly $([\Gamma\sigma^*] \text{ - } [B\sigma\sigma^*]) \in [\Gamma \text{ - } B\sigma]$. Set inclusion follows.

D -

$$\frac{\{[\Gamma\tau^*], B\tau^* \text{ - } C\tau^* \mid \tau^* \in \Sigma^*(a,H),\ (H \Leftarrow B) \in P^*\}}{\Gamma, a \text{ - } C} \qquad \frac{\{[\Gamma\sigma^*], A \text{ - } [C\sigma^*] \mid A \in \mathcal{D}([a\sigma^*])\}}{[\Gamma\sigma^*], [a\sigma^*] \text{ - } [C\sigma^*]}$$

...where each $\Sigma^*(a,H)$ is ground and finite. Take an arbitrary $A \in \mathcal{D}([a\sigma^*])$. Using the definitions of $\mathcal{D}$ and [ ], we get $([a\sigma^*] = A) \in [P]$, i.e. $([a\sigma^*] = A) \in [K]$, for some clause $K \in P$. Thus there exist some H′, B′ and a nonredundant grounding simple ρ for H′ and B′, such that $(H' \Leftarrow B') = K$ and $([H'\rho] = [B'\rho]) = ([a\sigma^*] = A)$. Since the parameter transform of K is simply a renaming of variables to parameters, there are H, B and a nonredundant ground simplifying ρ^*

for H and B, such that $(H \Leftarrow B)=K^*$ and $([H\rho^*]=[B\rho^*])=([a\sigma^*]=A)$. From this we get $(H \Leftarrow B) \in P^*$, $[H\rho^*]=[a\sigma^*]$ and $[B\rho^*]=A$.

Consider that ρ^* is nonredundant for H and B. Since H and B may have no parameters in common with Γ, a or C, we have $\Gamma=\Gamma\rho^*$, $a=a\rho^*$ and $C=C\rho^*$. Since ρ^* is simplifying, we have $H\rho^*=H\rho^*\sigma^*$ and $B\rho^*=B\rho^*\sigma^*$. By the definition of [ ], $H\rho^*=a\sigma^*$, thus $H\rho^*\sigma^*=a\rho^*\sigma^*$, so $\rho^*\sigma^*$ is a unifier of a and H. By the definition of a CSU^*, there is a $\tau^* \in \Sigma^*(a,H)$, such that $\rho^*\sigma^*=\tau^*\upsilon^*$, for some υ^*.

Now τ^*, H and B fulfil the condition in the set constructor of the premise of the finitary rule. To establish set inclusion, we must show that $([\Gamma\sigma^*],A - [C\sigma^*]) \in [\Gamma\tau^*, B\tau^* - C\tau^*]$. By the equalities in the previous paragraphs, this can be rewritten, first to $([\Gamma\rho^*\sigma^*], [B\rho^*\sigma^*] - [C\rho^*\sigma^*]) \in [\Gamma\tau^*, B\tau^* - C\tau^*]$, then to $([\Gamma\tau^*\upsilon^*], [B\tau^*\upsilon^*] - [C\tau^*\upsilon^*]) \in [\Gamma\tau^*, B\tau^* - C\tau^*]$. Since σ^* is ground and simplifying for Γ and C, and ρ^* is ground and simplifying for B, $\rho^*\sigma^*=\tau^*\upsilon^*$ is ground and simplifying for Γ, B and C. Thus υ^* is ground and simplifying for $[\Gamma\tau^*, B\tau^* - C\tau^*]$. By the definition of [ ], this proves the membership.

Contraction

$$\frac{\Gamma, C, C - C'}{\Gamma, C - C'} \qquad \frac{[\Gamma\sigma^*], [C\sigma^*], [C\sigma^*] - [C'\sigma^*]}{[\Gamma\sigma^*], [C\sigma^*] - [C'\sigma^*]}$$

From the definition of [ ], it is clear that $([\Gamma\sigma^*], [C\sigma^*], [C\sigma^*] - [C'\sigma^*]) \in [\Gamma, C, C - C']$.

Weakening

$$\frac{\Gamma - C'}{\Gamma, C - C'} \qquad \frac{[\Gamma\sigma^*] - [C'\sigma^*]}{[\Gamma\sigma^*], [C\sigma^*] - [C'\sigma^*]}$$

From the definition of [ ], it is clear that $([\Gamma\sigma^*] - [C'\sigma^*]) \in [\Gamma - C']$.

Specialisation

$$\frac{\Gamma - C}{\Gamma\tau^* - C\tau^*} \qquad [\Gamma\tau^*\sigma^*] - [C\tau^*\sigma^*] \qquad \text{(specialisation)}$$

This rule does not get translated into any inference rule of the infinitary calculus, as the mapping of the conclusion is already a subset of the mapping of the premise. From the definition of [ ], it is clear that $([\Gamma\tau^*\sigma^*] - [C\tau^*\sigma^*]) \in [\Gamma - C]$.

Induction will be covered in the next section.

6. SOUNDNESS OF INDUCTION

The induction schema will be motivated differently from the proper inference rules. Instead of providing a corresponding finitary inference step, we will show that the induction schema guarantees the existence of a finitary derivation without the induction, for each term in the set over which induction is done. The new derivation may itself contain other inductions, but they can all be removed in the same manner. As in the previous section, we are only interested in the variable-free finitary calculus.

To show soundness of induction, we need the following lemma:

LEMMA 6.1 Induction instantiation lemma
Given a derivation in the finitary calculus of the sequent S from the sequents $S_1,\ldots, S_n$

$$\begin{array}{c} S_1 \quad \cdots \quad S_n \\ \vdots \\ S \end{array}$$

and a ground parameter substitution, τ^*, such that the the members of $\mathcal{DOM}(\tau^*)$ and $\mathcal{PARM}(\tau^*)$ are all induction parameters of the derivation. Then there is a derivation

$$\begin{array}{c} S_1\tau^* \quad \cdots \quad S_n\tau^* \\ \vdots \\ S\tau^* \end{array}$$

PROOF
We will show this by constructing the new derivation inductively. Consider the last step of the derivation. Assume that there is a translation of the derivations of each of the premises of that step. The translation of the entire derivation will be a translation of the final step, with all the translated premise derivations above. Possibly the translation of the final step will have fewer premises than the original step. The superfluous premise derivations will then simply be omitted.

We will give a translation of the final step, divided into cases depending on the rule that step is an instance of. First note that applying the substitution τ^* to the premises and conclusion of an inference step does not change the structure of that step. For the inference rules whose correctness depends solely on the structure of the premises and conclusions (axiom, $-() () -$, $-\Rightarrow , \Rightarrow -$, contraction and weakening), i.e. those without side conditions, the translation is obtained simply by applying τ^* to the conclusion and all premises. The other rules will be treated individually.

For each of the remaining inference rules, the result of applying τ^* to the premises and conclusion will be shown as an inference step. This step will not have the form of a correct inference rule instance. We will show in each case either that the step can be put in the form of a correct rule instance (i.e. that the premises and conclusions are equal to premises and conclusions of the correct form), or we will provide as an explicit translation a correct subderivation of one or more inference steps, with the same premises (or a subset) and the same conclusion.

-Π

$$\frac{\Gamma\tau^* \text{ - } C(X^*)\tau^*}{\Gamma\tau^* \text{ - } (\Pi x\ C(x))\tau^*}$$

Since x is a bound variable, $(\Pi x\ C(x))\tau^* = \Pi x\ C\tau^*(x)$. As X^* must not be an induction parameter, $X^* = X^*\tau^*$. Thus we have $C(X^*)\tau^* = C\tau^*(X^*\tau^*) = C\tau^*(X^*)$. Applying these equalities, we get our translation

$$\frac{\Gamma\tau^* \text{ - } C\tau^*(X^*)}{\Gamma\tau^* \text{ -} \Pi x\ C\tau^*(x)}$$

which is a correct instance of the -Π rule. Since $\mathcal{PARM}(\tau^*)$ contains only induction parameters, $\mathcal{PARM}(\Gamma\tau^*)$ and $\mathcal{PARM}(C\tau^*)$ contain no non-induction parameters other than those in $\mathcal{PARM}(\Gamma)$ and $\mathcal{PARM}(C)$. By the side condition on the original inference step, $X^* \notin \mathcal{PARM}(\Gamma)$ and $X^* \notin \mathcal{PARM}(C)$, Thus $X^* \notin \mathcal{PARM}(\Gamma\tau^*)$ and $X^* \notin \mathcal{PARM}(C\tau^*)$, satisfying the side condition on the translation.

Π -

$$\frac{\Gamma\tau^*, C(t)\tau^* \text{ - } C'\tau^*}{\Gamma\tau^*, (\Pi x\ C(x))\tau^* \text{ - } C'\tau^*}$$

Now, $C(t)\tau^* = C\tau^*(t\tau^*)$ and, since x is a bound variable, $(\Pi x\ C(x))\tau^* = \Pi x\ C\tau^*(x)$. Applying these equalities, we get our translation

$$\frac{\Gamma\tau^*, C\tau^*(t\tau^*) \text{ - } C'\tau^*}{\Gamma\tau^*, \Pi x\ C\tau^*(x) \text{ - } C'\tau^*}$$

which is a correct instance of the Π - rule. Since $t \in \mathcal{RAN}(x) \rightarrow t\tau^* \in \mathcal{RAN}(x)$, the range condition is fulfilled.

-D

$$\frac{\Gamma\tau^* \text{ - } B\sigma\tau^*}{\Gamma\tau^* \text{ - } a\tau^*} \quad (H \Leftarrow B) \in P,\ a = H\sigma$$

From $a = H\sigma$, we get $a\tau^* = H\sigma\tau^*$. Let $\sigma' = (\sigma\tau^*) \setminus \mathcal{DOM}(\sigma)$. Since $\mathcal{PARM}(H) = \mathcal{PARM}(B) = \varnothing$, we have $H\sigma\tau^* = H\sigma'$ and $B\sigma\tau^* = B\sigma'$. Applying these equalities, we get our translation

$$\frac{\Gamma\tau^* \text{ - } B\sigma'}{\Gamma\tau^* \text{ - } a\tau^*} \quad (H \Leftarrow B) \in P,\ a\tau^* = H\sigma'$$

which is a correct instance of the -D rule.

D -

$$\frac{\left\{\Gamma\sigma^*\tau^*, B\sigma^*\tau^* \text{ - } C\sigma^*\tau^* \mid \sigma^* \in \Sigma^*(a,H),\ (H \Leftarrow B) \in P^*\right\}}{\Gamma\tau^*, a\tau^* \text{ - } C\tau^*}$$

The translation will be the derivation

$$\frac{\left\{\dfrac{\Gamma\sigma^*\tau^*, B\sigma^*\tau^* - C\sigma^*\tau^*}{\Gamma\tau^*\rho^*, B\rho^* - C\tau^*\rho^*} \;\middle|\; \rho^* \in \Sigma^*(a\tau^*, H),\ (H \Leftarrow B) \in P^*\right\}}{\Gamma\tau^*, a\tau^* - C\tau^*}$$

where each σ^* is defined to be some member of $\Sigma^*(a,H)$, such that $\sigma^*\nu^* = \tau^*\rho^*$, for some ν^*. Since $\Sigma^*(a,H)$ is a CSU*, this is well-defined if $\tau^*\rho^*$ is a unifier of a and H. It is clear from this definition that if σ^* is well-defined, each premise of the translation is also a premise of the original instantiated step above.

The last step clearly has the form of an instance of the D - rule. We must show that the side conditions are fulfilled. First, we must have $\mathcal{PARM}(P^*) \cap \mathcal{PARM}(\Gamma\tau^*, a\tau^* - C\tau^*) = \varnothing$. By the conditions of the induction instantiation lemma, $\mathcal{PARM}(\tau^*)$ may contain only induction parameters. Since the condition of the original rule $\mathcal{PARM}(P^*)$ may not contain induction parameters, $\mathcal{PARM}(P^*) \cap \mathcal{PARM}(\tau^*) = \varnothing$. By the condition on the original rule, we know that $\mathcal{PARM}(P^*) \cap \mathcal{PARM}(\Gamma, a - C) = \varnothing$, thus the condition is satisfied. The condition that P^* may contain no induction parameters carries over from the original step. Since we are dealing with the variable-free calculus here, the side condition that $\mathcal{C}^*(a\tau^*)\varphi \cong^* \mathcal{C}^*(a\tau^*\varphi)$, for all φ, is trivially satisfied.

Strictly speaking, we should also show that ρ^* is finite. Since the translation will always have a finite number of premises - which is what really matters for the finitaryness - we dispense with showing this condition.

We must also show that the step inside the set constructor is an instance of the specialisation rule. Take some ρ^* and $H \Leftarrow B$, and consider the corresponding $\Gamma\tau^*\rho^*, B\rho^* - C\tau^*\rho^*$. Since $\rho^* \in \Sigma^*(a\tau^*, H)$, $a\tau^*\rho^* = H\rho^*$. By the condition on the original step, there are no induction parameters in $\mathcal{PARM}(H)$ or $\mathcal{PARM}(B)$, so we get $H = H\tau^*$ and $B = B\tau^*$. Using this, we get $a\tau^*\rho^* = H\tau^*\rho^*$, i.e. $\tau^*\rho^*$ is a unifier of a and H. This means that σ^* is well-defined. Note that since σ^* is a member of a CSU* , $\sigma^* = \sigma^*\sigma^*$. Now, $\tau^*\rho^* = \sigma^*\nu^* = \sigma^*\sigma^*\nu^* = \sigma^*\tau^*\rho^*$. Applying these equalities to the step inside the set constructor, we obtain

$$\frac{\Gamma\sigma^*\tau^*, B\sigma^*\tau^* - C\sigma^*\tau^*}{\Gamma\sigma^*\tau^*\rho^*, B\sigma^*\tau^*\rho^* - C\sigma^*\tau^*\rho^*}$$

which is indeed an instance of the specialisation rule.

Specialisation

$$\frac{\Gamma\tau^* - C\tau^*}{\Gamma\sigma^*\tau^* - C\sigma^*\tau^*}$$

Let $\rho^* = (\sigma^*\tau^*) \backslash \mathcal{DOM}(\sigma^*)$. Let X^* be an induction parameter. Since there are no induction parameters in $\mathcal{DOM}(\sigma^*)$ – and thus not in $\mathcal{DOM}(\rho^*)$ either, we have $X^*(\sigma^*\tau^*) = X^*\tau^* = X^*(\tau^*\rho^*)$. Let Y^* be an non-induction parameter. Since $\mathcal{DOM}(\tau^*)$ contains only induction

parameters, we have $Y^*(\sigma^*\tau^*) = Y^*\rho^* = Y^*(\tau^*\rho^*)$. By the substitution equality lemma, we get $\sigma^*\tau^*=\tau^*\rho^*$. Applying this equality to the step above, we get

$$\frac{\Gamma\tau^* \text{ - } C\tau^*}{\Gamma\tau^*\rho^* \text{ - } C\tau^*\rho^*}$$

which is a correct instance of the specialisation rule.

Induction

$$\frac{\left\{\begin{array}{c|c}\left\{S(C)\tau^* \mid C\in\langle B\rangle\right\} & \\ \vdots & (H\Leftarrow B)\in T^* \\ S(H)\tau^* & \end{array}\right\}}{S(X^*)\tau^*}$$

The translation will be the derivation

$$\frac{\left\{\begin{array}{c|c}\left\{S\tau^*(C) \mid C\in\langle B\rangle\right\} & \\ \vdots & (H\Leftarrow B)\in T^* \\ S\tau^*(H) & \end{array}\right\}}{\dfrac{S\tau^*(Y^*)}{S(X^*)\tau^*}}$$

where Y^* is an arbitrary non-induction parameter such that $Y^*\notin \mathcal{PARM}(S)$ and $Y^*\notin \mathcal{PARM}(\tau^*)$. The first step is clearly an instance of the induction schema. Since the parameters of $\mathcal{PARM}(H)$ and $\mathcal{PARM}(\langle B\rangle)$ are not induction parameters of the derivation being translated (possibly they are induction parameters of the hypothetical derivations occurring as premises to the induction schema), we have $H=H\tau^*$ and $C=C\tau^*$, for every H and $C\in\langle B\rangle$. Thus $S(H)\tau^*=S\tau^*(H\tau^*)=S\tau^*(H)$ and $S(C)\tau^*=S\tau^*(C\tau^*)=S\tau^*(C)$, so the hypothetical derivations of our translation are the translations of the hypothetical derivations of the original induction.

To show that the last step is an instance of the specialisation rule, let $\sigma^*=\{Y^*/X^*\tau^*\}$. Since Y^* is not an induction parameter, we have $Y^*= Y^*\tau^*$. Since $Y^*\notin \mathcal{PARM}(S)$ and $Y^*\notin \mathcal{PARM}(\tau^*)$, we have $S=S\sigma^*$ and $S\tau^* = S\tau^*\sigma^*$. Now, $S\tau^*(Y^*) = S\tau^*(Y^*\tau^*) = S(Y^*)\tau^*$ and $S(X^*)\tau^*=S\tau^*(X^*\tau^*)=S\tau^*(Y^*\sigma^*)=S\tau^*\sigma^*(Y^*\sigma^*)=S\tau^*\sigma^*(Y^*\tau^*\sigma^*)=S(Y^*)\tau^*\sigma^*$. Applying these equalities to the last step, we obtain

$$\frac{S(Y^*)\tau^*}{S(Y^*)\tau^*\sigma^*}$$

which is an instance of the specialisation rule.

END OF PROOF. ®

Consider again the inference schema for induction:

$$\frac{\left\langle \begin{array}{c} \{S(C) \mid C \in \langle B \rangle\} \\ \vdots \\ S(H) \end{array} \;\middle|\; (H \Leftarrow B) \in T^* \right\rangle}{S(X^*)} \qquad \text{(induction)}$$

To establish the soundness of this schema, we must show that it implies the derivability of all the infinitary sequents in $[S(X^*)]$. Note that

$$\left[S(X^*)\right] = \bigcup_{t \in ran(X^*)} [S(t)]$$

so if we can show that there exists a finitary derivation without induction of each S(t), where $t \in \mathit{Def}(T)$, we are done. We will inductively construct a derivation of S(t) for all ground terms t, such that $\vdash_T t$ and $\mathit{PARM}(t)$ contains only induction parameters. By the definition of $\mathit{Def}(T)$, $t \in \mathit{Def}(T)$ iff $\vdash_T t$ for simple ground t, so it would be enough for us to only consider simple ground terms. That would not give us sufficiently strong induction hypotheses to carry out the construction, however, so will will consider general ground terms.

Consider the derivation of $\vdash_T t$, where t is some ground term. According to the restrictions on T, there is no induction in this derivation. Clearly then, the last inference step must be an instance of the $\vdash$D rule with some grounding σ and $(H' \Leftarrow B') \in T$, such that $t = H'\sigma$. Call the premise of this rule, $\vdash_T C$. Due to the restrictions on T, if C is not again a term, it must be a sequence of conditions or a Π-condition. In the latter cases, the $\vdash()$ or $\vdash\Pi$ rule must be used. Repeating this argument, we find that the derivation of $\vdash_T t$, must have the form

$$\frac{\begin{array}{ccc} \vdots & & \vdots \\ \vdash_T t_1 & \cdots & \vdash_T t_n \end{array}}{\begin{array}{c} \vdots \\ \vdash_T t \end{array}}$$

where the t_i are precisely the terms in $\langle B'\sigma \rangle$.

Now, instead of the clause $H' \Leftarrow B'$ of T and the variable substitution σ, consider the parameter transform, $H \Leftarrow B$, of $H' \Leftarrow B'$. Since the parameter transform is simply a renaming of variables to parameters, there is a nonredundant ground simplifying σ^* for H and B, such that $H\sigma^* = H'\sigma$ and $B\sigma^* = B'\sigma$. Take the hypothetical derivation of the induction schema, corresponding to $H \Leftarrow B$. We will have $t = H\sigma^*$. Without loss of generality, we can assume that σ^* is nonredundant for $H \Leftarrow B$. Since $\mathit{PARM}(H \Leftarrow B)$ contains only induction parameters, so will $\mathit{DOM}(\sigma^*)$. Since $\mathit{PARM}(t)$ contains only induction parameters, so will $\mathit{PARM}(\sigma^*)$, and the induction instantiation lemma is applicable. Applying σ^* to the hypothetical derivation, we obtain part of a derivation of S(t):

$$\frac{S(t_1) \cdots S(t_n)}{\begin{array}{c} \vdots \\ S(t) \end{array}}$$

where the $S(t_i)$ are precisely the sequents of $\{S(C) \mid C \in \langle B'\sigma \rangle\}$. Repeating this argument inductively, we obtain derivations of each $S(t_i)$. Putting these derivations together, we obtain a complete derivation of S(t), without the induction. As the derivation of $\vdash_T t$ has a finite number of steps, this procedure will terminate.

EXAMPLE 6.2

Consider the derivation of example 4.3. Taking t above to be a.b.nil, we will show that a derivation of append(a.b.nil,nil,a.b.nil) can be obtained by substituting in the hypothetical derivations of the induction. The parameter transform, $LIST^*$, of the definition of lists is :

$$\text{nil} \Leftarrow$$
$$U^*.L^* \Leftarrow L^*$$

We show $\vdash_{LIST}$ a.b.nil using the following derivation:

$$\dfrac{\dfrac{\dfrac{}{\vdash \text{nil}}\,(\vdash P)}{\vdash \text{b.nil}}\,(\vdash P)}{\vdash \text{a.b.nil}}\,(\vdash P)$$

The first step uses the first clause of the definition, with the corresponding parameter substitution σ^* being $\varnothing$. The middle step uses the second clause of the definition, with the parameter substitution σ^* being $\{U^*/b, L^*/nil\}$.The last step again uses the second clause of the definition, with the parameter substitution σ^* being $\{U^*/a, L^*/b.nil\}$.

The hypothetical derivations of example 4.3 were

$$\dfrac{}{\vdash \text{append(nil,nil,nil)}}\,(\vdash P) \qquad \text{and} \qquad \dfrac{\vdash \text{append}(L^*,\text{nil},L^*)}{\vdash \text{append}(U^*.L^*,\text{nil},U^*.L^*)}\,(\vdash P)$$

Applying each of the σ^*´s to the hypothetical derivation corresponding to the clause in question, we obtain the following three derivations

$$\dfrac{}{\vdash \text{append(nil,nil,nil)}}\,(\vdash P) \quad \dfrac{\vdash \text{append(nil,nil,nil)}}{\vdash \text{append(b.nil,nil,b.nil)}}\,(\vdash P) \quad \dfrac{\vdash \text{append(b.nil,nil,b.nil)}}{\vdash \text{append(a.b.nil,nil,a.b.nil)}}\,(\vdash P)$$

It is easy to see that they can be combined to give a derivation of append(a.b.nil,nil,a.b.nil), as required. ®

We have proved that there exists a finitary derivation of S(t), where the final induction schema has been removed, however the hypothetical derivations of that induction may themselves contain inductions which will have been incorporated in the constructed derivation. Each of these inductions may be removed in the same manner, but we need some proof that this procedure terminates.

We will define the **depth of nested inductions** as follows. For a derivation, the depth of nested inductions is the maximum of the depth of nested inductions of any inductions in that derivation. For an induction schema instance, the depth of nested inductions is one more than the maximum depth of nested inductions of its hypothetical derivations.

Since the constructed derivation of S(t) is finitary, it contains a finite number of inductions. Each of these must have a depth of nested inductions that is less than the original induction. By repeating the procedure above a finite number of times, these inductions can also be removing, possibly leaving inductions with still lower depth. Eventually the depth of nested inductions will be reduced to zero, meaning that no inductions remain. This completes the proof of the soundness of induction.

7. SOUNDNESS OF THE FINITARY CALCULUS WITH VARIABLES

So far, we have concerned ourselves only with complete, unchanging, derivations. However, the most common applications of formal proof systems in computer science deal with the construction of derivations. From a completely manual proof editing system, to a fully automatic theorem prover such as a Prolog implementation, the main concern is incrementally adding steps to a derivation.

Such proof constructions are usually done in a goal-directed fashion, starting with the sequent to be proved and applying inference rules in reverse until a complete proof has been found. When using the rules in this way, decisions must sometimes be made on which particular rule instance to use. This is most clear with the Π - rule, where the Π-bound variable is replaced with some arbitrary term. It does not matter to the Π - rule which term is chosen, but for steps being made later, a particular choice of term may be essential. The same situation may arise with the -D rule. The usual solution to this problem is to introduce placeholders, logical variables that are introduced instead of the arbitrary term and later substituted for the appropriate term as required.

For this purpose, we have the version of the finitary calculus where variables are permitted in derivations. Each such variable will represent an undetermined term and can be substituted (in the entire derivation) for some nonvariable term when necessary for the application of an inference rule.

A derivation with variables should be regarded as representing a set of potential derivations in the variable-free (and thus the infinitary) calculus. This should be constrasted to the use of parameters, which are intended to represent a set of **actual** - not potential - derivations. If every variable in the derivation is substituted by a ground term, only one potential derivation remains.

For the calculus with variables to be meaningful, **instantiation**, i.e. the application of a variable substitution, must be a sound operation. Soundness in this context means that the result of applying a variable substitution to a derivation must result in another correct derivation. Putting it another way, the set of derivations must be closed under instantiation. It turns out that

we cannot permit arbitrary substitutions, but a simple restriction must be applied. After defining this restriction, we prove that the set of derivations are closed under instantiation.

We call an inference step such that a parameter X^* occurs in some premise, but not in the conclusion, an **eliminating step** for X^*. An instance of the specialisation rule is also considered an eliminating step for every parameter in $\mathcal{DOM}(\sigma^*)$, where σ^* is the specialising substitution. If a variable Y occurs in the conclusion of a step eliminating X^*, Y is said to **predate** X^*. Consider a substitution θ. If, for any Y, there is some $X^* \in \mathcal{PARM}(Y\theta)$ such that X^* is predated by Y, we say that θ is an **inadmissible instantiation**, otherwise θ is **admissible.**

EXAMPLE 7.1
Consider the derivation

$$\dfrac{\dfrac{q(Y,B)\ \text{-}p(X^*)}{\Pi b\ q(Y,b)\ \text{-}p(X^*)}\ (\Pi\ \text{-})}{\Pi b\ q(Y,b)\ \text{-}\Pi x\ p(x)}\ (\text{-}\Pi)$$

Here Y predates X^*, but B does not predate X^*. The substitution $\{B/X^*\}$ is admissible, while the substitution $\{Y/X^*\}$ is inadmissible. ®

EXAMPLE 7.2
The following example shows why the admissibility condition is important. Consider again the definition FOL and the derivation

$$\dfrac{\dfrac{\dfrac{\text{-}X=Y^*}{\text{-}\Pi y\ X=y}\ (\text{-}\Pi)}{\text{-}\forall y\ X=y}\ (\text{-}P)}{\text{-}\exists x \forall y\ x=y}\ (\text{-}P)$$

If we applied the substitution $\{X/Y^*\}$ to this derivation, the first sequent would become $\text{-}Y^*=Y^*$, which would hold by the -D rule using the clause defining equality. However, this would produce an invalid derivation, since the side condition on the -Π step that Y^* must not occur in the conclusion of the step would no longer be fulfilled (which is fortunate, since the conclusion of the derivation is wrong!). Since X predates Y^*, that substitution would be inadmissible and the incorrect derivation could not be obtained. ®

In section 8, we will discuss the question of how to compute admissible substitutions.

As instantiation of variables is a similar operation to the instantiation of parameters, done in the proof of the parameter instantiation lemma, this proof will have a similar structure as the proof of that lemma.

Given a derivation in the finite calculus with variables of the sequent S from the sequents $S_1,\ldots, S_n$

$$\begin{array}{c} S_1 \quad \cdots \quad S_n \\ \vdots \\ S \end{array}$$

and a substitution, θ. If θ is an admissible instantiation, then there is a derivation

$$\begin{array}{c} S_1\theta \quad \cdots \quad S_n\theta \\ \vdots \\ S\theta \end{array}$$

with the same steps as the original one. We will prove this by showing, for each inference rule, either that a rule instance remains correct when variables are substituted, or at least that the conclusion of the step remains derivable from the premises of the step using some small derivation.

For the inference rules whose correctness depends solely on the structure of the premises and conclusions (axiom, $\text{-}()\ ()\ \text{-}$, $\text{-}\Rightarrow$, $\Rightarrow\text{-}$, contraction and weakening), i.e. those without side conditions, is is obvious that the instantiated step will be correct. For each of the remaining inference rules, the result of applying θ to the premises and conclusion will be shown as an inference step. This step will not be in the form of an inference rule of the finitary calculus. We will show in each case either that the step can be put in the form of a correct rule instance (i.e. that the premises and conclusions are equal to premises and conclusions of the correct form) or we will provide as an explicit translation a correct subderivation of one or more inference steps, with the same premises (or a subset) and the same conclusion.

$\text{-}\Pi$

$$\frac{\Gamma\theta \text{ - } C(X^*)\theta}{\Gamma\theta \text{ - } (\Pi x\ C(x))\theta}$$

Since x is a bound variable, $(\Pi x\ C(x))\theta = \Pi x\ C\theta(x)$. As X^* is a parameter, $X^* = X^*\theta$. Thus we have $C(X^*)\theta = C\theta(X^*\theta) = C\theta(X^*)$. Applying these equalities, we get

$$\frac{\Gamma\theta \text{ - } C\theta(X^*)}{\Gamma\theta \text{ -} \Pi x\ C\theta(x)}$$

which is a correct instance of the $\text{-}\Pi$ rule. According to the side condition on the original inference step, $X^* \notin \mathcal{PARM}(\Gamma)$ and $X^* \notin \mathcal{PARM}(C)$. Since θ is an admissible substitution, it follows that $X^* \notin \mathcal{PARM}(\Gamma\theta)$ and $X^* \notin \mathcal{PARM}(C\theta)$, fulfilling the side condition of the instantiated inference step.

$\Pi\ \text{-}$

$$\frac{\Gamma\theta, C(t)\theta \text{ - } C'\theta}{\Gamma\theta, (\Pi x\ C(x))\theta \text{ - } C'\theta}$$

Now, $C(t)\theta = C\theta(t\theta)$ and, since x is a bound variable, $(\Pi x\ C(x))\theta = \Pi x\ C\theta(x)$. Applying these equalities, we get

$$\frac{\Gamma\theta, C\theta(t\theta) \text{ - } C'\theta}{\Gamma\theta, \Pi x\ C\theta(x) \text{ - } C'\theta}$$

which is a correct instance of the Π - rule. Since $t \in \mathcal{RAN}(x) \rightarrow t\theta \in \mathcal{RAN}(x)$, the range condition is fulfilled.

-D

$$\frac{\Gamma\theta \text{ - } B\sigma\theta}{\Gamma\theta \text{ - } a\theta}\ (H \Leftarrow B) \in P,\ a = H\sigma$$

From $a = H\sigma$, we get $a\theta = H\sigma\theta$. Using this as the side condition of the step

$$\frac{\Gamma\theta \text{ - } B\sigma\theta}{\Gamma\theta \text{ - } a\theta}\ (H \Leftarrow B) \in P,\ a\theta = H\sigma\theta$$

we see that it is a correct instance of the -D rule.

D -

$$\frac{\left\{\Gamma\sigma^*\theta, B\sigma^*\theta \text{ - } C\sigma^*\theta \mid \sigma^* \in \Sigma^*(a,H),\ (H \Leftarrow B) \in P^*\right\}}{\Gamma\theta, a\theta \text{ - } C\theta}$$

The translation will be the derivation

$$\frac{\left\{\dfrac{\Gamma\sigma^*\theta, B\sigma^*\theta \text{ - } C\sigma^*\theta}{\Gamma\theta\rho^*, B\rho^* \text{ - } C\theta\rho^*} \;\middle|\; \rho^* \in \Sigma^*(a\theta,H),\ (H \Leftarrow B) \in P^*\right\}}{\Gamma\theta, a\theta \text{ - } C\theta}$$

where the B' and σ^* of every premise are as defined below, and the inner inference is specialisation.

The last step clearly has the form of an instance of the D - rule. We must show that $\mathcal{PARM}(P^*) \cap \mathcal{PARM}(\Gamma\theta, a\theta \text{ - } C\theta) = \varnothing$. Since by the condition on the original rule, $\mathcal{PARM}(P^*) \cap \mathcal{PARM}(\Gamma, a \text{ - } C) = \varnothing$, it suffices to show that $\mathcal{PARM}(P^*) \cap \mathcal{PARM}(\theta) = \varnothing$. As θ is an admissible substitution, $\mathcal{PARM}(P^*) \cap \mathcal{PARM}(\theta)$ can not contain any parameters that are among the parameters of some premise, but not already in $\mathcal{PARM}(\Gamma, a \text{ - } C)$. Should $\mathcal{PARM}(P^*) \cap \mathcal{PARM}(\theta)$ contain any parameter in P^* that is not in a premise, that parameter could simply be renamed in P^* to avoid the conflict. The condition that P^* may contain no induction parameters carries over from the original step.

From the side condition that $\mathcal{C}^*(a)\varphi \cong^* \mathcal{C}^*(a\varphi)$, for all φ, we have, in particular $\mathcal{C}^*(a)\theta\tau \cong^* \mathcal{C}^*(a\theta\tau)$. Note that the particular parameters occurring in $\mathcal{C}^*(a)\theta$ and $\mathcal{C}^*(a\theta)$ depend on the parameters in P^*. As the parameters in P^* can be renamed arbitrarily, we can choose them so that lemma 3.6 is applicable. Then, from $\mathcal{C}^*(a)\varphi \cong^* \mathcal{C}^*(a\varphi)$, with the present θ, we get $\mathcal{C}^*(a)\theta\tau \cong^* \mathcal{C}^*(a\theta)\tau$. By combining $\cong^*$-relations, we get $\mathcal{C}^*(a\theta)\tau \cong^* \mathcal{C}^*(a\theta\tau)$, for all τ, which is the remaining side condition for the last step.

Strictly speaking, we should also show that ρ^* is finite. Since the translation will always have a finite number of premises - which is what really matters for the finitaryness - we dispense with showing this condition.

Having confirmed the side conditions of the last step, we must show that the step inside the set constructor is an instance of the specialisation rule. From the condition $\mathcal{C}^*(a)\theta \cong^* \mathcal{C}^*(a\theta)$, for all θ, we obtain $\mathcal{C}^*(a)\theta \geq^* \mathcal{C}^*(a\theta)$. Applying the definitions of $\geq^*$ and $\mathcal{C}^*$, we get

$$\forall(H \Leftarrow B) \in P^* \; \forall \rho^* \in \Sigma^*(a\theta, H) \; \exists(H' \Leftarrow B') \in P^* \; \exists \sigma^* \in \Sigma^*(a, H') \; \exists \nu^* \; (H \Leftarrow B)\rho^* = (H' \Leftarrow B')\sigma^*\theta\nu^*$$

For every premise of the last step of the translation, we have $(H \Leftarrow B) \in P^*$ and a $\rho^* \in \Sigma^*(a\theta, H)$. Thus we find that there are some $(H' \Leftarrow B') \in P^*$ and $\sigma^* \in \Sigma^*(a, H')$, i.e. there is a corresponding premise of the original step. Furthermore, there is some υ^* such that $(H \Leftarrow B)\rho^* = (H' \Leftarrow B')\sigma^*\theta\upsilon^*$. Splitting this equality in two, we get $H\rho^* = H'\sigma^*\theta\upsilon^*$ and $B\rho^* = B'\sigma^*\theta\upsilon^*$. From $\rho^* \in \Sigma^*(a\theta, H)$ and $\sigma^* \in \Sigma^*(a, H')$, we have $a\theta\rho^* = H\rho^*$ and $a\sigma^* = H'\sigma^*$. From the latter equality, we can conclude $a\sigma^*\theta\upsilon^* = H'\sigma^*\theta\upsilon^*$. Combining these equalities, we obtain $a\theta\rho^* = a\sigma^*\theta\upsilon^*$. If we could also show that $\Gamma\theta\rho^* = \Gamma\sigma^*\theta\upsilon^*$ and $C\theta\rho^* = C\sigma^*\theta\upsilon^*$, then applying the latter three equalities and the equality $B\rho^* = B'\sigma^*\theta\upsilon^*$ above to the inner inference steps of the translation we would get

$$\frac{\Gamma\sigma^*\theta, B\sigma^*\theta \text{ - } C\sigma^*\theta}{\Gamma\sigma^*\theta\nu^*, B\sigma^*\theta\nu^* \text{ - } C\sigma^*\theta\nu^*}$$

which is indeed an instance of the specialisation rule. So, we have to show that $\Gamma\theta\rho^* = \Gamma\sigma^*\theta\upsilon^*$ and $C\theta\rho^* = C\sigma^*\theta\upsilon^*$. By the substitution equality axiom, these equalities would follow if $\theta\rho^*$ and $\sigma^*\theta\upsilon^*$ were equal on the variables and parameters of Γ and C. Now, again by the substitution equality axiom, we already know that $\theta\rho^*$ and $\sigma^*\theta\upsilon^*$ are equal on the variables and parameters of a. By the side conditions above, we know that Γ and C can have no parameters in common with P^*. Thus we only need to consider parameters and variables that are neither in a nor P^*. We will start with the variable case:

Suppose X is a variable neither in P^* (this is trivial) nor a. We immediately have $X\sigma^*\theta\upsilon^* = X\theta\upsilon^*$, so it suffices to show $X\theta\rho^* = X\theta\upsilon^*$. Again using the substitution equality axiom, this holds if $X^*\rho^* = X^*\upsilon^*$, for all $X^* \in \mathcal{PARM}(X\theta)$. We will return to this below. Note that since θ is admissible, $X^* \notin \mathcal{PARM}(P^*)$.

Suppose that X^* is a parameter neither in P^* nor a. We have $X^*\theta\rho^* = X^*\rho^*$. Since σ^* is nonredundant for a and H' and every parameter in H' is also in P^*, we have $X^*\sigma^*\theta\upsilon^* = X^*\theta\upsilon^* = X^*\upsilon^*$, so it suffices to show $X^*\rho^* = X^*\upsilon^*$.

Both the variable and parameter cases require that we show $X^*\rho^* = X^*\upsilon^*$. The only restriction on X^* common to both cases is that $X^* \notin \mathcal{PARM}(P^*)$, except for this X^* can be an arbitrary parameter. We will have two cases, namely that $X^* \in \mathcal{PARM}(a\theta)$ and $X^* \notin \mathcal{PARM}(a\theta)$.

First, assume that $X^* \in \mathcal{PARM}(a\theta)$, then there is some $Y \in \mathcal{VAR}(a)$, such that $X^* \in \mathcal{PARM}(Y\theta)$. By the equality $a\theta\rho^* = a\sigma^*\theta\upsilon^*$ above, and the substitution equality axiom, we have $Y\theta\rho^* = Y\sigma^*\theta\upsilon^*$. Furthermore, $Y\sigma^*\theta\upsilon^* = Y\theta\upsilon^*$, so $Y\theta\rho^* = Y\theta\upsilon^*$. Again by the substitution equality axiom, this implies that $X^*\rho^* = X^*\upsilon^*$, completing the proof of the first case.

Secondly, assume that $X^* \notin \mathcal{PARM}(a\theta)$. Since ρ^* is nonredundant for $a\theta$ and H, we have $X^*\rho^* = X^*$. By the definition of $\leq^*$, υ^* is nonredundant for $a\theta$ and $(H \Leftarrow B)\sigma^*\theta$. If we could show that $X^* \notin \mathcal{PARM}((H \Leftarrow B)\sigma^*\theta)$, we would have $X^*\upsilon^* = X^*$. Combining these equalities, we would obtain $X^*\rho^* = X^*\upsilon^*$.

We will show $X^* \notin \mathcal{PARM}((H \Leftarrow B)\sigma^*\theta)$ by contradiction. Assume that $X^* \in \mathcal{PARM}((H \Leftarrow B)\sigma^*\theta)$. Then there must be some $Y^* \in \mathcal{PARM}(H \Leftarrow B)$, such that $X^* \in \mathcal{PARM}(Y^*\sigma^*\theta)$. Since $X^* \notin \mathcal{PARM}(P^*)$ implies $X^* \notin \mathcal{PARM}(H \Leftarrow B)$, X^* and Y^* must be different, consequently it must be the case that $Y^* \in \mathcal{DOM}(\sigma^*)$. Now, we must have one, or both, of the following situations:

(1) $X^* \in \mathcal{PARM}(Y^*\sigma^*)$
(2) For some $Y \in \mathcal{VAR}(Y^*\sigma^*)$, $X^* \in \mathcal{PARM}(Y\theta)$

In the first situation, since σ^* belongs to a CSU^* of a and H′, by note 3.7 we can assume that all parameters $Z^* \in \mathcal{PARM}(\sigma^*)$ that are different from the parameters of and H′ are also different from any other given parameter. In particular we can assume that $X^* \neq Z^*$, leading to a contradiction.

In the second situation, since σ^* belongs to a CSU^* of a and H′, by note 3.7 we can assume that if Y is different from the variables of a and H′, then it is also different from any other given variable. In particular we can assume that $Y \notin \mathcal{DOM}(\theta)$. But in that case, we could not have $X^* \in \mathcal{PARM}(Y\theta)$, so the only remaining possibility (since H′ is ground) is that $Y \in \mathcal{VAR}(a)$. Thus we would have $X^* \in \mathcal{PARM}(a\theta)$, but this contradicts the main assumption of the second case above.

We have shown that assuming $X^* \in \mathcal{PARM}((H \Leftarrow B)\sigma^*\theta)$ leads to a contradiction, so $X^* \notin \mathcal{PARM}((H \Leftarrow B)\sigma^*\theta)$ must hold, completing the second case and the proof for the D - rule.

Specialisation

$$\frac{\Gamma\theta \text{ - } C\theta}{\Gamma\sigma^*\theta \text{ - } C\sigma^*\theta}$$

Define $\rho^* = (\sigma^*\theta) \setminus \mathcal{DOM}(\sigma^*)$. Let X be some variable in $\mathcal{VAR}(\Gamma) \cup \mathcal{VAR}(C)$. Since θ is admissible, no parameter in $\mathcal{PARM}(X\theta)$, can be eliminated in the original step. Using the definition of an eliminating step, we find that σ^* is nonredundant for parameters being eliminated. Thus we have $X\theta = X\theta\sigma^*$, consequently $X\theta = X\theta\rho^*$. Since σ^* is a parameter substitution, we have $X\sigma^*\theta = X^*\theta = X\theta\rho^*$. Let Y^* be some parameter. Since θ is a variable substitution, we have $Y^*\sigma^*\theta = Y^*\rho^* = Y^*\theta\rho^*$. By the substitution equality axiom, we have $\Gamma\sigma^*\theta = \Gamma\theta\rho^*$ and $C\sigma^*\theta = C\theta\rho$. Applying this equality to the step above, we get

$$\frac{\Gamma\theta \text{ - } C\theta}{\Gamma\theta\rho^* \text{ - } C\theta\rho^*}$$

which is a correct instance of the specialisation rule.

Induction

$$\frac{\left\langle \begin{array}{c|c} \{S(C)\theta \mid C \in \langle B \rangle\} & \\ \vdots & (H \Leftarrow B) \in T^* \\ S(H)\theta & \end{array} \right\rangle}{S(X^*)\theta}$$

We have $S(X^*)\theta = S\theta(X^*\theta) = S\theta(X^*)$ Since H and $\langle B \rangle$ are ground by definition, we have $H = H\theta$ and $C = C\theta$, for every H and $C \in \langle B \rangle$. Thus $S(H)\theta = S\theta(H\theta) = S\theta(H)$ and $S(C)\theta = S\theta(C\theta) = S\theta(C)$, applying these equalities, we get:

$$\frac{\left\langle \begin{array}{c|c} \{S\theta(C) \mid C \in \langle B \rangle\} & \\ \vdots & (H \Leftarrow B) \in T^* \\ S\theta(H) & \end{array} \right\rangle}{S\theta(X^*)}$$

which is indeed an instance of the induction schema .

8. PROCEDURAL ASPECTS OF THE CALCULUS WITH VARIABLES

As the calculus with variables is intended to facilitate the construction of proofs, it is intimately connected with operational aspects such as proof construction. We will define operational rules suitable for constructing a derivation is the most general fashion. Together with a search strategy, these operational rules can be used to define a proof procedure for the finitary calculus. We will not present any particular techniques for implementing these rules.

It should be noted that results presented in this section are essentially generalisations to the finitary calculus of previous work on logic programming languages based on partial inductive definitions [4,5,13,14,18]. While adaptions of these results to our finitary calculus have made and included for completeness, it would go outside the scope of the present work to find completely new results about the operational aspects. Unfortunately, this means that some aspects (particularly that of computing a-sufficient substitutions below) will have to be left to future work.

Let us consider the general way of developing - manually or automatically - a proof in a goal-directed fashion. We will start will the sequent (goal) to be proved. Some inference rule is then chosen to derive that sequent. We must chose what particular instance of that inference rule to use, and possibly instantiate the derivation to make the chosen rule applicable. In general, the inference rule has a number of premise sequents (subgoals) that must in their turn be proved. This procedure is repeated with one of the premises, until a derivation without premises is

found. The proof is then completed. Thus at any stage, the proof under development will consist of a derivation with the sequent to be proved as endsequent, and some number of unproved premises.

Formalising slightly, we can break down the procedure into a number of steps:

1) Choose one of the subgoals.
2) Choose an inference rule for proving the subgoal.
3) Choose the particular term the inference rule will operate on.
4) Apply a variable substitution to the derivation to make the rule applicable.
5) Choose the particular instance of the inference rule.
6) Add the new inference step to the derivation.
7) Repeat.

Steps 1 - 3 are essentially search problems. The choices made in these steps are determined by the search strategy of an automatic theorem prover system, or the intuition of the user in case of manual proof development. Although the problem of automatic theorem proving in the theory of partial inductive definitions is certainly an interesting one, we will not investigate it in the present article.

Steps 4 - 5 are partly search problems. If, for instance the D - rule was chosen and several clauses are applicable, it is a search problem which of these to choose. However, some of the choices of steps 4 - 5 can be made in a most general manner, in other cases we can show that some choices are definitely not general and need not be considered. For each inference rule, we will define an operational rule, that will perform steps 4 - 6.

Each operational rule will consist of four parts, the conclusion of the rule, the premises, any side conditions, and the substitution, θ, to be applied to the entire derivation. If the conclusion of an operational rule instance is a subgoal of the derivation being developed and the side conditions are fulfilled, the rule is applied by instantiating the derivation using the substitution and adding the basic inference rule corresponding to the operational rule with the given premises. For each operational rule, we must show its correctness, i.e. that the inference step added in this way is a valid inference step.

A requirement common to all operational rules is that the substitution θ must be admissible. We will return to the question of how to ensure this.

In many of the operational rules we make use of the definition P, or its parameter transform P^*. In order to satisfy the assumptions of notes 3.9, each time P (or P^*) is used, all variables (or parameters) must be renamed to be different from all other variable (or parameter) names used so far in the derivation construction process. Likewise, in order to satisfy the assumptions of note 3.7, every time a CSU(*) of two terms is constructed, all variables and parameters in that CSU(*), that do not already occur in the two terms being unified, must be different from all variables and parameters used so far in the derivation construction process. (Strictly speaking, the way notes 3.7 and 3.9 are used, it suffices if new parameters are different from all induction parameters at the particular point of the derivation, and from all parameters in the conclusion of the derivation step being introduced.)

Considering the inference rules, we can see that the applicability of the $\Rightarrow$-, -$\Rightarrow$, and -() rules depend only on the structure of the conclusion. If steps 1 - 3 above choose one of these rules with a sequent of the correct form, that rule will be immediately applicable without any further choices and without having to make any instantiation of the proof. Neither do these rules have any side conditions that have to be verified. Recall that instantiating a proof eliminates potential derivations, so we do not want to make any unnecessary instantiations. For the rule () -, the situation is similar. A choice has to be made as to which term in the sequence to use, however this is again a search problem. For these four rules, then, the operational rule will be the same as the basic inference rule with an empty substitution

The contraction rule is necessary in some derivations, but should not be used freely. As is the case with using contraction in theorem proving for logic, the present contraction rule should be used in a controlled way together with certain other rules (Π -, $\Rightarrow$ -, () - and D -). The best way of handling contraction would be to combine it with those rules. How this is done, however, is a nontrivial strategy problem that we will not consider here. Instead we will have the operational contraction rule as a separate rule. Just as for the rules considered in the previous paragraph, the operational rule for contraction will be the same as the basic inference rule with an empty substitution.

The weakening and specialisation rules are only necessary in conjunction with induction hypotheses. For this purpose, we define a special "hypothesis" operational rule. Should operational rules for pure weakening or specialisation be required, they are simple to define.

The remaining inference steps will be treated individually. We will state the operational rule and verify that it is correct. By "the derivation", we will mean the derivation developed so far when the operational rule is applied.

Axiom

$$\Gamma, a \text{ - } b \qquad \theta \in \Sigma(a,b) \qquad \text{(axiom)}$$

By applying θ to the derivation, the sequent will get the form $\Gamma\theta, a\theta$ - $b\theta$. Since $a\theta = b\theta$, this is a correct instance of the axiom rule. Here there is a choice as to which particular unifier of the CSU to take as θ. However, there is clearly no need to consider unifiers other than those of some arbitrary CSU of a and b.

-Π

$$\frac{\Gamma \text{ - } C(X^*)}{\Gamma \text{ -} \Pi x\ C(x)} \qquad \theta = \varnothing \qquad (\text{-}\Pi)$$

where X^* is an arbitrary parameter that does not occur anywhere in the derivation or the program
and $\mathcal{RAN}(x) = \mathcal{RAN}(X^*)$.

This choice of X^* clearly satisfies all side conditions of the -Π rule.

Π -

$$\frac{\Gamma, C(X) - C'}{\Gamma, \Pi x\ C(x) - C'} \qquad \theta = \varnothing \qquad (\Pi -)$$

where X is an arbitrary variable that does not occur anywhere in the derivation

and $\mathcal{RAN}(X) = \mathcal{RAN}(x)$.

This choice of X clearly satisfies all side conditions of the $\Pi -$ rule. The basic rule permits an arbitrary term in the range of x, instead of X. However, as X can be instantiated to an arbitrary term later in the procedure, choosing a variable is the most general choice.

-D

$$\frac{\Gamma\theta - B\theta}{\Gamma - a} \qquad \theta \in \Sigma(a,H), \text{ for some } (H \Leftarrow B) \in P \qquad (-D)$$

After applying the substitution, the conclusion will have the form $\Gamma\theta - a\theta$. This sequent clearly fulfils the side condition of the basic rule if we take the σ of the side condition to be the current θ, as $\theta \in \Sigma(a,H)$ implies $a\theta = H\theta$. The requirement that every $\theta \in \Sigma(a,H)$ is nonredundant for a and H means that no variables except those in a and H are instantiated. In particular, no variables in B will be instantiated unless they are already in H. In the formulation of the basic -D rule, the substitution σ could instantiate such variables also. As these variables can be instantiated in later steps, if need be, not instantiating them here, is the most general choice.

When using this rule, we must still choose the particular clause to apply if more than one is possible, and the particular unifier from the CSU. However, we have clearly no need to consider unifiers other than those of an arbitrary CSU of a and H.

D -

$$\frac{\left\{\Gamma\theta\sigma^*, B\sigma^* - C\theta\sigma^* \mid \sigma^* \in \Sigma^*(a\theta,H),\ (H \Leftarrow B) \in P^*\right\}}{\Gamma, a - C} \qquad \theta \text{ given below} \qquad (D -)$$

θ will be computed by an algorithm producing a-sufficient substitutions (defined later in this section). This will ensure that the side condition that $\mathcal{C}^*(a\theta)\varphi \cong^* \mathcal{C}^*(a\theta\varphi)$, for all φ, is fulfilled, as that is precisely the requirement on a-sufficient substitutions. After applying the substitution, the conclusion will have the form $\Gamma\theta, a\theta - C\theta$. Thus the step will be a correct instance of the basic D - rule.

The precautions described above to satisfy the assumptions of note 3.9 will also ensure the validity of the side conditions that $\mathcal{PARM}(P^*) \cap \mathcal{PARM}(\Gamma, a\theta - C) = \varnothing$ and that $\mathcal{PARM}(P^*)$ must not contain any induction parameters. Since we are dealing with an operational rule here, we do not need to verify the side condition that each $\Sigma^*(a\theta,H)$ is finite. If some $\Sigma^*(a\theta,H)$ was infinite, computation of the premises of the rule would not terminate, so an incorrect derivation would never be constructed!

Hypothesis

$$\frac{\Gamma' \text{ - } C'}{\dfrac{\Gamma'\sigma^* \text{ - } C'\sigma^*}{\Gamma \text{ - } C}} \qquad \theta=\varnothing \qquad \begin{matrix}\text{(specialisation)}\\ \text{(weakening)}\end{matrix}$$

The hypothesis rule is only used in conjunction with induction and adds one specialisation and one weakening step (both possibly trivial) to the derivation. An induction hypothesis $\Gamma' \text{ - } C'$ and a parameter substitution σ^* should be chosen such that $\Gamma'\sigma^* \subseteq \Gamma$, $C=C'\sigma^*$, and $\mathcal{DOM}(\sigma^*)$ does not contain any induction parameters. Since $\Gamma' \text{ - } C'$ is an induction hypothesis, this step does not introduce any premises that need to be proved.

Induction

$$\frac{\left\{\begin{array}{c|c} \{S(C) \mid C \in \langle B \rangle\} & \\ \vdots & (H \Leftarrow B) \in T^* \\ S(H) & \end{array}\right\}}{S(X^*)} \qquad \theta=\varnothing \qquad \text{(induction)}$$

The the definition T^* to do induction over, and the induction parameter, X^*, should be chosen such that the side condition $\mathit{ran}(X^*)=\mathit{Def}(T)$ is fulfilled. The precautions described above to satisfy the assumptions of note 3.9 will also ensure that these remaining side conditions, $\mathcal{PARM}(H) \cap \mathcal{PARM}(S)=\varnothing$ and $\mathcal{PARM}(H) \cap \mathcal{PARM}(\langle B \rangle)=\varnothing$, for every H and $\langle B \rangle$, hold.

EXAMPLE 8.1

We construct a derivation of the sequent $\Pi x\ p(x) \text{ - } p(c)$. In the first step, the Π - rule is applied, giving:

$$\frac{p(X) \text{ -} p(c)}{\Pi x\ p(x) \text{ -} p(c)}\ (\Pi \text{ -})$$

In the next step, the axiom rule is applied. A unifying substitution of p(X) and p(c) is {X/c}, giving:

$$\frac{\dfrac{}{p(c) \text{ -} p(c)}\ \text{(Axiom)}}{\Pi x\ p(x) \text{ -} p(c)}\ (\Pi \text{ -})$$

®

EXAMPLE 8.2

Given the following definition of the append relation:

append(nil,X,X) ⇐
append(X.A,B,X.C) ⇐ append(A,B,C)

we construct a derivation of the sequent - append(a.b.nil,c.nil,L), expressing the statement that some L is result of appending together the lists a.b.nil and c.nil. During construction of the proof, L will be instantiated to some term with this property. In the first step, the -D rule is applied, with the substitution {X/a, A/b.nil, B/c.nil, L/a.C} giving:

$$\frac{\text{-append(b.nil,c.nil,C)}}{\text{-append(a.b.nil,c.nil,a.C)}}\ (\text{-P})$$

In the next step, the -D rule is again applied. To satisfy the requirements of note 3.9, the variables in the definition will be renamed by adding a ′ after their names. The substitution will be {X′/b, A′/nil, B′/c.nil, C/b.C′} giving:

$$\frac{\dfrac{\text{-append(nil,c.nil,C')}}{\text{-append(b.nil,c.nil,b.C')}}\ (\text{-P})}{\text{-append(a.b.nil,c.nil,a.b.C')}}\ (\text{-P})$$

In the next and final step, the -D rule is again applied. To satisfy the requirements of note 3.9, the variables in the definition will be renamed by adding a ″ after their names. The substitution will be {X′/c.nil, C′/c.nil} giving:

$$\frac{\dfrac{\dfrac{}{\text{-append(nil,c.nil,c.nil)}}\ (\text{-P})}{\text{-append(b.nil,c.nil,b.c.nil)}}\ (\text{-P})}{\text{-append(a.b.nil,c.nil,a.b.c.nil)}}\ (\text{-P})$$

®

A-sufficient substitutions

The operational D - rule presents a particular problem, in that a substitution θ must be chosen, such that the side condition $\mathcal{C}^*(a\theta)\varphi \cong^* \mathcal{C}^*(a\theta\varphi)$ holds. We have the following definition:

DEFINITION 8.3 (a-sufficient substitutions)
A variable substitution θ is called **a-sufficient** iff $\mathcal{C}^*(a\theta)\varphi \cong^* \mathcal{C}^*(a\theta\varphi)$ holds. ®

The requirement of a-sufficiency is to ensure that an instance of the D |- rule remains valid even after variables occurring in the rule instance have been instantiated. Computing an a-sufficient substitution is a formidable problem. In fact, there is the double problem of finding an efficient algorithm to enumerate a-sufficient substitutions, while avoiding redundant ones. An a-sufficient substitution can be "redundant" in approximately the same sense that a unifier is "not most general" . Suppose that using the two a-sufficient substitutions θ′ and θ″ gives D - rule instances with the sets of premises S′ and S″, respectively. If $S' \leq^* S''$, θ″ can be considered redundant in relation to θ′. Intuitively, this means that using θ″ we have to prove both the same premises as if we use θ′, and also additional ones.

Hallnäs and Schroeder-Heister [14] introduced the concept of a-sufficient substitutions in connection with a much restricted finitary calculus with inference rules similar to ours (see section 9). Their formulation of a-sufficient substitutions is different from ours, but it is not difficult to show that for finitary definitions and derivations that can be expressed in both our finitary calculus and the one from their work, the two formulations coincide. Algorithms to

compute a-sufficient substitutions are given in [14] and in [18]. As of this writing, our work has not progressed far enough to give an algorithm for the computation of a-sufficient substitutions in the more general case of our finitary calculus. We will just note that such an algorithm could probably be developed using the algorithms from [14] or [18] as a starting point.

There are, however, some special cases which obviously satisfies the requirement of an a-sufficient substitution. In particular, every substitution that is grounding for a will be an a-sufficient substitution, since in that case applying the substitution φ will have no effect.

EXAMPLE 8.4
Given the following definition of the ≤ relation on natural numbers

$$0 \leq X \Leftarrow$$
$$s(X) \leq s(Y) \Leftarrow X \leq Y$$

we construct a derivation of the sequent $\text{-}\, s(0) \leq X \Rightarrow \perp$, expressing the statement that some X is not greater than or equal to one. During construction of the proof, X will be instantiated to some number with this property. We assume that all variables range over the natural numbers expressed using the constant 0 (zero) and the successor function s. In the first step, the $\text{-}\Rightarrow$ rule is applied, giving:

$$\frac{s(0)\leq X \;\text{-}\perp}{\text{-}s(0)\leq X \Rightarrow \perp}\;(\text{-}\Rightarrow)$$

In the next step, the D - rule is applied. Some s(0)≤X-sufficient substitution must be computed. It is easy to see that {X/0} is such a substitution, as applying it to s(0)≤X causes that term to be ground. As the D - rule will have no premises with this substitution, the derivation is completed with X being instantiated to 0.

$$\frac{\dfrac{}{s(0)<0 \;\text{-}\perp}\;(\text{P -})}{\text{-}s(0)\leq 0 \Rightarrow \perp}\;(\text{-}\Rightarrow)$$

®

Ensuring admissibility

The substitutions θ being computed by the operational rules are used to instantiate the entire derivation, which requires them to be admissible. Since every θ is computed as a unifier, it should be most convenient to include this check in the unification algorithm. The obvious way of ensuring that every θ is admissible would be to maintain the predating relation between variables and parameters. Whenever an attempt is made to construct a binding {X/t}, a check could be made to see if X predates any parameter of t. In that case, the binding would be inadmissible. The problem with this approach is that the relation needs to be updated whenever a variable is bound, since this may cause new variables to predate some parameter.

EXAMPLE 8.5

Suppose that X predates Z^*, but Y does not. Now, if f(Y) is substituted for X, then Y will also predate Z^*. If a predating relation is kept, when binding X to f(Y), the relation should be updated so that Y predates the same parameters as does X (in addition to any variables that Y predated initially). ®

Maintaining explicit predating relations in this way is cumbersome and not very elegant. However, we can do better. The admissibility problem is closely related to a problem occurring in logic program execution when explicit quantifiers are present. Consider how a Prolog extension with explicit quantification would attempt to execute the goal $\forall y\ X=y$. Since free variables in goals are existentially quantified, this goal is equivalent to the formula $\exists x \forall y\ x=y$. In our calculus, as we saw in example 7.2, the admissibility condition prevents this formula from being proven.

The language λProlog [22] would solve this goal by generating the new goal X=f(X), where f is a Skolem function. Clearly, this equality cannot be solved. The language L_λ [20], developed from λProlog, takes a different approach. L_λ would first generate the new goal X=c, where c is a new constant. Normally, the goal X=c could be solved by binding X to c. In this case, however, that would be unsound. The unification algorithm of L_λ maintains its soundness by using information about quantifier ordering to prevent such unsound bindings from being made.

It is easy to see that the universal quantification of y inside the scope of the existential quantification of x is completely analogous to the predating of the parameter Y^* by the variable X, in example 7.2. In general, the relation "the variable y is universally quantified inside the scope of the existentially quantified variable x" corresponds exactly to the relation "X predates Y^*". In [21], Miller describes a generalisation of the method used by L_λ, to do general unification of arbitrary higher-order terms with these constraints. This method could be used in our calculus, provided that the term theory used is the same as that in [21]. (λ-terms with αβη-convertibility as equality.)

9. RELATED WORK

As mentioned in the introduction, there are two main alternative finite formulations of the theory of partial inductive definitions. The most important one is the work by Hallnäs and Schroeder-Heister [13,14]. Although there is no formal connection between that work and the theory of partial inductive definitions there is an obvious relation and it is clear that the system D(P) of Hallnäs and Schroeder-Heister is a essentially a true subset of our finitary calculus with variables, using a first-order term theory with identity as equality. Likewise, their system LD(P) is - apart from some superficial differences in organising the derivation and the instantiating substitution - essentially a true subset of our procedural system from section 8. The GCLA programming language [4,5] is a computer implementation of the system LD(P).

The important difference between D(P) / LD(P) and our systems is the lack of parameters and the consequent lack of everything that depends on parameters. In particular, there is no induction and no Π-condition. The lack of parameters implies that the D - rule is formulated without the parameter transform P^*. Also, the D - rule cannot be used with clauses where the

body contains a variable that does not occur in the head, as such variables would unavoidably require a parameter in the premise corresponding to that clause.

In [15], Hanschke presents another limited finitary calculus. That system is similar to the D(P) / LD(P) systems, extended with ^ and ⊥ conditions, our Π condition, and with a new condition using a Σ-operator - Σx C(x). There is no -Π rule in Hanschke´s system, and as the Π - rule does not require parameters, that extension is straightforward. The intention of the Σ-condition is to go around the restriction shared with D(P) / LD(P) that clauses used with D - cannot have a variable in the body that does not also occur in the head. If all such variables are bound using the Σ-operator, the clause can be used with the D - rule. The Σ-conditions will then appear in the premise of that rule. In a sense, Σ becomes the dual of Π. To handle Σ-conditions, Hanschke defines a Σ - rule. A sound formulation of such a rule would be

$$\frac{\Gamma, C(X^*) - C'}{\Gamma, \Sigma x\ C(x) - C'} \qquad (\Sigma\ -)$$

Where $X^* \notin \mathcal{PARM}(\Gamma)$, $X^* \notin \mathcal{PARM}(C)$ and $X^* \notin \mathcal{PARM}(C')$

As his system does not have parameters, a Skolem constant is used instead of the parameter X^*. In the general case this leads to inconsistencies (a Skolem constant cannot be substituted for something else in the D - rule, as a parameter can), but in the particular case for which his system is used, his formulation is acceptable. (The actual rule given in [15] contains an implicit contraction, but that is a trivial difference.)

10. REFERENCES

[1] Aczel, Peter, *An Introduction to Inductive Definitions*, in: Handbook of Mathematical Logic (Barwise, J., ed.), North-Holland, Amsterdam 1977.

[2] Aronsson, Martin, *STRIPS-Like Planning Using GCLA*, SICS research report R89009, Swedish Institute of Computer Science, 1989.

[3] Aronsson, Martin, *A Definitional Approach to the Combination of Functional and Relational Programming*, SICS research report R91:10, Swedish Institute of Computer Science, 1991.

[4] Aronsson, M., Eriksson, L.-H., Gäredal, A., Hallnäs, L. and Olin, P.,*The Programming Language GCLA - A Definitional Approach to Logic Programming*, New Generation Computing, vol. 7 no. 4 (1990), pp. 381-404.

[5] Aronsson, M., Eriksson L.-H., Hallnäs, L. and Kreuger, P., *A Survey of GCLA: A Definitional Approach to Logic Programming*, In: P. Schroeder-Heister (ed.), *Extensions of Logic Programming*, Springer Lecture Notes in Computer Science 475, 1991

[6] Eriksson, Lars-Henrik, *Pi Users Manual (Preliminary)*, internal note, Swedish Institute of Computer Science, 1991.

[7] Eriksson, Lars-Henrik and Hallnäs, Lars, *A Programming Calculus Based on Partial Inductive Definitions*, SICS research report R88013, Swedish Institute of Computer Science, 1988.

[8] Fredholm, Daniel, *On Function Definitions I*, Licentiate thesis, Department of Computer Sciences, Chalmers University of Technology, Gothenburg, 1990.

[9] Fredholm, Daniel and Serafimovski, Svetozar, *Partial Inductive Definitions as Type-Systems for λ-Terms*, to appear in BIT, (also published in Dybjer et.al. (eds), *Proceedings on the Workshop on Programming Logic*, report PMG-R54, Department of Computer Sciences, Chalmers University of Technology, Gothenburg, 1989).

[10] Hallnäs, Lars, *A Note on Non-Monotonic Reasoning*, In: Brown, F.M. (ed.) *The Frame Problem in Artificial Intelligence*, Proceedings of the 1987 Workshop, Morgan Kaufmann, Los Altos, 1987.

[11] Hallnäs, Lars, *Partial Inductive Definitions*, Theoretical Computer Science, vol. 87, no. 1 (1991).

[12] Hallnäs, Lars and Nordström, Bengt, *A Definitional View of Functional Programming*, In: Dybjer et.al. (eds.), *Proceedings on the Workshop on Programming Logic*, report PMG-R54, Department of Computer Sciences, Chalmers University of Technology, Gothenburg, 1989

[13] Hallnäs, Lars and Schroeder-Heister, Peter, *A Proof-Theoretical Approach to Logic Programming, Part I*, Journal of Logic and Computation, vol. 1, no. 2 (1990).

[14] Hallnäs, Lars and Schroeder-Heister, Peter, *A Proof-Theoretical Approach to Logic Programming, Part II*, Journal of Logic and Computation, vol. 1, no. 5 (1991).

[15] Hanschke, Philipp, *Terminological Reasoning and Partial Inductive Definitions*, this volume.

[16] Huet, G.P., *A Unification Algorithm for Typed λ-calculus*, Theoretical Computer Science, 1 (1975), pp. 27-57.

[17] Huet, G.P., *Unification en théorie des types*, Séminaire IRIA Théorie des Automates, des languages et de la programmation, 1973.

[18] Kreuger, Per, *GCLA II - A Definitional Approach to Control*, this volume.

[19] Lassez, J.-L., Maher, M.J. and Marriott, K.G., *Unification Revisited*, In: Minker, J. (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, Los Altos, 1988.

[20] Miller, Dale, *A Logic Programming Language with Lambda-Abstraction, Function Variables, and Simple Unification*, Journal of Logic and Computation vol.1, no. 4 (1991).

[21] Miller, Dale, *Unification Under a Mixed Prefix*, to appear in Journal of Symbolic Computation. Also published as report MS-CIS-91-81, Dept. of Computer and Information Science, University of Pennsylvania, 1991.

[22] Nadathur, Gopalan and Miller, Dale, *An Overview of λProlog*, Fifth International Conference of Logic Programming, MIT Press, pp 810-827.

[23] Palamidessi, Catuscia, *Algebraic Properties of Idempotent Substitutions*, Technical report TR-33/89, Dipartimento di informatica, Università di Pisa, 1989.

[24] Palmkvist, Johan, *Implementation of a Planning System Using GCLA*, SICS technical report T89018, Swedish Institute of Computer Science, 1989.