

A Logic of Non-Monotone Inductive Definitions

MARC DENECKER

K.U.Leuven, Belgium

EUGENIA TERNOVSKA

Simon Fraser University, Canada

Well-known principles of induction include monotone induction and different sorts of non-monotone induction such as inflationary induction, induction over well-founded sets and iterated induction. In this work, we define a logic formalizing induction over well-founded sets and monotone and iterated induction. Just as the principle of positive induction has been formalized in FO(LFP), and the principle of inflationary induction has been formalized in FO(IFP), this paper formalizes the principle of iterated induction in a new logic for Non-Monotone Inductive Definitions (ID-logic). The semantics of the logic is strongly influenced by the well-founded semantics of logic programming.

This paper discusses the formalisation of different forms of (non-)monotone induction by the well-founded semantics and illustrates the use of the logic for formalizing mathematical and common-sense knowledge. To model different types of induction found in mathematics, we define several subclasses of definitions, and show that they are correctly formalized by the well-founded semantics. We also present translations into classical first or second order logic. We develop modularity and totality results and demonstrate their use to analyze and simplify complex definitions. We illustrate the use of the logic for temporal reasoning. The logic formally extends Logic Programming, Abductive Logic Programming and Datalog, and thus formalizes the view on these formalisms as logics of (generalized) inductive definitions.

Categories and Subject Descriptors: ... [..]: ...

1. INTRODUCTION

This paper studies the extension of classical logic with a general notion of inductive definitions. An inductive definition is a recipe to construct a mathematical object, usually one or more sets, by describing when to add elements given the presence or absence of other objects. In mathematical texts, inductive definitions are usually represented as collections of rules representing a set of base cases and inductive cases. Inductive rules may be *monotone* or *non-monotone*. A familiar example

Author's address: Marc Denecker, Department of Computer Science, K.U.Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium. Phone: +32 16 327544 — Fax: +32 16 327996. Email: marc@cs.kuleuven.ac.be

Evgenia Ternovska, Computational Logic Laboratory, School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6. Phone: +1 604 268-7008 — Fax :+1 604 291-3045. Email: ter@cs.sfu.ca

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2006 ACM 1529-3785/2006/0700-0001 \$5.00

in which both type of rules occur is the definition of the satisfaction relation \models between a truth assignment I and a formula. In case of propositional logic, this relation is defined by induction on the subformula order on formulas:

- $I \models p$ if $p \in I$,
- $I \models \psi \wedge \phi$ if $I \models \psi$ and $I \models \phi$,
- $I \models \psi \vee \phi$ if $I \models \psi$ or $I \models \phi$,
- $I \models \neg\psi$ if $I \not\models \psi$.

The second and third rules are monotone, while the last rule is non-monotone because it adds the pair $(I, \neg\psi)$ to the truth relation if the pair (I, ψ) does not belong to it. This is an example of an inductive definition over a well-founded order, given here by the subformula relation.

It is well-known that, in general, inductive definitions cannot be expressed in first-order logic (FO). Yet, inductively defined concepts are often very useful in practice. For example, in the context of databases, query languages have been extended with fixpoint constructs to represent inductively definable concepts. Also description logics have been extended with such fixpoint constructs. The lack of expressive power of FO logic to represent recursion (including recursion through negation) has motivated its extension with fixpoint constructs. In these logics, several forms of induction have been modelled: e.g. monotone induction, partial fixpoint induction, inflationary fixpoint induction. However, there are forms of non-monotone induction which are quite common in mathematics that are not modelled well by fixpoint logics. In particular, inductions over well-founded orders and iterated inductions [Kreisel 1963; Buchholz et al. 1981] are not handled naturally in any of the fixpoint logics.

Recently, the authors of [Denecker 1998; Denecker et al. 2001] investigated such generalized forms of induction in mathematics and pointed out that semantical studies in the area of logic programming may contribute to a better understanding of them. In particular, it was argued that the well-founded semantics of logic programming [Van Gelder et al. 1991] extends monotone induction and formalises and generalises induction over well-founded sets and iterated induction. This observation motivated the first author to define ID-logic, an extension of classical logic with a notion of inductive definition based on the well-founded semantics [Denecker 2000]. In [Denecker and Ternovska 2004b], we extended ID-logic by allowing arbitrary boolean combinations of FO formulas and definitions. There, we also investigated the modularity properties of definitions in ID-logic and presented equivalence preserving transformations of subclasses of ID-logic to classical logic.

On the computational level, ID-logic has recently been proposed as the underlying language for a constraint programming framework [Mitchell and Ternovska 2005]. This framework is based on the search for models which satisfy an ID-logic theory and expand a given structure by new relations. A parameterized version of the framework was shown to capture the complexity class NP, and other complexity classes can also be captured. Several ID-logic solvers implementing this computational paradigm have been developed [Pelov and Ternovska 2005; Mariën et al. 2006], and active research on formal foundations is going on.

The computational role of ID-logic is not restricted to model expansion. ID-logic formally extends several computational logics such as Logic Programming and Datalog, and Abductive Logic Programming. Prolog systems such as XSB [Rao et al. 1997] and query systems for Datalog can be seen as query systems for fragments of ID-logic. Abductive inference systems from Abductive Logic Programming such as the Asystem [Kakas et al. 2001] can be used to perform abductive reasoning in (fragments of) ID-logic. Another topic of research is the development of deductive inference methods for ID-logic extending methods from fixpoint logics such as [Compton 1993].

The current paper contains the complete presentation of the ideas in the two conference papers [Denecker 2000; Denecker and Ternovska 2004b] and extends these studies in several respects. We define ID-logic as an extension of classical logic with inductive definitions. This enterprise is similar to the extension FO(LFP) of classical logic with monotone fixpoints, or to the extension FO(IFP) with inflationary fixpoints. The main difference with fixpoint logics is that it incorporates other forms of non-monotone induction, which arise much more often in mathematics and common-sense reasoning. The main goal of this paper is to demonstrate that this integration of classical logic and (non-monotone) inductive definitions leads to a coherent, natural and useful logic for knowledge representation. To this end, we study the following topics:

- We explain the different notions of inductive definition and give an intuitive argument why well-founded semantics correctly formalises them. This was attempted before in [Denecker et al. 2001], but we improve the argument.
- Definitions over well-founded sets occur very frequently in mathematics. Other forms of non-monotone induction such as iterated inductive definitions occur not as frequently. We will show examples that illustrate both concepts. We will show that in the context of knowledge representation, both types of definitions arise very naturally.
- We demonstrate the use of inductive definitions to model several well-known principles of knowledge representation: Domain Closure Axiom, Closed World Assumption, default reasoning and default inheritance, and temporal reasoning.
- We define some special classes of definitions, to model different types of definitions found in mathematics: monotone inductive definitions, definitions over well-founded orders, and iterated inductive definitions. For each class, we define an alternative formal semantics which more directly captures that particular kind of induction. We then show that for each class, this new semantics coincides with the well-founded semantics. This gives further evidence to the thesis that well-founded semantics uniformly formalises different forms of inductive definitions.
- Non-monotone inductive definitions are descriptions of complex mathematical construction processes. A technical part of the paper is devoted to the development of a set of tools to analyse inductive definitions: (1) dependency relations give a view on the internal structure of a definition; (2) modularity results allow one to split up big definitions into a conjunction of often much simpler sub-definitions; (3) totality conditions guarantee that a definition is *total*, i.e., defines its relation(s) in an exact and total manner; and (4) translations from subclasses

of definitions to first- and second-order formulas show connections with classical logic.

- We present a temporal reasoning example and show a simple and natural solution using iterated inductive definitions in ID-logic. To our knowledge, this example cannot be represented in any of the existing situation calculus formalisms. We also use this example to demonstrate the analysis techniques: we show the internal structure of the definition, break it up in simple sub-theories and translate the theory to classical logic.
- Here, we will not study ID-logic from an expressivity theoretic point of view. Well-founded semantics has been investigated from this perspective, e.g. in [Van Gelder et al. 1991; Van Gelder 1993; Schlipf 1995b]. The perspective of this paper is *epistemological* and *pragmatic*, i.e., this paper is a formal study of inductive definitions, a form of knowledge that appears informally in mathematics; we study the relation with common-sense knowledge representation principles; the mathematical tools developed here serve to clarify the meaning of inductive definitions expressed in ID-logic.

We consider it as one of our most important achievements that ID-logic is a coherent and conceptually clean *integration* of classical logic and logic programming. These two formalisms are usually viewed as serving such thoroughly different representational and computational purposes, that an integration seems impossible or even undesirable. But if we see logic programs as declarative representations of inductive definitions, then it is natural to integrate logic programs with classical logic, in order to compensate for the latter's representational weakness on inductive definability.

Many of the concepts, techniques and results used here are generalizations or variants of concepts, techniques or results introduced in logic programming. There are however, some important differences. To be able to integrate logic programming in classical logic, we need to drop two standard constraints of logic programming. The first is that a logic program defines *all* its predicates. An ID-logic definition defines only a subset of *defined predicates* of the theory and does not constrain other predicates. This feature is a significant improvement from the knowledge representation point of view.

The second is that logic programming semantics are typically based on Herbrand interpretations. To achieve an integration with classical logic, we base the semantics of ID-logic on general interpretations. Consequently, the use of classical logic forces us to generalise many concepts of Logic Programming to arbitrary interpretations rather than Herbrand interpretations, and to first-order rule bodies rather than conjunctions of literals. We devote Section 4.3 to a discussion of the history of the view of logic programs as inductive definitions.

2. FORMAL STUDY OF INDUCTIVE DEFINITIONS

Mathematical induction refers to a class of construction techniques used in mathematics. There, a set, or a set of sets, or a structure, is frequently defined as the result of an iterative constructive process, consisting of repeated applications of a basic set of operations. Mathematicians often describe such a construction by an *inductive definition*. The scientific study of the concept of inductive definition, as

used in mathematical texts, is an important topic of mathematical logic. In this section, we discuss several *informal* forms of inductive definitions and how they have been formalized. Then we will introduce the definition construct of ID-logic and explain what forms of induction it formalizes. The section refines ideas presented earlier in [Denecker 1998; Denecker et al. 2001].

Mathematicians often follow certain *linguistic conventions* to phrase such definitions. For example, reconsider the definition from Section 1:

Definition 2.1. The truth relation \models between interpretations I and propositional sentences φ is defined by induction on the subformula order (or on the length) of formulas:

- $I \models p$ if $p \in I$,
- $I \models \psi \wedge \phi$ if $I \models \psi$ and $I \models \phi$,
- $I \models \psi \vee \phi$ if $I \models \psi$ or $I \models \phi$,
- $I \models \neg\psi$ if $I \not\models \psi$.

This definition starts with the phrase that the defined set “*is defined by induction (on some order)*” followed by a set of basic and inductive rules, where the latter type of rules specify when to add elements to the defined set, given the presence or absence of other elements.

In most inductive definitions in mathematics, the inductive rules are all *monotone*, meaning that they add elements to the defined sets given the presence of other elements in the set. For example:

Definition 2.2. The transitive closure T_G of a directed graph G is inductively defined by the following rules:

- $(x, y) \in T_G$ if $(x, y) \in G$;
- $(x, y) \in T_G$ if for some vertex z , $(x, z) \in T_G$ and $(z, y) \in T_G$.

Other typical examples of monotone inductive definitions are the definition of a subgroup generated by a set of group elements, or the definitions of a term, formula, etc. in logic. A fundamental application in logic is that of the deductive closure $Cn(T)$ of a propositional or first-order logic theory T which is inductively defined by the formulas of T and a sound and complete set of inference rules.

The set defined by such definitions can be characterized in a *downward*, non-constructive way, as the least set closed under application of the rules, i.e., as the intersection of all sets for which the conclusion of each rule is satisfied whenever the condition is satisfied. Alternatively, the defined set can also be characterized in an *upward*, constructive way as the limit of a process starting from the empty set and iteratively applying rules until saturation occurs. Although these characterizations are very different, it is a fundamental fact of monotone induction that both views correspond.

Monotone induction has been extensively studied from an expressivity-theoretic point of view and is a key concept in recursion theory. Its study was started by [Post 1943] and was continued in many later studies such as [Spector 1961; Moschovakis 1974a; Aczel 1977]. In these formal studies, a monotone inductive definition was

often formalized by a formula $\varphi(\bar{x}, X)$ where X is a predicate variable of arity n with only positive occurrences in φ and \bar{x} a tuple of n variables. This formula encodes the set of rules of an inductive definition by specifying all the conditions under which tuple \bar{x} belongs to the defined predicate X . For instance, for the transitive closure example above we have:

$$\varphi_{\text{trans}}((x, y), T_G) := G(x, y) \vee \exists z(T_G(x, z) \wedge T_G(z, y)).$$

Given a structure I , the formula $\varphi(\bar{x}, X)$ characterises an operator $\Gamma_{\varphi(\bar{x}, X)}$ mapping a relation R to the relation R' consisting of tuples \bar{a} such that $\varphi(\bar{a}, R)$ is true in I . For positive formulas, this operator is monotone, i.e., preserves the subset relation \subseteq , and has a least fixpoint, which is both the least set closed under application of the operator (i.e., the least pre-fixpoint of $\Gamma_{\varphi(\bar{x}, X)}$, i.e., the least R such that $\Gamma_{\varphi(\bar{x}, X)}(R) \subseteq R$) and the fixpoint obtained by iterating the operator on the empty relation. A logic to represent monotone inductive definitions is the least fixpoint logic FO(LFP) (see, e.g. [Ebbinghaus and Flum 1999]).

Mathematicians also define concepts through non-monotone inductive rules, which add elements to the defined relation given that the *absence* of other elements has been established. An example is the fourth rule of Definition 2.1:

$$I \models \neg\varphi \text{ if } I \not\models \varphi.$$

Definition 2.1 is an example of an inductive definition over a well-founded order. Such definitions describe the membership of an element in the defined relation in terms of the presence or absence of elements in the defined relation that are strictly smaller with respect to some well-founded (pre-)order. By applying the rules to the minimal elements and then iterating them for higher levels, possibly a transfinite number of times, the defined predicate can be constructed, independent of whether the inductive rules are monotone or non-monotone. This shows that for this type of induction, the upward interpretation of a definition continues to hold, under the proviso that rules should be applied along the well-founded order, i.e., rules defining higher elements in the order are applied after the lower levels of the relation have been constructed. On the other hand, the *downward* interpretation is not longer correct, i.e., in general the defined set is not the least set closed under the rules. Consider the following (simplified) definition:

Definition 2.3. The set of even numbers is defined by induction over the standard order of the natural numbers:

—0 is an even number;

— $n + 1$ is an even number if n is not an even number.

This set of rules does not have a least set closed under the rules. In particular, $\{0, 2, 4, 6, \dots\}$ and $\{0, 1, 3, 5, 7, \dots\}$ are both minimal sets closed under the rules (i.e., they both contain 0 and the successor of each of the numbers not in the set). The operator of the corresponding formula

$$\varphi_{\text{even}}(x, E) := x = 0 \vee \exists y(x = s(y) \wedge \neg E(y)), \quad (1)$$

is non-monotone, has many minimal but no least pre-fixpoints but has a unique fixpoint, the set of even numbers.

Definitions over well-founded (pre-)orders are very frequent in mathematics. Examples are all definitions over ordinal numbers, such as the ordinal powers of a monotone operator. Another example is that of the *rank* of an element in a well-founded order, defined as the least ordinal strictly larger than the ranks of all lesser elements. This form of inductive definition is strongly related to the principle of proof by induction over the element relation \in of set theory, which has been studied extensively in proof theory [Pohlers 1989]. In Section 6, we will present a general formalization of inductive definitions over a well-founded order as a subclass of definitions in ID-logic.

Induction over a well-founded order is based on a fundamental and widely used mathematical mechanism. A common way to extend a mathematical theory is by introducing a new concept and defining it in terms of the existing concepts of the theory. Here, it does not matter whether the defined concept depends positively or negatively on the already existing concepts. An essential property of such an extension is that it has no impact on the properties of the original concepts; i.e., all properties of the original concepts remain exactly as before the extension. We will call this the *principle of definitional extension*. In [Schlipf 1995b], it was called the principle of stratification and was investigated there in the context of several semantics of logic programming. Also here in this paper, we will present several results formalizing this principle.

The principle of induction over a well-founded order is clearly a fine-grained, iterative application of definitional extensions, where the inclusion of a potential element in the defined set is defined non-recursively in terms of the inclusion of strictly smaller domain elements in the set. A further generalization of this idea is found in *iterated inductive definitions*. Just like induction over a well-founded order, the basic idea underlying iterated induction is to iterate basic construction steps over a segment of ordinals or a well-founded order, but here the basic construction steps are monotone inductions.

A nice example of an iterated inductive definition is the definition of a *stable theory* of some propositional theory T , presented in [Marek 1989]¹. Intuitively, the stable set is an extension of the deductive closure $CN(T)$ of T with a set of formulas of modal logic, constructed by closing T under the inference rules of propositional logic augmented with two additional inference rules:

$$\frac{\vdash \psi}{\vdash K\psi} \quad \text{and} \quad \frac{\not\vdash \psi}{\vdash \neg K\psi}.$$

Observe that the second rule is non-monotone.

Formally, let τ_0 be the vocabulary of T , and for each $n > 0$, $\tau_n := \tau_{n-1} \cup \{K(\varphi) \mid \varphi \text{ is a propositional formula over } \tau_{n-1}\}$. Define the modal depth $l(\varphi)$ of a modal formula φ as the least n such that φ is a propositional formula over τ_n . Equivalently, the modal depth is the maximal nesting depth of the modal operator in φ . The propositional language over τ_n consists of all modal formulas with modal depth $\leq n$. Then we define

$$S_0 = CN(T),$$

¹The original definition in [Moore 1983] defines a stable set through a fixpoint equation.

computed over τ_0 , and for each $n > 0$,

$$S_n = Cn(T \cup \{K(\varphi) \mid \varphi \in S_{n-1}\} \cup \{\neg K(\varphi) \mid \varphi \notin S_{n-1}\}),$$

computed over τ_n .

Clearly, each S_n is a set of formulas of modal depth $\leq n$. There is an easy inductive argument to see that a formula ϕ of modal depth $\leq n$ belongs to S_{n+1} iff it belongs to S_n . Indeed, ϕ belongs to S_{n+1} iff it can be proven from $T' = T \cup \{K(\varphi) \mid \varphi \in S_n\} \cup \{\neg K(\varphi) \mid \varphi \notin S_n\}$. Observe that theory T' can be split up in a theory T'_1 over τ_n and a theory T'_2 over $\tau_{n+1} \setminus \tau_n$. Since ϕ is over τ_n , it can be proven from T' iff it can be proven from $T'_1 = T \cup \{K(\varphi) \mid \varphi \in S_n \wedge l(\varphi) \leq n-1\} \cup \{\neg K(\varphi) \mid \varphi \notin S_n \wedge l(\varphi) \leq n-1\}$. By the induction hypothesis, T'_1 is exactly $T \cup \{K(\varphi) \mid \varphi \in S_{n-1}\} \cup \{\neg K(\varphi) \mid \varphi \notin S_{n-1}\}$. By definition of S_n , ϕ belongs to S_n iff it can be proven from T'_1 iff it can be proven from T' iff ϕ belongs to S_{n+1} .

It follows that the sequence $(S_n)_{n \in \mathbb{N}}$ is monotonically increasing. The limit of this sequence is called the stable set of T . The stable theory of T can be shown to be the set of all modal formulas φ such that for the collection W of models of T and for each model $M \in W$, it holds that $W, M \models \varphi$. Intuitively, the stable set can be viewed as the set of modal formulas believed by an ideally rational agent with perfect introspection whose base beliefs are represented by T .

The logical study of iterated induction was started in [Kreisel 1963] and extended in later studies of so-called Iterated Inductive Definitions (IID) in [Feferman 1970], [Martin-Löf 1971], and [Buchholz et al. 1981]. The IID formalism defined in [Feferman 1970; Buchholz et al. 1981] is a formalism to define sets of natural numbers through iterated induction. To represent an iterated inductive definition of a set H , one associates with each natural number an appropriate *level index*, an ordinal number. This level index can be understood as the index of the sub-definition which determines whether the number belongs to the defined set or not. The iterated inductive definition is described by a finite parametrized formula $\varphi(n, x, P, H)$, where n represents a level index, x is a natural number, P is a unary predicate variable with only positive occurrences in φ and ranging over natural numbers, and H is the defined relation represented as a binary predicate ranging over tuples (n, x) of natural numbers x and their level indices n . The formula $\varphi(n, x, P, H)$ encodes that n is the level index of x , and x can be derived (using the inductive definition with level index n) from the set P and the restriction of H to tuples with level index $< n$.

We illustrate IID by sketching how to encode the definition of a stable theory. Let us assume some Gödel numbering of modal formulas. For each modal formula ϕ , denote its Gödel number by $|\phi|$. The IID formula $\varphi(n, x, P, H)$ encodes that x is the Gödel number of a formula ψ such that n is $l(\psi)$, the modal depth of ψ , and one of the following conditions hold:

- there exists a ψ' such that $\psi = K(\psi')$ and $H(n-1, |\psi'|)$ holds, or
- there exists a ψ' such that $\psi = \neg K(\psi')$ and $\neg H(n-1, |\psi'|)$ holds, or
- there is an instance of an inference rule $\frac{\vdash \psi_1, \dots, \vdash \psi_n}{\vdash \psi}$ of propositional logic and for each $i, 1 \leq i \leq n$, either $l(\psi_i) < n$ and $H(l(\psi_i), |\psi_i|)$ holds or $l(\psi) = n$ and $P(|\psi_i|)$ holds.

The first two items encode the modal inference rules. Using standard methods of Gödel numbering, the above specification can be encoded in FO and leads to a formula in which P occurs only positively.

Using $\varphi(n, x, P, H)$, the set H is characterized by two axioms. The first one expresses that H is closed under φ :

$$\forall n \forall x (\varphi(P(\sigma)/H(n, \sigma)) \rightarrow H(n, x)).$$

In this formula, $\varphi(P(\sigma)/H(n, \sigma))$ denotes the formula obtained from φ by substituting $H(n, \sigma)$ for each expression $P(\sigma)$ in φ .

The second axiom is a second-order axiom expressing that for each n , the subset $\{x \mid (n, x) \in H\}$ of \mathbb{N} is the least set of natural numbers closed under φ :

$$\forall n \forall P [\forall x (\varphi \rightarrow P(x)) \rightarrow \forall x (H(n, x) \rightarrow P(x))].$$

The two axioms axiomatise H as the set of all pairs (n, x) where x belongs to the set defined by iterated induction, and n is its level index.

2.1 A logic of inductive definitions

We have now introduced the different sorts of inductive definitions that we want to formalize in ID-logic. A definition in ID-logic will be represented simply as a set of rules of the form:

$$\forall \bar{x} (P(\bar{x}) \leftarrow \psi),$$

where P is a relational symbol defined by the definition, and ψ an arbitrary first-order formula. For example, the non-monotone definition of even numbers will be represented by the set:

$$\left\{ \begin{array}{l} \forall x (E(x) \leftarrow x = 0), \\ \forall x (E(s(x)) \leftarrow \neg E(x)) \end{array} \right\}. \quad (2)$$

From a representational point of view, this syntax has several interesting features. First, formalizations of definitions in ID-logic preserve the rule-based structure of informal definitions in mathematics. Second, this syntax includes simultaneous inductive definitions. Consider for example the following simultaneous inductive definition of even and odd numbers:

$$\left\{ \begin{array}{l} \forall x (E(x) \leftarrow x = 0), \\ \forall x (E(s(x)) \leftarrow O(x)), \\ \forall x (O(s(x)) \leftarrow E(x)) \end{array} \right\}.$$

Third, the logic offers a uniform formalization of the above mentioned types of definitions. Syntax and semantics of ID-logic are designed for uniform formalization of non-inductive (recursion-free) definitions, positive or monotone inductive definitions, definitions over well-founded orders and iterated inductive definitions.

A feature of the representation of non-monotone definitions in ID-logic is that, in contrast to e.g. IID, no well-founded order or level indices have to be specified. Intuitively, it is not so difficult to see that this order or this level indexing is not really needed to construct the defined relation(s). For example, in Definition 2.1, the satisfaction relation \models was defined by induction on the subformula order, but could have been equivalently defined by induction on two other well-founded strict orders,

those induced by the length or by the depth of formulas. Indeed, the formulas that are mentioned in the condition of rules of Definition 2.1 are strict subformulas and have strictly smaller length and depth than formulas in the head. The point is that a mathematician has a certain liberty in choosing the order associated to a definition. This order should be well-founded and a potential element of the defined relation(s) mentioned in the head of an inductive rule should be strictly larger than those on which it depends, i.e., those mentioned in the condition of the rule. In this way, it is guaranteed that the definition is well-behaved, in the sense that the defined relation can be constructed along this dependency relation. However, the defined relation does not depend on which order is chosen. As we will see in Section 4, the well-founded model construction delays the application of rules defining an atom until enough information has been established about the atoms on which it depends. In other words, the well-founded model construction performs an iterated induction along the implicit dependency order encoded in the rules. This fact can be viewed as the contribution of the well-founded model construction to the semantical study of the above non-monotone forms of inductive definitions.

The generality and uniformity of definitions in ID-logic also causes problems. In mathematics, not every set of basic and inductive rules defines a relation. Similarly, not every definition formula expresses a correct definition. For example, in the natural numbers, the definition

$$\{ \forall x (E(x) \leftarrow \neg E(s(x))) \} \quad (3)$$

induces a dependency of an atom $E(n)$ on $E(n+1)$ and would correspond to an “informal” definition of the set of even numbers by induction on \geq , the inverse of the standard order of the natural numbers. But this is not a well-founded order. The construction of the set is ill-defined because there are no minimal elements in this order and therefore, there are no base cases of the induction. In mathematics, such definitions are errors. This matches with ID-logic’s treatment of definition formulas such as (3): the well-founded model construction for this definition fails and the definition is inconsistent.

In summary, monotone induction, induction over a well-founded order and iterated induction are well-known and frequently used forms of induction in mathematics. Mathematicians often follow certain linguistic conventions to express such inductive definitions by sets of base rules and inductive rules. The definition construct in ID-logic is designed to allow a uniform, faithful translation of such informal definitions in the logic.

The close correspondence between definition formulas in ID-logic and definitions in mathematics, is important for knowledge representation and modelling. The technical details of well-founded model construction are non-trivial (although simpler semantical theories hold for most forms of inductive definitions, see Section 6), but to understand or represent a definition in ID-logic, it is not necessary to mentally compute this well-founded model. To a great deal, we can rely on our understanding of its informal semantics.

2.2 Another form of non-monotone induction: Inflationary Induction.

In order to extend the theory of monotone inductive definitions to the class of all formulas, Moschovakis [Moschovakis 1974b] proposed to associate with an arbitrary

formula $\varphi(\bar{x}, X)$ (possibly non-positive) the operator $\Gamma'_{\varphi(\bar{x}, X)}$, where

$$\Gamma'_{\varphi(\bar{x}, X)}(R) := \Gamma_{\varphi(\bar{x}, X)}(R) \cup R.$$

Operator $\Gamma'_{\varphi(\bar{x}, X)}$ is not monotone, but it is *inflationary*, that is, for every R , $R \subseteq \Gamma'_{\varphi(\bar{x}, X)}(R)$. Thus, by iterating this operator starting at the empty relation, an ascending sequence can be constructed. This sequence eventually reaches a fixpoint of $\Gamma'_{\varphi(\bar{x}, X)}$. This fixpoint was later called the *inflationary fixpoint*, and the corresponding logic FO(IFP) was introduced [Gurevich and Shelah 1986]. This logic introduces inflationary, and its dual, deflationary, fixpoint constructs. The inflationary fixpoint logic played an important role in descriptive complexity theory and has been used to characterize the complexity class PTIME [Immerman 1986; Livchak 1983; Vardi 1982]. FO(IFP) is well-known to be a very expressive logic. For some applications of inflationary and deflationary induction, we address the reader to [Graedel and Kreutzer 2003].

Inflationary induction and iterated induction (as formalized in ID-logic) are different extensions of monotone induction. For instance, inflationary induction applied on the non-monotone Definition 2.3 yields the set of all natural numbers. More precisely, this is the set obtained with one application of the operator $\Gamma'_{\varphi_{\text{even}}}$, where φ_{even} is given in formula (1).

Our argument to extend FO with induction over well-founded order and iterated induction rather than inflationary induction stems from knowledge representation. Induction over well-founded order and iterated induction have with many applications in mathematics. Part of our project is to demonstrate that these forms of inductive definitions have also many applications outside mathematics, in knowledge representation and computational logic. In comparison, *natural* applications of inflationary induction in mathematics and computational logic seem to be quite rare. Although inflationary induction is expressive, in practice rephrasing even simple iterated inductive definitions through inflationary induction may be extremely difficult. The author of [Van Gelder 1993] discusses the case of the complement NT_G of the transitive closure of a graph G . The obvious way to define this in ID-logic is:

$$\left\{ \begin{array}{l} \forall x \forall y (T_G(x, y) \leftarrow G(x, y) \vee \exists z (T_G(x, z) \wedge T_G(z, y))), \\ \forall x \forall y (NT_G(x, y) \leftarrow \neg T_G(x, y)) \end{array} \right\}.$$

This is a simple iterated inductive definition with 2 levels. On the other hand, according to [Van Gelder 1993], it was “*a significant research achievement*” when a set of (function-free) logic programming rules (rules with a conjunction of literals as body) was discovered expressing the same relation under the inflationary semantics. The simplest known solution is a complex set of rules including several intermediate relations and it is far from obvious how computing the inflationary fixpoint of this program constructs the relation.

3. PRELIMINARIES

3.1 Preliminaries from Logic

We begin by fixing notation and terminology for the basic syntactic and semantic notions related to first- and second-order logic.

We assume an infinite supply of distinct symbols, which are classified as follows:

1. Logical symbols:
 - a) Parentheses: $(,)$;
 - b) Logical connectives: \wedge, \neg ;
 - c) Existential quantifier: \exists ;
 - d) Binary equality symbol: $=$ (optional);
 - e) Two propositional symbols: **t** and **f**.
2. Non-logical symbols:
 - a) countably many object symbols. Object symbols are denoted by low-case letters;
 - b) for each positive integer $n > 0$, countably many n -ary function symbols of arity n . Function symbols are denoted by low-case letters;
 - c) for each positive integer n , countably many n -ary relation symbols, also called predicate or set symbols of arity n . We use upper-case letters to denote predicates.

As usual, we identify object symbols with 0-ary function symbols and propositional symbols with predicate symbols of arity 0.

Remark 3.1. In most parts of this paper, we do not make a formal distinction between variable and constant symbols. Symbols occurring free in a formula can be viewed as constants; symbols in the scope of a quantifier can be viewed as variables. In examples, we tend to quantify over x, y, X, Y , and leave c, g, f and P, Q free and treat them as constants.

We define a vocabulary as any set of non-logical symbols. We denote vocabularies by $\tau, \tau_{\Delta}^{\circ}, \dots$. We shall denote the set of function symbols of τ by τ_{fn} , and we use $\sigma, \sigma_1, \sigma_2$ etc., to refer to an arbitrary symbol of the vocabulary. We write $\bar{\sigma}$ to denote a sequence of symbols $(\sigma_1, \sigma_2, \dots)$ or, depending on the context, simply the set of symbols $\{\sigma_1, \sigma_2, \dots\}$. Likewise, \bar{X} denotes a sequence or a set of relational symbols (i.e, set variables or constants), and \bar{x} is used to denote a sequence or a set of object symbols, etc..

A *term* is defined inductively as follows:

- an object symbol is a term;
- if t_1, \dots, t_n are terms and f is an n -ary function symbol, where $n \geq 1$, then $f(t_1, \dots, t_n)$ is a term.

A formula is defined by the following induction:

- if P is an n -ary predicate constant or variable, and t_1, \dots, t_n are terms then $P(t_1, \dots, t_n)$ is a formula, called an *atomic formula* or simply an *atom*;
- if ϕ, ψ are formulas, then so are $\neg\phi, \phi \wedge \psi$;
- if x is an object symbol, f a function symbol, X is a predicate symbol and ϕ is a formula, then $\exists x \phi, \exists f \phi$ and $\exists X \phi$ are formulas.

A bounded occurrence of symbol σ in formula ϕ is an occurrence of σ in a subformula $\exists\sigma\psi$ of ϕ . A free occurrence of σ in ϕ is an unbounded occurrence. The set of symbols which occur free in ϕ is denoted $\text{free}(\phi)$. A relation symbol X

has a negative (positive) occurrence in formula F if X has a free occurrence in the scope of an odd (even) number of occurrences of the negation symbol \neg .

A formula ϕ is a formula *over vocabulary* τ if its free symbols belong to τ ($free(\phi) \subseteq \tau$). We use $SO[\tau]$ to denote the set of all formulas over τ ; and we use $FO[\tau]$ to denote the set of first-order formulas over τ , that is those without quantified predicate or function variables.

We use $(\phi \vee \psi)$, $(\phi \supset \psi)$, $(\phi \equiv \psi)$, $\forall x \phi$, $\forall f \phi$ and $\forall X \phi$, in the standard way, as abbreviations for the formulas $\neg(\neg\phi \wedge \neg\psi)$, $\neg(\phi \wedge \neg\psi)$, $\neg(\phi \wedge \neg\psi) \wedge \neg(\psi \wedge \neg\phi)$, $\neg\exists x (\neg\phi)$, $\neg\exists f (\neg\phi)$, $\neg\exists X (\neg\phi)$, respectively.

Having defined the basic syntactic concepts, we define the semantic concepts. Let A be a non-empty set. A *value* for an n -ary relation (function) symbol σ of vocabulary τ in A is an n -ary relation (function) in A . A value for a 0-ary function symbol, i.e., an object constant or variable, is an element of the domain A . A value for a 0-ary relation symbol Y is either \emptyset or $\{()\}$, the singleton of the empty tuple. We identify these two values with *false*, respectively *true*. The value of the equality symbol is always the identity relation on A . The value of **t** is $\{()\}$ (true) and the value of **f** is \emptyset (false).

A *structure* I for a given vocabulary τ (in short, a τ -*structure*) is a tuple of a domain $dom(I)$, which is a non-empty set, and a mapping of each symbol σ in τ to a value σ^I in $dom(I)$. If $\sigma \in \tau$ and I is a τ -structure, we say that I *interprets* σ . We also use letters J, K, L, M to denote structures. Given I , τ_I denotes the set of symbols interpreted by I .

Let us introduce notation for constructing and modifying structures with a shared domain A . Let I be a τ -structure, and $\bar{\sigma}$ be a tuple of symbols not necessarily in τ . Structure $I[\bar{\sigma} : \bar{v}]$ is a $\tau \cup \bar{\sigma}$ -structure, which is the same as I , except symbols $\bar{\sigma}$ are interpreted by values \bar{v} in $dom(I)$. Given a τ -structure I and a sub-vocabulary $\tau' \subseteq \tau$, the restriction of I to the symbols of τ' is denoted $I|_{\tau'}$. Vice versa, a structure I is called an extension of I_o if $I_o = I|_{\tau_{I_o}}$.

Let t be a term, and let I be a structure interpreting each symbol in t . We define the *denotation* t^I of t under I by the usual induction:

- if t is an object symbol σ , then t^I is σ^I , the value of σ in I ;
- if $t = f(t_1, \dots, t_n)$, then $t^I := f^I(t_1^I, \dots, t_n^I)$.

Next we define the *satisfaction* or *truth* relation \models . Let I be a structure and let ϕ be a formula such that each free symbol in ϕ is interpreted by I . We define $I \models \phi$ (in words, ϕ is true in I , or I satisfies ϕ) by the following standard induction:

- $I \models X(t_1, \dots, t_n)$ if $(t_1^I, \dots, t_n^I) \in X^I$;
- $I \models \psi_1 \wedge \psi_2$ if $I \models \psi_1$ if $I \models \psi_2$;
- $I \models \neg\psi$ if $I \not\models \psi$;
- $I \models \exists\sigma \psi$ if for some value v of σ in the domain $dom(I)$ of I , $I[\sigma : v] \models \psi$.

Note that the truth of a formula ϕ is only well-defined in a structure interpreting each free symbol of ϕ . We shall denote the truth value of ϕ in I by ϕ^I , i.e., if $I \models \phi$ then ϕ^I is true ($\{()\}$) and otherwise, it is false (\emptyset).

Sometimes, we wish to investigate the truth value of a formula ϕ as a function of the values assigned to a specific tuple of symbols $\bar{\sigma}$. We then call these symbols

the *parameters* of ϕ and denote the formula by $\phi(\bar{\sigma})$. Let I be some structure and let \bar{v} be a tuple of values for $\bar{\sigma}$ in the domain $\text{dom}(I)$. We often write $I \models \phi[\bar{v}]$ to denote $I[\bar{\sigma} : \bar{v}] \models \phi$.

Let X be an n -ary relation symbol and A some domain. We define a *domain atom of X in A* as any pair (X, \bar{d}) where \bar{d} is an arbitrary n -tuple of elements A , and denote such a domain atom by $X[\bar{d}]$. For I a structure with domain A , the value of $X[\bar{d}]$ in I is true if $\bar{d} \in X^I$; otherwise it is false. For a vocabulary τ , we define At_A^τ as the set of all domain atoms in domain A over relation symbols in τ .

Suppose we are given a structure I with domain $\text{dom}(I)$, a tuple \bar{x} of n variables and a first-order formula $\phi(\bar{x})$ such that all its free symbols not in \bar{x} are interpreted by I . The relation *defined by $\phi(\bar{x})$ in the structure I* is defined as follows:

$$R := \{ \bar{a} \in (\text{dom}(I))^n \mid I \models \phi[\bar{a}] \}.$$

We call R *first-order definable* in I . In this paper, we study inductive and non-monotone inductive definability. In this context, defined relations are not, in general, first-order definable.

3.2 Preliminaries from Set and Lattice Theories

3.2.1 Orders, Lattices, operators and fixpoints. A *pre-ordered set* is a structured set $\langle W, \leq \rangle$, where W is an arbitrary set and \leq is a pre-order on W , i.e., a reflexive and transitive binary relation. As usual, $x < y$ is a shorthand for $x \leq y \wedge y \not\leq x$. A *pre-well-founded set* is a pre-ordered set where \leq is a pre-order such that every non-empty set $S \subseteq W$ contains a minimal element, i.e., an element x such that for each $y \in S$, if $y \leq x$ then $x \leq y$. Equivalently, it is a set without infinite descending sequence of elements $x_0 > x_1 > x_2 > \dots$.

A *partially ordered set*, or simply *poset*, is an asymmetric pre-ordered set $\langle W, \leq \rangle$, i.e., one such that $x \leq y$ and $y \leq x$ implies $x = y$. A *well-founded set* is a pre-well-founded poset.

A *lattice* is a poset $\langle L, \leq \rangle$ such that every finite set $S \subseteq L$ has a least upper bound $\text{lub}(S)$, the *supremum of S* , and a greatest lower bound $\text{glb}(S)$, the *infimum of S* . A lattice $\langle L, \leq \rangle$ is *complete* if every (not necessarily finite) subset of L has both a supremum and an infimum. Consequently, a complete lattice has a least element (\perp) and a greatest element (\top). An example of a complete lattice is the power set lattice $\langle \text{Pow}(A), \subseteq \rangle$ of some set A . For any set S of elements of this lattice (i.e., for any set S of subsets of A), its least upper bound is the union of these elements, $\text{lub}(S) = \cup S$. Thus, the greatest element \top of $\langle \text{Pow}(A), \subseteq \rangle$ is $\cup \text{Pow}(A)$, which is A . Similarly, $\text{glb}(S) = \cap S$, and the least element \perp of this lattice is $\cap \text{Pow}(A)$, which is \emptyset .

Given a lattice $\langle L, \leq \rangle$, an operator $\Gamma : L \rightarrow L$ is *monotone* with respect to \leq if $x \leq y$ implies $\Gamma(x) \leq \Gamma(y)$. Operator Γ is *non-monotone*, if it is not monotone. A *pre-fixpoint* of Γ is a lattice element x such that $\Gamma(x) \leq x$. The following theorem was obtained by Tarski in 1939 and is sometimes referred to as the Knaster-Tarski theorem because it improves their earlier joint result. The theorem was published in [Tarski 1955], and it is one of the basic tools to study fixpoints of operators on lattices.

THEOREM 3.2 EXISTENCE OF A LEAST FIXPOINT. *Every monotone operator over*
ACM Transactions on Computational Logic, Vol. V, No. N, October 2006.

a complete lattice $\langle L, \leq \rangle$ has a complete lattice of fixpoints (and hence a least fixpoint $\text{lfp}(\Gamma)$ and greatest fixpoint $\text{gfp}(\Gamma)$).

This least fixpoint $\text{lfp}(\Gamma)$ is the least pre-fixpoint of Γ and is the supremum of the sequence $(x^\xi)_\xi$ which is defined inductively

$$x^\xi := \Gamma(x^{<\xi}), \text{ and } x^{<\xi} := \text{lub}\{x^\eta \mid 0 \leq \eta < \xi\}.$$

Notice that $x^{<0}$ is, by definition, \perp .

An operator Γ is *anti-monotone* if $x \leq y$ implies $\Gamma(y) \leq \Gamma(x)$. The square $\Gamma^2 = \Gamma \circ \Gamma$ of an anti-monotone operator is monotone.

An *oscillating pair* of an operator Γ is a pair (x, y) such that $\Gamma(x) = y$ and $\Gamma(y) = x$. An anti-monotone operator Γ in a complete lattice has a maximal oscillating pair (x, y) , i.e., for any oscillating pair (x', y') , it holds that $x \leq x'$ and $y' \leq y$. Since (y, x) is also an oscillating pair, it follows that $x \leq y$. Moreover, since each fixpoint z of Γ corresponds to an oscillating pair (z, z) , it follows that $x \leq z \leq y$. If in addition $x = y$ then x is the unique fixpoint of Γ .

The maximal oscillating pair (x, y) of Γ can be constructed by an alternating fixpoint computation. Define four sequences $(x^\xi)_\xi, (x^{<\xi})_\xi, (y^\xi)_\xi, (y^{<\xi})_\xi$ by the following transfinite induction:

- $x^{<\xi} = \text{lub}(\{x^\eta : \eta < \xi\})$,
- $x^\xi = \Gamma(y^{<\xi})$,
- $y^{<\xi} = \text{glb}(\{y^\eta : \eta < \xi\})$,
- $y^\xi = \Gamma(x^{<\xi})$.

Note that $x^{<0} = \perp$ and $y^{<0} = \top$. It can be shown that for each ξ , $x^{<\xi} \leq x^\xi \leq y^\xi \leq y^{<\xi}$. The following theorem holds.

THEOREM 3.3. [Van Gelder 1993] *The sequence $(x^\xi)_\xi$ is ascending and its supremum is $\text{lfp}(\Gamma^2)$. The sequence $(y^\xi)_\xi$ is descending and its infimum is $\text{gfp}(\Gamma^2)$. The pair $(\text{lfp}(\Gamma^2), \text{gfp}(\Gamma^2))$ is the maximal oscillating pair of Γ .*

In the sequel, we denote the maximal oscillating pair of an anti-monotone operator G by $\text{OSC}(G)$.

3.2.2 Lattice Congruences. Let $\langle L, \leq \rangle$ be a complete lattice and let \cong be an equivalence relation (i.e., a reflexive, symmetric and transitive relation) on L . For any $x \in L$, we denote its equivalence class $\{y \in L \mid x \cong y\}$ by $|x|$. The collection of equivalence classes is denoted by $|L|$. The relation \cong can be extended to tuples: $(x_1, \dots, x_n) \cong (y_1, \dots, y_n)$ if $x_1 \cong y_1$ and \dots and $x_n \cong y_n$. It is extended to subsets of L by defining for all $S, S' \subseteq L$: $S \cong S'$ if for each $x \in S$ there exists $x' \in S'$ such that $x \cong x'$ and vice versa, for each $x' \in S'$ there exists $x \in S$ such that $x \cong x'$. We sometimes call an element of $\tilde{x} \in |L|$ a *witness* of \tilde{x} .

An equivalence relation \cong on L is called a *lattice congruence* of $\langle L, \leq \rangle$ if for each pair $S, S' \subseteq L$, $S \cong S'$ implies that $\text{lub}(S) \cong \text{lub}(S')$ and $\text{glb}(S) \cong \text{glb}(S')$. We can define a binary relation \leq on $|L|$: for all $\tilde{x}, \tilde{y} \in |L|$, define $\tilde{x} \leq \tilde{y}$ if for some $x \in \tilde{x}, y \in \tilde{y}$: $x \leq y$. It can be shown easily that if \cong is a lattice congruence, then the structure $\langle |L|, \leq \rangle$ is a complete lattice.

A lattice congruence \cong on L defines a pre-order \leq_\cong on L , defined as $x \leq_\cong y$ iff $|x| \leq |y|$ iff $\exists x', y' : x \cong x' \leq y' \cong y$.

We say that an operator $\Gamma : L^m \rightarrow L^n$ preserves \cong if for any pair of $\bar{x}, \bar{y} \in L^m$, $\bar{x} \cong \bar{y}$ implies $\Gamma(\bar{x}) \cong \Gamma(\bar{y})$. When Γ preserves \cong , we define its quotient $|\Gamma|$ as the operator from $|L|^m$ to $|L|^n$ mapping every $\tilde{x} \in |L|^m$ to $|\Gamma(\bar{x})|$ where \bar{x} is an arbitrary witness of \tilde{x} .

The following straightforward proposition describes the relationships between Γ and $|\Gamma|$.

PROPOSITION 3.4. *Assume that $\Gamma : L \rightarrow L$ is an operator which preserves \cong .*

- (a) *If Γ is (anti-)monotone, then $|\Gamma|$ is (anti-)monotone.*
- (b) *If Γ is monotone then $|\text{lfp}(\Gamma)| = \text{lfp}(|\Gamma|)$ and $|\text{gfp}(\Gamma)| = \text{gfp}(|\Gamma|)$.*
- (c) *If Γ is anti-monotone, then $|\text{OSC}(\Gamma)| = \text{OSC}(|\Gamma|)$.*

This proposition has a straightforward extension to operators of more arguments.

3.2.3 *Structure lattices.* The type of lattices that play a central role in this paper are the sets of structures that extend a given structure. For a given vocabulary τ and structure K_o such that $\tau_{K_o} \subseteq \tau$, define $\mathcal{S}_{K_o}^\tau$ as the set of τ -structures that extend K_o , i.e., the set of τ -structures I such that $I|_{\tau_{K_o}} = K_o$.

For any pair I_1, I_2 of τ -structures, define $I_1 \sqsubseteq I_2$ if for each interpreted relation symbol X , $X^{I_1} \subseteq X^{I_2}$. This relation is reflexive and transitive. It is not asymmetric, since I_1 and I_2 may be identical on all relation symbols but different on constant or function symbols. However, if K_o interprets all function symbols of τ , that is, if $\tau_{\text{fn}} \subseteq \tau_{K_o}$, then \sqsubseteq is a complete lattice order in the set $\mathcal{S}_{K_o}^\tau$. Its least element is the structure $\perp_{K_o} := K_o[\bar{X} : \emptyset]$ assigning the empty relations to all symbols X in $\tau \setminus \tau_{K_o}$ and its largest element \top_{K_o} is the structure assigning the Cartesian product A^n to each n -ary symbol $X \in \tau \setminus \tau_{K_o}$. For any subset $S \subseteq \mathcal{S}_{K_o}^\tau$, we denote its least upper-bound by $\sqcup S$ and its greatest lower-bound by $\sqcap S$.

The lattice $\langle \mathcal{S}_{K_o}^\tau, \leq \rangle$ contains many sub-lattices. In particular, for any structure K extending K_o such that $\tau_{K_o} \subseteq \tau_K \subseteq \tau$, $\langle \mathcal{S}_K^\tau, \sqsubseteq \rangle$ is a sub-lattice of $\langle \mathcal{S}_{K_o}^\tau, \sqsubseteq \rangle$.

In this paper, the family of structure lattices and congruences on them plays an important role.

4. ID-LOGIC

In this section, we present an extension of classical logic with non-monotone inductive definitions. This logic extends the logic defined by the first author in [Denecker 2000].

4.1 Syntax

First, we introduce the notion of a definition. We introduce a new binary connective \leftarrow , called the *definitional implication*. A *definition* Δ is a set of rules of the form

$$\forall \bar{x} (X(\bar{t}) \leftarrow \varphi) \tag{4}$$

where

- \bar{x} is a tuple of object variables,
- X is a predicate symbol (i.e., a predicate constant or variable) of some arity r ,
- \bar{t} is a tuple of terms of length r ,

— φ is an arbitrary first-order formula.

The definitional implication \leftarrow must be distinguished from material implication. A rule $\forall \bar{x} (X(\bar{t}) \leftarrow \varphi)$ in a definition does not correspond to the disjunction $\forall \bar{x} (X(\bar{t}) \vee \neg \varphi)$, but implies it. Note that in front of rules, we allow only universal quantifiers. In the rule (4), $X(\bar{t})$ is called the *head* and φ is the *body* of the rule.

The definitions of bound and free occurrence of a symbol in a formula and of $free(\Delta)$ extend to the case of a rule and a definition Δ (see Section 3.1). Δ is a definition over τ if $free(\Delta) \subseteq \tau$.

Example 4.1. The following expression is a simultaneous definition of the sets of even and odd numbers on the structure of the natural numbers with zero and the successor function:

$$\left\{ \begin{array}{l} \forall x (E(x) \leftarrow x = 0), \\ \forall x (E(s(x)) \leftarrow O(x)), \\ \forall x (O(s(x)) \leftarrow E(x)) \end{array} \right\}. \quad (5)$$

Example 4.2. This is the definition of the transitive closure of a directed graph G :

$$\left\{ \begin{array}{l} \forall x \forall y (T(x, y) \leftarrow G(x, y)), \\ \forall x \forall y (T(x, y) \leftarrow \exists z (T(x, z) \wedge T(z, y))) \end{array} \right\}. \quad (6)$$

A *defined symbol* of Δ is a relation symbol that occurs in the head of at least one rule of Δ ; other relation, object and function symbols are called *open*. In Example 4.2, T is a defined predicate symbol, and G is an open predicate symbol. We call Δ a *positive* definition if no defined predicate X has a negative occurrence in the body of a rule of Δ . The definitions in Example 4.1 and Example 4.2 are positive.

Let Δ be a definition over τ . The subset of defined symbols of definition Δ is denoted τ_{Δ}^d . The set of open symbols of Δ in τ is denoted τ_{Δ}^o . The sets τ_{Δ}^d and τ_{Δ}^o form a partition of τ , i.e., $\tau_{\Delta}^d \cup \tau_{\Delta}^o = \tau$, and $\tau_{\Delta}^d \cap \tau_{\Delta}^o = \emptyset$.

Now we are ready to define the well-formed formulas of the logic. A *well-formed formula* of the Logic for Non-Monotone Inductive Definitions, briefly a *ID-formula*, is defined by the following induction:

- If X is an n -ary predicate symbol, and t_1, \dots, t_n are terms then $X(t_1, \dots, t_n)$ is a formula.
- If Δ is a definition then Δ is a formula.
- If ϕ, ψ are formulas, then so are $(\neg \phi)$ and $(\phi \wedge \psi)$.
- If ϕ is a formula, then $\exists \sigma \phi$ is a formula.

The definitions of bound and free occurrence of a symbol in a formula further extend to all ID-formulas ϕ . A formula ϕ is an ID-formula over a vocabulary τ if $free(\phi) \subseteq \tau$. We use $SO(ID)[\tau]$ to denote the set of all formulas of our logic over fixed vocabulary τ . The first-order fragment $FO(ID)[\tau]$ is defined in the same way, except that quantification over set and function symbols is not allowed.

Example 4.3. The Peano induction axiom is:

$$\forall P [P(0) \wedge \forall n (P(n) \supset P(s(n))) \supset \forall n P(n)].$$

This axiom can be formulated in ID-logic as:

$$\exists N \left[\left\{ \begin{array}{l} \forall x (N(x) \leftarrow x = 0), \\ \forall x (N(s(x)) \leftarrow N(x)) \end{array} \right\} \wedge \forall x N(x) \right]. \quad (7)$$

The first conjunct in this formula defines the set variable N as the set of the natural numbers through the standard induction. The second conjunct expresses that each domain element is a natural number. An equivalent alternative formalization is:

$$\forall N \left[\left\{ \begin{array}{l} \forall x (N(x) \leftarrow x = 0), \\ \forall x (N(s(x)) \leftarrow N(x)) \end{array} \right\} \supset \forall x N(x) \right]. \quad (8)$$

The equivalence of axioms (7) and (8) follows from the fact that the defined set is unique. The uniqueness is guaranteed by the semantics we define next.

In the sequel, we use $T_{\mathbb{N}}$ to denote the ID-theory consisting of axiom (7) and the two other Peano axioms:

$$\begin{aligned} & \forall n \neg(s(n) = 0), \\ & \forall n \forall m (s(n) = s(m) \supset n = m). \end{aligned}$$

4.2 Semantics

The exposition below is a synthesis of different approaches to the well-founded semantics, in particular those presented in [Van Gelder 1993; Fitting 2002; Denecker et al. 2001]. We begin by defining the operator associated with a definition Δ . We shall assume that definitions are finite sets of rules. The theory can easily be extended to the infinite case (using infinitary logic).

Any definition containing multiple rules with the same predicate in the head can be easily transformed into a definition with only one rule per defined predicate.

Example 4.4. The following definition of even numbers

$$\left\{ \begin{array}{l} \forall x (E(y) \leftarrow y = 0), \\ \forall x (E(s(s(x))) \leftarrow E(x)) \end{array} \right\}$$

is equivalent to this one:

$$\left\{ \forall x (E(y) \leftarrow y = 0 \vee \exists x (y = s(s(x)) \wedge E(x))) \right\}.$$

In general, let Δ be an arbitrary definition with defined relational symbols $\bar{X} := (X_1, \dots, X_n)$. For each defined symbol X of Δ , we define:

$$\varphi_X(\bar{x}) := \exists \bar{y}_1 (\bar{x} = \bar{t}_1 \wedge \varphi_1) \vee \dots \vee \exists \bar{y}_m (\bar{x} = \bar{t}_m \wedge \varphi_m), \quad (9)$$

where \bar{x} is a tuple of new object variables, and $\forall \bar{y}_1 (X(\bar{t}_1) \leftarrow \varphi_1), \dots, \forall \bar{y}_m (X(\bar{t}_m) \leftarrow \varphi_m)$ are the rules of Δ with X in the head. Then Δ is equivalent to the definition Δ_1 consisting of rules $\forall \bar{x} (X(\bar{x}) \leftarrow \varphi_X(\bar{x}))$. The formulas $\varphi_X(\bar{x})$ play an important role in defining the semantics of definitions.

Let Δ be definition over a vocabulary τ .

Definition 4.5. We introduce a unary operator $\Gamma_{\Delta} : \mathcal{I} \mapsto \mathcal{I}$ where \mathcal{I} is the class of all τ -structures. We have $I' = \Gamma_{\Delta}(I)$ iff

- $dom(I) = dom(I')$,
- for each open symbol σ , $\sigma^{I'} = \sigma^I$ and

—for each defined symbol $X \in \tau_{\Delta}^d$,

$$X^{I'} := \{\bar{a} \mid I \models \varphi_X[\bar{a}]\},$$

where φ_X is defined by equation (9).

Let I_o be a structure interpreting the open symbols of Δ in τ . Lattice $\langle \mathcal{S}_{I_o}^{\tau}, \sqsubseteq \rangle$ consists of all τ -structures that extend I_o . Operator Γ_{Δ} is an operator on this lattice. If Δ is a positive definition (no negative occurrences of defined symbols in rule bodies), then Γ_{Δ} will be monotone. The least fixpoint is the limit of the sequence $(I^{\xi})_{\xi}$ which is defined inductively:

$$I^{\xi} := \Gamma_{\Delta}(I^{<\xi}), \text{ and } I^{<\xi} := \bigsqcup \{I^{\eta} \mid 0 \leq \eta < \xi\}.$$

Notice that $I^{<0}$ is, by definition, the bottom element $\perp_{I_o} := I_o[\bar{X} : \emptyset]$ in the lattice.

In general, Γ_{Δ} is a non-monotone operator with no or multiple minimal fixpoints. Iterating the operator starting from the bottom element may oscillate and never reach a fixpoint, or, when it does reach a fixpoint, this fixpoint may not be the intended fixpoint.

Example 4.6. Consider the following propositional definition:

$$\Delta_0 := \left\{ \begin{array}{l} P \leftarrow \mathbf{t}, \\ Q \leftarrow \neg P, \\ Q \leftarrow Q, \\ R \leftarrow \neg Q \end{array} \right\}.$$

Formally, structures of Δ are mappings of the symbols P, Q, R to 0-ary relations. We will represent such a structure in a more traditional way as the set of the propositional symbols that are true (i.e., that are interpreted by $\{()\}$).

Notice that, in definition Δ , Q depends on P and R on Q . In ID-logic this definition is understood as a 3-level iterated inductive definition $(\Delta_1, \Delta_2, \Delta_3)$, where

$$\begin{aligned} \Delta_1 &:= \{P \leftarrow \mathbf{t}\}, \\ \Delta_2 &:= \{Q \leftarrow \neg P, Q \leftarrow Q\}, \\ \Delta_3 &:= \{R \leftarrow \neg Q\}. \end{aligned}$$

By applying iterated induction, we obtain $\{P\}$ for the first level, then \emptyset for the second and $\{R\}$ for the third. Consequently, the intended model of this definition is $\{P, R\}$. On the other hand, twice iterating the operator Γ_{Δ} from the empty structure, yields the fixpoint $\{P, Q\}$ which is also a minimal fixpoint.

The intuition underlying the semantics is to use definitions to perform iterated induction, while following the implicit dependency order given by the rules. We explain how this intuition is formalized in the well-founded semantics. We compute a converging sequence of pairs $(I^{\xi}, J^{\xi})_{\xi \geq 0}$ of τ -structures extending I_o . In each pair, I^{ξ} represents a lower bound to the intended model of Δ extending I_o ; J^{ξ} represents an upper bound: domain atoms true in I^{ξ} can be derived from the definition; atoms false in J^{ξ} cannot be derived; for all atoms false in I^{ξ} and true in J^{ξ} , it is not determined yet whether they can be derived or not. Thus, the pair (I^{ξ}, J^{ξ}) represents approximate information about what can and what cannot be derived from Δ in I_o .

The construction process starts with the pair $(\perp_{I_o}, \top_{I_o})$ of the least and largest element in the lattice $\mathcal{S}_{I_o}^T$. This pair obviously consists of a lower and an upper bound of what can be derived from the definition. Assuming we have obtained a pair (I^ξ, J^ξ) of a safe lower and upper bound, we then apply an operation which transforms this pair into a new pair $(I^{\xi+1}, J^{\xi+1})$ with an improved lower and upper bound. By iterating this operation, a sequence $(I^\xi, J^\xi)_{\xi \geq 0}$ of increasing precision is constructed. The sequence of lower bounds $(I^\xi)_{\xi \geq 0}$ is monotonically increasing and has a limit I (its *lub*); the sequence of upper bounds $(J^\xi)_{\xi \geq 0}$ is monotonically decreasing and has a limit J (its *glb*) such that $I \sqsubseteq J$. The pair of limits (I, J) is the result of the construction and represents the information that can be derived from Δ in the context of the structure I_o . The definition Δ properly defines its defined symbols in I_o if $I = J$, that is, if for each defined domain atom $X[\bar{a}]$, $X[\bar{a}]^I = X[\bar{a}]^J$. If $I = J$, then we will call Δ *total* in I_o and I the *extension of I_o defined by Δ* . If $I \neq J$, then there will be no extension of I_o defined by Δ .

We now explain how a pair (I^ξ, J^ξ) of lower and upper-bound is refined into a new pair $(I^{\xi+1}, J^{\xi+1})$. The idea is to compute the new lower bound $I^{\xi+1}$ and upper bound $J^{\xi+1}$ by monotone induction using the existing bounds (I^ξ, J^ξ) . We cannot use Γ_Δ for this, due to its non-monotonicity, but there is a way.

In general, defined symbols have positive and negative occurrences in the rule bodies $\varphi_X(\bar{x})$. The negative occurrences are responsible for the non-monotone behaviour of the operator Γ_Δ : adding *more* tuples to the value of a negatively occurring defined symbol in $\varphi_X(\bar{x})$ has an anti-monotone effect on the derived relation and may lead to the derivation of *fewer* tuples \bar{a} satisfying this formula. Thus we can eliminate the non-monotonicity of Γ_Δ and set up a monotone induction process using Δ if we *fix* the value of negative occurrences of defined symbols in rule bodies. Suppose we choose a fixed structure M to evaluate the negative occurrences of defined symbols in rule bodies. We can then perform a monotonic derivation process $\perp_{I_o}, K^1, K^2, \dots$ in which each K^{i+1} is derived from Δ by evaluating positive occurrences of defined symbols in each $\varphi_X(\bar{x})$ with respect to K^i and negative occurrences with respect to M . This process will be monotone.

We first choose M to be I^ξ : negative occurrences of defined symbols are interpreted by the lower bound of what can be derived. Thus, during the derivation process of $\perp_{I_o}, K^1, K^2, \dots$, we systematically underestimate the truth of negative occurrences of defined predicates. Due to the anti-monotone effect of negative occurrences of defined symbols on what can be derived, in each stage K^i , too many atoms may be derived. Consequently, the limit of this derivation process yields an upper bound of what can be derived, and we take it to be our new upper bound $J^{\xi+1}$. Second, we choose M to be J^ξ , our best upper bound so far on what can be derived. Thus, during the derivation process $\perp_{I_o}, L^1, L^2, \dots$, we systematically overestimate the truth of negative occurrences of defined symbols, and in each derivation stage K^i , too few atoms are derived. Therefore, the limit of this sequence represents a lower bound of what can be derived and we define it to be $I^{\xi+1}$. We have constructed our new approximating pair $(I^{\xi+1}, J^{\xi+1})$, by two monotone inductions.

It is now easy to see in what sense the well-founded model construction follows the natural dependency order between domain atoms, induced by the rules of a

definition. Assume that at some stage (I^ξ, J^ξ) , the truth of a domain atom $X[\bar{a}]$ has not yet been fixed (i.e., $X[\bar{a}]^{I^\xi} \neq X[\bar{a}]^{J^\xi}$), but the truth values of all atoms on which $X[\bar{a}]$ depends negatively have been derived. In the fixpoint computations K^0, K^1, K^2, \dots with limit $J^{\xi+1}$ and L^0, L^1, L^2, \dots leading to $I^{\xi+1}$, the structures $K^0 = \perp_{I_0} = L^0$ evidently coincide on all atoms and a fortiori, also on those on which $X[\bar{a}]$ depends. This property is preserved during the induction, since the structures I^ξ and J^ξ which are used to evaluate negative occurrences of defined symbols, coincide on all atoms on which $X[\bar{a}]$ depends negatively. Therefore, the new lower and upper-bounds $I^{\xi+1}$ and $J^{\xi+1}$ will coincide also on the value of $X[\bar{a}]$. Consequently, in this step the truth value of $X[\bar{a}]$ is obtained.

Now, we will formalize the above concepts. Let Δ be a definition over vocabulary τ ($free(\Delta) \subseteq \tau$). The basis of the construction of the well-founded model is an operator \mathbb{T}_Δ mapping pairs of τ -structures to τ -structures. Given such a pair (I, J) , the operator \mathbb{T}_Δ operates like Γ_Δ , but evaluates the bodies of the rules in a different way. In particular, it evaluates positive occurrences of defined symbols in rule bodies by I , and negative occurrences of defined symbols by J .

To formally define this operator, we simply rename the negative occurrences in rule bodies of Δ . We extend the vocabulary τ with, for each defined symbol X , a new relation symbol X' of the same arity. The extended vocabulary $\tau \cup \bar{X}'$ will be denoted τ' . For every formula φ , we denote by φ' the formula obtained by substituting the symbol X for each negative occurrence of a defined symbol X . By applying this in each rule body in Δ , we obtain a new definition Δ' . For example, given the following definition

$$\Delta := \left\{ \begin{array}{l} \forall x \forall y (P(x) \leftarrow S(x, y) \wedge \neg P(y)), \\ \forall x \forall y (P(x) \leftarrow \neg Q(x, y) \wedge P(y)) \end{array} \right\},$$

we rename selected occurrences of P by P' , as described above, and obtain

$$\Delta' := \left\{ \begin{array}{l} \forall x \forall y (P(x) \leftarrow S(x, y) \wedge \neg P'(y)), \\ \forall x \forall y (P(x) \leftarrow \neg Q(x, y) \wedge P(y)) \end{array} \right\}.$$

The definition of Δ' defines the same predicates as Δ and its open symbols are those of Δ augmented with the new primed predicates \bar{X}' . Moreover, a defined symbol X has only positive occurrences and a primed symbol X' only negative occurrences in rule bodies of Δ' . Thus, Δ' is a positive definition over the vocabulary τ' . For each defined symbol X , the formula (9) constructed using Δ' instead of Δ is exactly φ'_X , the renaming of φ_X .

For any pair of τ -structures I, J which share the same domain, define I_J as the τ' -structure $J[\bar{X} : \bar{X}^I, \bar{X}' : \bar{X}^J]$. This I_J is a τ' -structure which satisfies the following:

- its domain is the same as the domain of I and J ,
- each open symbol of Δ is interpreted by J ,
- each defined symbol of Δ is interpreted by I ,
- the value of each new symbol X' is X^J , the value of X in J .

It is clear that for some defined symbol X , evaluating φ'_X under I_J simulates the non-standard evaluation of φ_X where J is “responsible” for the open and the

negative occurrences of the defined predicates, while I is “responsible” for the positive ones.

Let Δ be a definition over some vocabulary τ .

Definition 4.7. We introduce a partially defined binary operator $\mathbb{T}_\Delta : \mathcal{I} \times \mathcal{I} \mapsto \mathcal{I}$, where \mathcal{I} is the class of all τ -structures. The operator is defined on pairs of structures which share the same domain, and is undefined otherwise. We have $I' = \mathbb{T}_\Delta(I, J)$ iff

- $\text{dom}(I') = \text{dom}(J) = \text{dom}(I)$,
- for each open symbol σ , $\sigma^{I'} := \sigma^J$ and
- for each defined symbol $X \in \tau_\Delta^d$,

$$X^{I'} := \{\bar{a} \mid I_J \models \varphi'_X[\bar{a}]\}.$$

This definition is equivalent to defining $\mathbb{T}_\Delta(I, J) := \Gamma_{\Delta'}(I, J)|_\tau$, for any pair of τ -structures I, J such that $\text{dom}(I) = \text{dom}(J)$.

The following proposition shows a connection between operators \mathbb{T}_Δ and Γ_Δ .

PROPOSITION 4.8. *For any τ -structure I , it holds that $\mathbb{T}_\Delta(I, I) = \Gamma_\Delta(I)$.*

PROOF. Follows immediately from the fact that $I_I \models \varphi'_X[\bar{a}]$ iff $I \models \varphi_X[\bar{a}]$. \square

Remark 4.9. The relationship between \mathbb{T}_Δ and $\Gamma_\Delta(I)$ is even stronger in three- and four-valued logic. In particular, by interpreting Definition 4.5 in three-valued logic, the operator Γ_Δ has a natural extension Φ_Δ which operates on three-valued interpretations [Fitting 1985]. But three-valued interpretations \tilde{I} can be viewed as pairs (I, J) of 2-valued interpretations for which $I \sqsubseteq J$: for each domain atom $P[\bar{a}]$, $P[\bar{a}]^{\tilde{I}} = P[\bar{a}]^J = P[\bar{a}]^I$ if $P[\bar{a}]^I$ is **t** or **f**, and $P[\bar{a}]^{\tilde{I}} = \mathbf{f}$, $P[\bar{a}]^J = \mathbf{t}$ otherwise. This effectively implies that Φ_Δ can be viewed as an operator of pairs of interpretations. It turns out that $\Phi_\Delta(I, J) = (\mathbb{T}_\Delta(I, J), \mathbb{T}_\Delta(J, I))$. This property is the basis for an algebraic treatment of the concept of well-founded model in [Denecker et al. 2000].

PROPOSITION 4.10. *Let I_\circ be a fixed τ_Δ° -structure. In the lattice $\mathcal{S}_{I_\circ}^\tau$, the operator $\mathbb{T}_\Delta(I, J)$ is monotone in its first argument, and anti-monotone in its second argument.*

PROOF. Select arbitrary τ -structures $I, I', J, J' \in \mathcal{S}_{I_\circ}^\tau$ such that $I \sqsubseteq I'$ and $J' \sqsubseteq J$. We need to show that $\mathbb{T}_\Delta(I, J) \sqsubseteq \mathbb{T}_\Delta(I', J')$. Let $L = \mathbb{T}_\Delta(I, J)$ and $L' = \mathbb{T}_\Delta(I', J')$.

Let us verify that for each defined symbol X , $X^L \subseteq X^{L'}$. Let \bar{a} be any element of X^L . It holds that $I_J \models \varphi'_X[\bar{a}]$. The structure I'_J assigns the same value to open symbols in φ'_X , greater value to the defined symbols \bar{X} which occur positively in φ'_X , and lesser value to the defined symbols \bar{X}' which occur negatively in φ'_X . Consequently, it holds that $I'_J \models \varphi'_X[\bar{a}]$. We find that $\bar{a} \in X^{L'}$. We obtain our proposition. \square

COROLLARY 4.11. *Let I, M, J be three τ -extensions of a τ_Δ° -structure I_\circ such that $I \sqsubseteq M \sqsubseteq J$. Then it holds that $\mathbb{T}_\Delta(I, J) \sqsubseteq \Gamma_\Delta(M) \sqsubseteq \mathbb{T}_\Delta(J, I)$.*

PROOF. Since $I \sqsubseteq M \sqsubseteq J$, Proposition 4.10 entails that $\mathbb{T}_\Delta(I, J) \sqsubseteq \mathbb{T}_\Delta(M, M) = \Gamma_\Delta(M) \sqsubseteq \mathbb{T}_\Delta(J, I)$. \square

This corollary shows that T_Δ can be used to approximate Γ_Δ over an interval of structures. Indeed, if (I, J) is an approximation of M (i.e., $M \in [I, J]$) then the corollary shows that $(\mathsf{T}_\Delta(I, J), \mathsf{T}_\Delta(J, I))$ is an approximation of $\Gamma_\Delta(M)$. We shall elaborate on the approximation process in a moment.

Let J be a τ -structure, and J_o its restriction to τ_Δ^o . The unary operator $\lambda I \mathsf{T}_\Delta(I, J)$, often denoted by $\mathsf{T}_\Delta(\cdot, J)$, is a monotone operator in the lattice $\mathcal{S}_{J_o}^\tau$; and its least fixpoint in this lattice is computed by

$$\mathit{lfp}(\mathsf{T}_\Delta(\cdot, J)) := \bigsqcup_{\xi} E^\xi, \quad \text{where}$$

$$E^\xi := \mathsf{T}_\Delta(E^{<\xi}, J), \quad \text{and} \quad E^{<\xi} := \bigsqcup_{\eta < \xi} E^\eta.$$

Definition 4.12. Define the *stable operator* $ST_\Delta : \mathcal{I} \mapsto \mathcal{I}$ as follows:

$$ST_\Delta(J) := \mathit{lfp}(\mathsf{T}_\Delta(\cdot, J)) \quad (\text{computed in } \mathcal{S}_{J_o}^\tau).$$

The operator $\mathsf{T}_\Delta(I, J)$ performs one derivation step by interpreting positive occurrences of defined symbols by I and negative occurrences by J . The stable operator performs a monotone induction during which negative occurrences of defined predicates X in Δ are interpreted by the fixed value X^J .

PROPOSITION 4.13. *Let I_o be a fixed τ_Δ^o -structure. Operator ST_Δ is anti-monotone in $\mathcal{S}_{I_o}^\tau$.*

PROOF. Let $I \sqsubseteq J$ be τ -extensions of I_o . Let $J' \in \mathcal{S}_{I_o}^\tau$ be an arbitrary pre-fixpoint of $\mathsf{T}_\Delta(\cdot, I)$. Then, by anti-monotonicity of T_Δ in the second argument (Proposition 4.10), $\mathsf{T}_\Delta(J', J) \sqsubseteq \mathsf{T}_\Delta(J', I) \sqsubseteq J'$. Thus J' is a pre-fixpoint of $\mathsf{T}_\Delta(\cdot, J)$ and this entails that $ST_\Delta(J)$, the least (pre-)fixpoint of $\mathsf{T}_\Delta(\cdot, J)$ is less than $ST_\Delta(I)$, the least (pre-)fixpoint of $\mathsf{T}_\Delta(\cdot, I)$. \square

Fix some τ_Δ^o -structure I_o with domain A of the open symbols of Δ in τ .

As is standard for anti-monotone operators on a complete lattice (see Section 3.2), the operator ST_Δ gives rise to a sequence $(I^\xi, J^\xi)_{\xi \geq 0}$ in $\mathcal{S}_{I_o}^\tau$ defined by

$$\begin{aligned} I^\xi &:= ST_\Delta(J^{<\xi}), \quad \text{where} \quad J^{<\xi} := \sqcap_{\eta < \xi} J^\eta, \\ J^\xi &:= ST_\Delta(I^{<\xi}), \quad \text{where} \quad I^{<\xi} := \sqcup_{\eta < \xi} I^\eta. \end{aligned}$$

The anti-monotonicity of Γ_Δ implies that the sequence $(I^\xi)_{\xi \geq 0}$ is increasing and $(J^\xi)_{\xi \geq 0}$ is decreasing. Moreover, for each ξ , $I^\xi \sqsubseteq J^\xi$. Thus, it holds that the sequence $(I^\xi, J^\xi)_{\xi \geq 0}$ is indeed a sequence of increasingly precise approximations. This sequence has a limit (I, J) , which is the maximal oscillating pair of ST_Δ . Equivalently, I and J are fixpoints of the square ST_Δ^2 , $\mathit{lfp}(ST_\Delta^2)$ and $\mathit{gfp}(ST_\Delta^2)$, respectively.

In the lattice $\mathcal{S}_{I_o}^\tau$, we define

$$I_o^{\Delta\downarrow} := \mathit{lfp}(ST_\Delta^2), \quad \text{and} \quad I_o^{\Delta\uparrow} := \mathit{gfp}(ST_\Delta^2).$$

We extend this notation to any structure L which interprets at least τ_Δ^o and define

$$L^{\Delta\downarrow} := (L|_{\tau_\Delta^o})^{\Delta\downarrow}, \quad \text{and} \quad L^{\Delta\uparrow} := (L|_{\tau_\Delta^o})^{\Delta\uparrow}.$$

Note that $L^{\Delta\downarrow}$ and $L^{\Delta\uparrow}$ agree with L on the open symbols but not necessarily on the defined symbols.

Let I be any structure such that $\tau_{\Delta}^{\circ} \subseteq \tau_I \subseteq \tau$.

Definition 4.14 Δ -extension of I . If $I^{\Delta\downarrow} = I^{\Delta\uparrow}$, define the Δ -extension of I , denoted I^{Δ} , as $I^{\Delta} := I^{\Delta\downarrow}$. Otherwise, if $I^{\Delta\downarrow} \neq I^{\Delta\uparrow}$, then I has no Δ -extension.

Note that for any τ_{Δ}° -structure I_{\circ} , there is at most one Δ -extension of I_{\circ} .

PROPOSITION 4.15. *Let I_{\circ} be a fixed τ_{Δ}° -structure. If the Δ -extension of I_{\circ} exists, then it is a minimal fixpoint of Γ_{Δ} and the unique fixpoint of ST_{Δ} in $S_{I_{\circ}}^{\tau}$.*

Recall from Example 4.6 that Γ_{Δ} may have multiple minimal fixpoints.

PROOF. If $I = I_{\circ}^{\Delta}$ exists, then, since (I, I) is the maximal oscillating pair of ST_{Δ} , it is the unique fixpoint of this operator. I is also the least fixpoint of $\Upsilon_{\Delta}(\cdot, I)$. By Proposition 4.8, I is a fixpoint of Γ_{Δ} . Assume that $J \sqsubseteq I$ is also a fixpoint of Γ_{Δ} . By anti-monotonicity, $\Upsilon(J, I) \sqsubseteq \Upsilon(J, J) = J$, hence J is a pre-fixpoint of $\Upsilon_{\Delta}(\cdot, I)$. Since I is the least pre-fixpoint of this operator, $I \sqsubseteq J$. \square

Example 4.16. We illustrate the stable operator and the iterative process described above with the definition of Example 4.6. This definition has no open symbols and is equivalent to the following definition:

$$\left\{ \begin{array}{l} P \leftarrow \mathbf{t}, \\ Q \leftarrow \neg P \vee Q, \\ R \leftarrow \neg Q \end{array} \right\}.$$

For a propositional definition, the mapping $ST_{\Delta}(J)$ is the least fixpoint of the positive definition obtained by substituting each negative occurrence of a defined symbol and each occurrence of an open symbol by its truth value in J . For instance, the stable operator maps the empty structure \emptyset to the least fixpoint of the definition:

$$\left\{ \begin{array}{l} P \leftarrow \mathbf{t}, \\ Q \leftarrow \neg \mathbf{f} \vee Q, \\ R \leftarrow \neg \mathbf{f} \end{array} \right\}.$$

This yields the structure $\{P, Q, R\}$. Similarly, the stable operator maps the structure $\{P, Q, R\}$ to the least fixpoint of the definition:

$$\left\{ \begin{array}{l} P \leftarrow \mathbf{t}, \\ Q \leftarrow \neg \mathbf{t} \vee Q, \\ R \leftarrow \neg \mathbf{t} \end{array} \right\}.$$

This yields the structure $\{P\}$. Using this technique, we can construct the following table:

$$\begin{array}{llll} I^{<0} & := \emptyset, & J^{<0} & := \{P, Q, R\} \\ I^0 = I^{<1} & := \{P\}, & J^0 = J^{<1} & := \{P, Q, R\} & (P \text{ is derived}) \\ I^1 = I^{<2} & := \{P\}, & J^1 = J^{<2} & := \{P, R\} & (\neg Q \text{ is derived}) \\ I^2 & := \{P, R\}, & J^2 & := \{P, R\} & (R \text{ is derived}) \end{array}$$

The unique extension of this definition is $\{P, R\}$, which is exactly the intended model as explained in Example 4.6.

Example 4.17. Consider the definition:

$$\Delta_{\text{even}} := \left\{ \begin{array}{l} \forall x (E(x) \leftarrow x = 0), \\ \forall x (E(s(x)) \leftarrow \neg E(x)) \end{array} \right\},$$

which is equivalent to:

$$\left\{ \forall x [E(y) \leftarrow y = 0 \vee \exists x(y = s(x) \wedge \neg E(x))] \right\}.$$

We show that in the extension of Δ_{even} in the natural numbers, E is interpreted by the set of even numbers. Note that this definition has no positive occurrences of the defined predicate E . Therefore, $\Upsilon_{\Delta_{\text{even}}}(I, J) = \Gamma_{\Delta_{\text{even}}}(J)$, for all I, J sharing the same domain. The well-founded model computation starts in the least precise pair extending the natural numbers and proceeds as follows:

$$\begin{array}{ll} E^{I^{<0}} & := \emptyset, & E^{J^{<0}} & := \mathbb{N} \\ E^{I^0} = E^{I^{<1}} & := \{0\}, & E^{J^0} = E^{J^{<1}} & := \mathbb{N} \\ E^{I^1} = E^{I^{<2}} & := \{0\}, & E^{J^1} = E^{J^{<2}} & := \mathbb{N} \setminus \{1\} \\ E^{I^2} = E^{I^{<3}} & := \{0, 2\}, & E^{J^2} = E^{J^{<3}} & := \mathbb{N} \setminus \{1\} \\ E^{I^3} = E^{I^{<4}} & := \{0, 2\}, & E^{J^3} = E^{J^{<4}} & := \mathbb{N} \setminus \{1, 3\} \\ \dots & & & \end{array}$$

After iterating this process ω steps, we obtain the fixpoint:

$$E^{I^\omega} = E^{J^\omega} = \{2n \mid n \in \mathbb{N}\}.$$

Now we are ready to define the satisfaction relation between structures and well-formed formulas of the logic.

Definition 4.18 ϕ true in structure I . Let ϕ be a ID-formula and I any structure such that $\text{free}(\phi) \subseteq \tau_I$.

We define $I \models \phi$ (in words, ϕ is true in I , or I satisfies ϕ , or I is a model of ϕ) by the following induction:

- $I \models X(t_1, \dots, t_n)$ if $(t_1^I, \dots, t_n^I) \in X^I$;
- $I \models \psi_1 \wedge \psi_2$ if $I \models \psi_1$ and $I \models \psi_2$;
- $I \models \neg\psi$ if $I \not\models \psi$;
- $I \models \exists\sigma \psi$ if for some value v of σ in the domain $\text{dom}(I)$ of I , $I[\sigma : v] \models \psi$;
- $I \models \Delta$ if $I = I^{\Delta\downarrow} = I^{\Delta\uparrow}$.

Given an ID-theory T over τ , a τ -structure I satisfies T (is a model of T) if I satisfies each $\phi \in T$. This is denoted by $I \models T$.

Observe that a definition in ID-logic is a truth conditional construct (i.e., it has a truth value in structure). On the other hand, the definitional implication \leftarrow is not a truth functional connective (i.e., its truth is not described as a boolean function of the truth values of head and body), in fact definitional rules are not even assigned a truth value in structure.

Example 4.19. An ID-theory can contain multiple definitions for the same predicate. A simple illustration is when a natural class is partitioned in subclasses in

different ways, depending on the property used. For example, humans can be partitioned in males and females, but also in adults and children, etc.. This is modelled by the following formula:

$$\left\{ \begin{array}{l} \forall x (Human(x) \leftarrow Male(x)), \\ \forall x (Human(x) \leftarrow Female(x)) \end{array} \right\} \wedge \left\{ \begin{array}{l} \forall x (Human(x) \leftarrow Adult(x)), \\ \forall x (Human(x) \leftarrow Child(x)) \end{array} \right\}.$$

This formula implies that the class humans is the union of the classes males and females, and also of the classes adults and children. The definition

$$\left\{ \begin{array}{l} \forall x (Human(x) \leftarrow Male(x) \vee Female(x)), \\ \forall x (Human(x) \leftarrow Adult(x) \vee Child(x)) \end{array} \right\}.$$

is weaker, in the sense that it does not entail that humans are either males or females.

Example 4.20. Consider the theory $T_{\mathbb{N}}$ of Example 4.3. We prove that each model I of $T_{\mathbb{N}}$ is isomorphic to the structure of the natural numbers. Let I be a model of this theory. First, since I satisfies the two first-order Peano axioms, the domain elements $0^I, s(0)^I, \dots, s^n(0)^I, \dots$ are pair-wise distinct and the set of these domain elements constitutes a subset of $dom(I)$, isomorphic to the natural numbers. Therefore, it suffices to show that this set is exactly the domain of I . Since I satisfies the ID-axiom replacing the induction axiom, there exists a set $S \subseteq dom(I)$ such that $I[N : S]$ satisfies

$$\left\{ \begin{array}{l} \forall x (N(x) \leftarrow x = 0), \\ \forall x (N(s(x)) \leftarrow N(x)) \end{array} \right\} \wedge \forall x N(x).$$

Since $I[N : S]$ satisfies $\forall x N(x)$, S must be $dom(I)$. As proven later in Theorem 6.3, $I[N : S]$ satisfies the positive definition in this axiom iff S is the least set containing 0^I and closed under s^I . Hence, $dom(I)$ is exactly the set $\{0^I, s(0)^I, \dots, s^n(0)^I, \dots\}$.

The aim of a definition is to *define* its defined symbols, i.e., that the definition does not produce undefined atoms. This is a natural quality requirement for a definition. This principle is formalized in the semantics by the condition that $I_o^{\Delta\downarrow} = I_o^{\Delta\uparrow}$.

Definition 4.21 total definition. Let I be a structure such that $\tau_{\Delta}^o \subseteq \tau_I$. Definition Δ is *total* in I if $I^{\Delta\downarrow} = I^{\Delta\uparrow}$. If $\tau_K \subseteq \tau_{\Delta}^o$, we say that Δ is *total in K* if Δ is total in each τ_{Δ}^o -structure extending K . The definition Δ is *total in a theory T* if Δ is total in each model of T . A definition Δ is *total* if it is total in each τ_{Δ}^o -structure I_o .

The next example shows that a (useful) definition which is total in one structure, may not be total in other structures.

Example 4.22. Consider the definition of Example 4.17:

$$\Delta_{even} := \left\{ \begin{array}{l} \forall x (E(x) \leftarrow x = 0), \\ \forall x (E(s(x)) \leftarrow \neg E(x)) \end{array} \right\}.$$

Recall that the stable operator of this definition is identical to $\Gamma_{\Delta_{even}}$. In Example 4.17, we showed that this definition is total in the structure of the natural numbers.

In Example 4.20, we saw that the natural numbers are the unique model of $T_{\mathbb{N}}$ (modulo isomorphism). Consequently, Δ_{even} is total in $T_{\mathbb{N}}$.

The definition is not total in many other structures, in particular in those where the successor function contains cycles or infinite descending chains. For example, Δ_{even} is not total in the structure I_o with domain $\{0, 1\}$, and $s^{I_o}(0) = 1$, $s^{I_o}(1) = 0$. In this structure, the maximal oscillating pair (I, J) of $\Gamma_{\Delta_{even}}$ interprets E as follows:

$$\begin{aligned} E^I &:= \{0\}, \\ E^J &:= \{0, 1\}. \end{aligned}$$

Definition Δ_{even} is not total either in the structure I_o' with domain \mathbb{Z} and $s^{I_o'}$ the standard successor function on \mathbb{Z} . In this structure, the maximal oscillating pair (I, J) of $\Gamma_{\Delta_{even}}$ interprets E as follows:

$$\begin{aligned} E^I &:= \{2n \mid n \in \mathbb{N}\}, \\ E^J &:= \{n \mid n < 0\} \cup \{2n \mid n \in \mathbb{N}\}. \end{aligned}$$

What is the cause of the non-totality of a definition? In the above examples, the dependency order, induced by the rules, contains infinite descending chains in which atoms depend negatively on the same or other atoms. When this happens, the stable operator oscillates between a structure in which all atoms of the chain are false and one in which these atoms are true.

The next proposition expresses that total definitions in ID-logic satisfy the definitional extension principle (see Section 2).

PROPOSITION 4.23. *Let T be a theory over τ° and Δ a definition such that $\tau_\Delta^\circ = \tau^\circ$. If Δ is total in T , then there is one-to-one correspondence between τ° -models N of T and τ -models M of $T \cup \{\Delta\}$ such that $M = N^\Delta$. Consequently, $T \cup \{\Delta\}$ is satisfiable (i.e., has a model) iff T is satisfiable, and for each τ° -formula φ , $T \cup \{\Delta\} \models \varphi$ iff $T \models \varphi$.*

The proof is straightforward. The property does not hold in case of non-total definitions. For example, adding the definition $\{P \leftarrow \neg P\}$ to a theory causes inconsistency.

Totality is a fundamental property in our theory of non-monotone induction. Unfortunately, the problem of determining, for a given definition Δ and τ_Δ° -structure I_o , whether Δ is total in I_o , is undecidable [Schlipf 1995a]. Below, we will prove totality results for important subclasses of definitions.

4.3 Historical note

The well-founded model construction was first presented in [Van Gelder et al. 1991] but the link with iterated induction and induction over well-founded orders was laid only later in [Denecker 1998; Denecker et al. 2001]. Nevertheless, the view of logic programs as representations of *definitions* has been implicit in many studies on the semantics of logic programming, for example in the completion semantics [Clark 1978], in the *standard* semantics of stratified logic programs [Apt et al. 1988], in the original work of well-founded semantics [Van Gelder et al. 1991], and in [Schlipf 1995b]. In [Van Gelder 1993], the well-founded semantics was extended to rule sets with arbitrary FO-bodies. Van Gelder presents this logic in the spirit of fixpoint

logics, as a logic of alternating fixpoints and calls it *AFP*, for *Alternating Fixpoint Partial model*. This logic can be viewed as an ancestor of ID-logic.

The view of logic programs as definitions was never standard due to the existence of an alternative view originally proposed by Gelfond and Lifschitz. They proposed to interpret *negation as failure* literals `not P` in logic programs as epistemic literals *I do not know P* or as default negation *it can be assumed that* $\neg P$ [Gelfond and Lifschitz 1988]. This view was further developed in the stable semantics which is the basis of Answer Set Programming [Niemelä 1999; Marek and Truszczyński 1999]. The differences between the ASP view and the definition view are remarkable. While in the default view, negation as failure is an epistemic operator and the rule operator of LP corresponds to material implication, in the definitional view, it is the rule operator that has a non-classical interpretation while negation has the classical logic meaning. Here we present three arguments for this thesis. First, negation in non-monotone inductive rules in mathematics has the standard meaning of negation, e.g. in the rule

$$I \models \neg\varphi \text{ if } I \not\models \varphi, \text{ i.e., if } I \text{ does not satisfy } \varphi.$$

Second, if negation in definitions would have an epistemic modal meaning, this would be displayed in the formal semantics, e.g. by the use of Kripke-structures, or possible world collections, or by the use of first-order belief sets (sets of believed formulas, as in default logic, or sets of believed literals, as in ASP). This is not the case. Third, all transformations of ID-logic definitions to classical logic presented in Section 6 translate negation in rules to classical negation.

The stable and well-founded semantics are mathematically closely related and coincide in many cases. It is remarkable then that they were proposed as formalizations of quite different informal semantics. This phenomenon is explained in [Denecker 2004].

5. ANALYSIS OF INDUCTIVE DEFINITIONS

In this section, we study techniques to analyse inductive definitions in ID-logic: a technique to investigate the internal structure of a definition, a modularity result to split up a definition in an equivalent conjunction of sub-definitions and a technique to prove totality of a large definition from its sub-definitions.

5.1 Reduction Relations

Several times, we informally discussed the concept of dependency relation between defined atoms induced by a definition. In this section, we formalize this concept through the notion of *reduction relation* and investigate its basic properties. In the following sections, this concept will prove to be a useful tool to analyze definitions.

Assume a definition Δ over τ and a structure K_o with domain A such that $\tau_{K_o} \subseteq \tau_\Delta^o$.

Recall that At_A^τ denotes the set of domain atoms over vocabulary τ in domain A . Let \prec be a binary relation on At_A^τ . Given a domain atom $P[\bar{a}]$, we define $I \cong_{\prec P[\bar{a}]} J$ if $\tau_I = \tau_J$, $f^I = f^J$ for every constant and function symbol f appearing in Δ , and $Q[\bar{b}]^I = Q[\bar{b}]^J$ for all $Q[\bar{b}] \prec P[\bar{a}]$. This relation is extended to tuples by defining $(I, J) \cong_{\prec P[\bar{a}]} (I', J')$ if $I \cong_{\prec P[\bar{a}]} I'$ and $J \cong_{\prec P[\bar{a}]} J'$.

Definition 5.1 reduction relation. A binary relation \prec on At_A^τ is a *reduction relation* (or briefly, a reduction) of a rule $\forall \bar{x}(P(\bar{t}[\bar{x}]) \leftarrow \varphi(\bar{x})) \in \Delta$ if for all τ -structures I, J, I', J' extending K_o , for all tuples \bar{a} and \bar{d} such that $\bar{a} = \bar{t}^{J[\bar{x}:\bar{d}]}$, if $(I, J) \cong_{\prec P[\bar{a}]} (I', J')$ then it holds that $I_J \models \varphi'[\bar{d}]$ iff $I'_{J'} \models \varphi'[\bar{d}]$.

The relation \prec is a reduction relation of Δ in K_o if for each defined predicate P of Δ , \prec is a reduction relation of the rule $\forall \bar{x}(P(\bar{x}) \leftarrow \varphi_P)$ in K_o , where φ_P is as defined in (9).

Let \prec be a reduction of a rule (or definition) in K_o .

PROPOSITION 5.2. *If K_o' extends K_o (i.e., $K_o'|_{\tau_{K_o}} = K_o$), then \prec is a reduction of the rule (or definition) in K_o' .*

PROPOSITION 5.3. *Any superset \prec' of \prec is a reduction relation of the rule (or definition) in K_o .*

PROPOSITION 5.4. *Let \prec be a reduction relation of each rule in Δ in K_o . Then \prec is a reduction relation of Δ in K_o .*

The converse is not true. For instance, $\{(R, P)\}$ is a reduction relation of the definition $\{P \leftarrow R; P \leftarrow R \wedge Q\}$ but not of its second rule.

PROOF. Assume that $(I, J) \cong_{\prec P[\bar{a}]} (I', J')$ and let $P[\bar{a}]$ be an arbitrary defined domain atom. If $I_J \models \varphi'_P[\bar{a}]$, then there exists a rule $\forall \bar{x}(P(\bar{t}) \leftarrow \varphi(\bar{x})) \in \Delta$ and domain elements \bar{d} such that $\bar{a} = \bar{t}^{J[\bar{x}:\bar{d}]}$ and $I_J \models \varphi'[\bar{d}]$. Because \prec is a reduction relation of the rule, $I'_{J'} \models \varphi'[\bar{d}]$. Because $J \cong_{\prec P[\bar{a}]} J'$, we have $\bar{t}^{J[\bar{x}:\bar{d}]} = \bar{t}^{J'[\bar{x}:\bar{d}]}$. Consequently, $I'_{J'} \models \varphi'_P[\bar{a}]$. The opposite direction follows by symmetry. \square

The proposition below shows that the relevant tuples in a reduction relation are those representing dependencies of defined atoms on domain atoms that are not interpreted by K_o .

PROPOSITION 5.5. *If \prec is a reduction of definition Δ (or rule in Δ) in K_o , then so is $\prec_{K_o}^\Delta := \{(P[\bar{a}], Q[\bar{b}]) \in \prec \mid P \notin \tau_{K_o} \text{ and } Q \in \tau_{K_o}^d\}$.*

The above propositions suggest a methodology for constructing reductions of a definition: define reductions for each of its rules and take the union.

Example 5.6. Consider the definition

$$\Delta := \left\{ \begin{array}{l} \forall x (E(x) \leftarrow x = 0), \\ \forall x (E(s(x)) \leftarrow O(x)), \\ \forall x (O(s(x)) \leftarrow E(x)) \end{array} \right\}.$$

Below is a list of reduction relations, one for each rule, in the structure of natural numbers:

$$\begin{array}{l} \emptyset, \\ \{(O[n], E[n+1]) \mid n \in \mathbb{N}\}, \\ \{(E[n], O[n+1]) \mid n \in \mathbb{N}\}. \end{array}$$

The union of these relations is a reduction relation of Δ .

A reduction relation \prec specifies for each domain atom $P[\bar{a}]$ on which atoms it *directly* depends. The transitive closure \prec^* of \prec is also a reduction relation

and specifies all atoms on which $P[\bar{a}]$ depends directly or indirectly. A transitive reduction relation generalizes the well-known concept of dependency graph of a logic program to general structures and FO-formulas in the body of rules. Whereas the dependency graph of a logic program is unique, a definition Δ may have many reduction relations in K_o . In particular, the total binary relation $\prec_t = At_A^r \times At_A^r$ is always a reduction relation. A reduction relation in general overestimates the dependencies between domain atoms in a definition. Only the least reduction relation of a definition reflects the true dependencies. However, as shown in the next example, some definitions do not have a least reduction relation.

Example 5.7. Consider the following definition in the context of the natural numbers:

$$\Delta := \{ P \leftarrow \exists n \forall m (m > n \supset Q(m)) \}.$$

The predicate Q is open in this definition. This definition defines P to be true if there exists a number n such that Q contains at least all natural numbers larger than n . It can easily be verified that for each $n \in \mathbb{N}$, the relation

$$\prec_n = \{(Q(m), P) \mid m > n\}$$

is a reduction relation of Δ in \mathbb{N} . The intersection of these relations is \emptyset , and this is not a reduction relation of Δ .

Let \prec be a transitive reduction relation of Δ in K_o . The main theorem of this section is that all models of Δ extending K_o agree on a defined domain atom $P[\bar{a}]$ if they agree on all function symbols of Δ and on all open atoms $Q[\bar{b}] \prec P[\bar{a}]$.

THEOREM 5.8. *For each defined domain atom $P[\bar{a}]$, for all models I, J of Δ extending K_o , if $I|_{\tau_\Delta} \cong_{\prec P[\bar{a}]} J|_{\tau_\Delta}$ then $P[\bar{a}]^I = P[\bar{a}]^J$.*

The proof is based on the following basic proposition.

PROPOSITION 5.9. *Let \prec be a reduction relation of Δ in K_o , $P[\bar{a}]$ a domain atom and let I, I', J, J' be τ -structures extending K_o such that $(I, J) \cong_{\prec P[\bar{a}]} (I', J')$.*

- (a) *If P is defined then $P[\bar{a}]^{\top_{\Delta}(I, J)} = P[\bar{a}]^{\top_{\Delta}(I', J')}$.*
- (b) *If \prec is transitive, then $\top_{\Delta}(I, J) \cong_{\prec P[\bar{a}]} \top_{\Delta}(I', J')$.*

PROOF. (a) Since P is a defined predicate of Δ , $P[\bar{a}]^{\top_{\Delta}(I, J)}$ is the truth value of $\varphi'_P[\bar{a}]$ in I, J and likewise $P[\bar{a}]^{\top_{\Delta}(I', J')}$ is the truth value of $\varphi'_P[\bar{a}]$ in I', J' . Since \prec is a reduction relation, the truth value of this formula is the same in I, J as in I', J' .

(b) Assume that \prec is transitive. Let $Q[\bar{b}]$ be an arbitrary domain atom such that $Q[\bar{b}] \prec P[\bar{a}]$. If Q is an open predicate of Δ then $Q[\bar{b}]^{\top_{\Delta}(I, J)} = Q[\bar{b}]^J = Q[\bar{b}]^{J'} = Q[\bar{b}]^{\top_{\Delta}(I', J')}$. Let Q be a defined predicate of Δ . By transitivity of \prec , the set of atoms on which $Q[\bar{b}]$ depends is a subset of the set of atoms on which $P[\bar{a}]$ depends. This, and the fact that $(I, J) \cong_{\prec P[\bar{a}]} (I', J')$ implies that $(I, J) \cong_{\prec Q[\bar{b}]} (I', J')$. By application of (a) we obtain that $Q[\bar{b}]^{\top_{\Delta}(I, J)} = Q[\bar{b}]^{\top_{\Delta}(I', J')}$. \square

If \prec is a transitive reduction relation, item (b) of this proposition states that \top_{Δ} preserves $\cong_{\prec P[\bar{a}]}$, for each domain atom $P[\bar{a}]$. This is a key property. The reduction relation \prec defines a collection of lattice congruences $\cong_{\prec P[\bar{a}]}$ in $\mathcal{S}_{K_o}^r$, one for each

domain atom $P[\bar{a}]$ (confer Section 3.2.2). The operator \top_Δ is the basic operator in the well-founded model construction. The fact that it preserves the congruence $\cong_{\prec P[\bar{a}]}$ “propagates” to the stable operator ST_Δ and then to the construction of the well-founded model. This observation is summarized in the following proposition.

Let \prec be a transitive reduction of Δ in K_o and assume that $\tau_{\text{fn}} \subseteq \tau_{K_o} \subseteq \tau_\Delta^o$ so that $\langle \mathcal{S}_{K_o}^\tau, \sqsubseteq \rangle$ is a complete lattice.

PROPOSITION 5.10. *For every domain atom $P[\bar{a}]$, the relation $\cong_{\prec P[\bar{a}]}$ is a lattice congruence of $\langle \mathcal{S}_{K_o}^\tau, \sqsubseteq \rangle$. If \prec is a transitive reduction of Δ in K_o , then \top_Δ and ST_Δ preserve $\cong_{\prec P[\bar{a}]}$ in the lattice $\langle \mathcal{S}_{K_o}^\tau, \sqsubseteq \rangle$. For all τ_Δ^o -structures I_o, J_o , if $I_o \cong_{\prec P[\bar{a}]} J_o$ then $I_o^{\Delta\downarrow} \cong_{\prec P[\bar{a}]} J_o^{\Delta\downarrow}$ and $I_o^{\Delta\uparrow} \cong_{\prec P[\bar{a}]} J_o^{\Delta\uparrow}$.*

The proof of this proposition is by straightforward inductions first on the iteration of $\top_\Delta(\cdot, I)$ and then of ST_Δ . Theorem 5.8 is a corollary to this proposition.

5.2 Modularity

In this section, we split a definition Δ into sub-definitions $\{\Delta_1, \Delta_2, \dots, \Delta_n\}$. We study under what conditions we can guarantee that for structure I ,

$$I \models \Delta \quad \text{iff} \quad I \models \Delta_1 \wedge \Delta_2 \wedge \dots \wedge \Delta_n.$$

This is the subject of the Modularity theorem. The theorem tells us when we can understand a large definition as a conjunction of component definitions. Frequently, these component definitions have a simpler form — they may be positive definitions or non-recursive definition. Therefore, the ability to decompose definitions without side effects is useful for analyzing large definitions — some properties of large definitions are implied by properties of sub-definitions. Thus, the Modularity theorem is an important tool for simplifying logical formulas with definitions.

Everywhere in this section, we fix a definition Δ over some vocabulary τ .

Definition 5.11 partition of definitions. A partition of definition Δ is a set $\{\Delta_1, \dots, \Delta_n\}$, $1 < n$, such that $\Delta = \Delta_1 \cup \dots \cup \Delta_n$, and if defined symbol P appears in the head of a rule of Δ_i , $1 \leq i \leq n$, then all rules of Δ with P in the head belong to Δ_i .

If $\{\Delta_1, \dots, \Delta_n\}$ is a partition of Δ , then $\cup_i \tau_{\Delta_i}^d = \tau_\Delta^d$, and $\tau_{\Delta_i}^d \cap \tau_{\Delta_j}^d = \emptyset$ whenever $i \neq j$. Notice that each Δ_i has some “new” open symbols. For instance, if P is defined in Δ , but not in Δ_i , then it is a new open symbol of Δ_i . Of course, it holds that $\tau = \tau_\Delta^o \cup \tau_\Delta^d = \tau_{\Delta_i}^o \cup \tau_{\Delta_i}^d$, $1 \leq i \leq n$.

The following theorem demonstrates that a model of a definition, is, at the same time, a model of each of its sub-definitions.

THEOREM 5.12 DECOMPOSITION. *Let Δ be a definition over τ with partition $(\Delta_1, \dots, \Delta_n)$. Let I be a τ -structure. If $I \models \Delta$ then $I \models \Delta_1 \wedge \dots \wedge \Delta_n$.*

The following example shows that the inverse direction of Theorem 5.12 does not hold in general.

Example 5.13. Let $\Delta, \Delta_1, \Delta_2$ be the following definitions:

$$\Delta := \left\{ \begin{array}{l} P \leftarrow Q, \\ Q \leftarrow P \end{array} \right\}, \Delta_1 := \{ P \leftarrow Q \}, \Delta_2 := \{ Q \leftarrow P \}.$$

Definition Δ is total, and its unique model is \emptyset in which both P and Q are false. According to Theorem 5.12, \emptyset satisfies Δ_1 and Δ_2 . Note that $\{P, Q\}$ is not a model of Δ and yet, it satisfies Δ_1 and Δ_2 . Indeed, $\{P, Q\}$ is the Δ_1 -extension of the $\tau_{\Delta_1}^o$ -structure $\{Q\}$ and the Δ_2 -extension of the $\tau_{\Delta_2}^o$ -structure $\{P\}$.

In the proof of Theorem 5.12, we will use the following basic lemmas comparing fixpoints of monotone and anti-monotone operators.

PROPOSITION 5.14. *Let Γ_1, Γ_2 be two monotone operators in a lattice with least fixpoints $lfp(\Gamma_1) = o_1, lfp(\Gamma_2) = o_2$ respectively.*

- (a) *if $x \leq o_1$ implies $\Gamma_1(x) \leq \Gamma_2(x)$ then $o_1 \leq o_2$;*
- (b) *if $o_2 \leq x$ implies $\Gamma_1(x) \leq \Gamma_2(x)$ then $o_1 \leq o_2$.*

PROPOSITION 5.15. *Let Γ_1, Γ_2 be anti-monotone operators on lattice L with maximal oscillating pair (o_1, u_1) , respectively (o_2, u_2) .*

- (a) *If $x \leq o_1$ implies $\Gamma_2(x) \leq \Gamma_1(x)$ and $u_1 \leq y$ implies $\Gamma_1(y) \leq \Gamma_2(y)$ then $o_1 \leq o_2 \leq u_2 \leq u_1$. If in addition $o_1 = u_1$, then $o_1 = o_2 = u_2 = u_1$.*
- (b) *If $x \leq o_1$ implies $u_1 \leq \Gamma_2(x)$ and $u_1 \leq y$ implies $\Gamma_2(y) \leq o_1$, then $o_2 \leq o_1 \leq u_1 \leq u_2$.*

PROOF. (of Theorem 5.12) Let $I \models \Delta$ and denote $I_o := I|_{\tau_{\Delta}^o}$ and $I_{oi} := I|_{\tau_{\Delta_i}^o}$. We prove that $I \models \Delta_i$.

Observe that $(I, I) = \text{OSC}(ST_{\Delta})$ in $\mathcal{S}_{I_o}^r$ and $(I_{oi}^{\Delta_i \downarrow}, I_{oi}^{\Delta_i \uparrow}) = \text{OSC}(ST_{\Delta_i})$ in $\mathcal{S}_{I_{oi}}^r$. We wish to apply Proposition 5.15(a) to compare the oscillation pairs of ST_{Δ} and ST_{Δ_i} . To do that, we have to extend ST_{Δ_i} to the greater lattice $\mathcal{S}_{I_o}^r$.

Let $I_{oi}[\tau_{\Delta_i}^d : K]$ denote the τ -structure interpreting symbols of $\tau_{\Delta_i}^o$ as in I_{oi} and symbols of $\tau_{\Delta_i}^d$ as in K . Define the operator Γ on $\mathcal{S}_{I_o}^r$ as the mapping from $K \in \mathcal{S}_{I_o}^r$ to $ST_{\Delta_i}(I_{oi}[\tau_{\Delta_i}^d : K])$. Clearly, Γ maps $\mathcal{S}_{I_o}^r$ into its sub-lattice $\mathcal{S}_{I_{oi}}^r$, since $\Gamma(K) = ST_{\Delta_i}(I_{oi}[\tau_{\Delta_i}^d : K])$ extends I_{oi} . Also, if $K \in \mathcal{S}_{I_{oi}}^r$ then $K = I_{oi}[\tau_{\Delta_i}^d : K]$, hence Γ and ST_{Δ_i} coincide in $\mathcal{S}_{I_{oi}}^r$. It is also easy to see that Γ is an anti-monotone operator. It follows from this that $\text{OSC}(\Gamma) = \text{OSC}(ST_{\Delta_i})$.

We will now show that for each $K \in \mathcal{S}_{I_o}^r$,

$$K \sqsubseteq I \text{ implies } \Gamma(K) \sqsubseteq ST_{\Delta}(K), \text{ and} \quad (10)$$

$$I \sqsubseteq K \text{ implies } ST_{\Delta}(K) \sqsubseteq \Gamma(K). \quad (11)$$

Application of Proposition 5.15(a) will then yield the desired result

$$I = I_{oi}^{\Delta_i \downarrow} = I_{oi}^{\Delta_i \uparrow}.$$

Fix an arbitrary $K \in \mathcal{S}_{I_o}^r$. It holds that

$$ST_{\Delta}(K) = lfp(\mathbb{T}_{\Delta}(\cdot, K))$$

where the first argument of \mathbb{T}_{Δ} ranges over $\mathcal{S}_{I_o}^r$. Likewise, for every $K \in \mathcal{S}_{I_o}^r$,

$$\Gamma(K) = lfp(\mathbb{T}_{\Delta_i}(\cdot, I_{oi}[\tau_{\Delta_i}^d : K]))$$

but here the first argument of \mathbb{T}_{Δ_i} ranges over $\mathcal{S}_{I_{oi}}^r$. To compare $ST_{\Delta}(K)$ and $\Gamma(K)$, we want to apply the fixpoint comparison Proposition 5.14 for the monotone operators $\mathbb{T}_{\Delta}(\cdot, K)$ and $\mathbb{T}_{\Delta_i}(\cdot, I_{oi}[\tau_{\Delta_i}^d : K])$. A problem is that the fixpoint computations are done in different lattices. Observe however that $\mathbb{T}_{\Delta_i}(\cdot, I_{oi}[\tau_{\Delta_i}^d : K])$ is

well-defined and monotone in the greater lattice $\mathcal{S}_{I_o}^T$ and that it maps any $J \in \mathcal{S}_{I_o}^T$ into $\mathcal{S}_{I_{oi}}^T$. It follows that $\Gamma(K) = \text{lfp}(\mathbb{T}_{\Delta_i}(\cdot, I_{oi}[\tau_{\Delta_i}^d : K]))$ computed in the lattice $\mathcal{S}_{I_o}^T$.

For arbitrary $J \in \mathcal{S}_{I_o}^T$, let us denote $J_1 := \mathbb{T}_{\Delta}(J, K)$ and $J_2 := \mathbb{T}_{\Delta_i}(J, I_{oi}[\tau_{\Delta_i}^d : K])$. For every domain atom $Q[\bar{b}]$ such that $Q \in \tau_{\Delta_i}^o \cap \tau_{\Delta}^d$, it holds that:

$$Q[\bar{b}]^{J_1} = \varphi'_Q[\bar{b}]^{J_K},$$

$$Q[\bar{b}]^{J_2} = Q[\bar{b}]^{I_{oi}} = Q[\bar{b}]^I = \varphi'_Q[\bar{b}]^{I_I}$$

where the last equality follows from $I = \mathbb{T}_{\Delta}(I, I)$.

For every domain atom $P[\bar{a}]$ such that $P \in \tau_{\Delta_i}^d$, it holds that:

$$P[\bar{a}]^{J_1} = \varphi'_P[\bar{a}]^{J_K},$$

$$P[\bar{a}]^{J_2} = \varphi'_P[\bar{a}]^{J_{I_{oi}[\tau_{\Delta_i}^d : K]}}.$$

In the evaluation of $\varphi'_Q[\bar{b}]^{J_K}$ and $\varphi'_P[\bar{a}]^{J_K}$, positive occurrences of predicates are interpreted by J while negative occurrences are interpreted by K . When $K \sqsubseteq I$ and $I \sqsubseteq J$ then also $K \sqsubseteq I_{oi}[\tau_{\Delta_i}^d : K]$, and by using well-known monotonicity laws of truth assignment, it is easy to see that $Q[\bar{b}]^{J_2} \leq Q[\bar{b}]^{J_1}$ and $P[\bar{a}]^{J_2} \leq P[\bar{a}]^{J_1}$, i.e.,

$$\mathbb{T}_{\Delta_i}(J, I_{oi}[\tau_{\Delta_i}^d : K]) \sqsubseteq \mathbb{T}_{\Delta}(J, K). \quad (12)$$

Likewise, if $I \sqsubseteq K$ and $J \sqsubseteq I$ then also $I_{oi}[\tau_{\Delta_i}^d : K] \sqsubseteq K$ and $Q[\bar{b}]^{J_1} \leq Q[\bar{b}]^{J_2}$ and $P[\bar{a}]^{J_1} \leq P[\bar{a}]^{J_2}$, i.e.,

$$\mathbb{T}_{\Delta}(J, K) \sqsubseteq \mathbb{T}_{\Delta_i}(J, I_{oi}[\tau_{\Delta_i}^d : K]). \quad (13)$$

To prove (10), assume that $K \sqsubseteq I$. For any $J \in \mathcal{S}_{I_o}^T$ such that $ST_{\Delta}(K) \sqsubseteq J$, it holds that $I = ST_{\Delta}(I) \sqsubseteq ST_{\Delta}(K) \sqsubseteq J$ (by anti-monotonicity of ST_{Δ}), and hence, (12) holds. By application of Proposition 5.14(b), we obtain the desired result that $\Gamma(K) \sqsubseteq ST_{\Delta}(K)$.

Similarly, to prove (11), assume that $I \sqsubseteq K$. For any $J \in \mathcal{S}_{I_o}^T$ such that $J \sqsubseteq ST_{\Delta}(K)$, it holds that $J \sqsubseteq ST_{\Delta}(K) \sqsubseteq ST_{\Delta}(I) = I$, hence (13) holds. Now the conditions of Proposition 5.14(a) are satisfied and we obtain that $ST_{\Delta}(K) \sqsubseteq \Gamma(K)$. This concludes the proof. \square

Theorem 5.12 gives one direction of the Modularity theorem. Now our goal is to come up with some condition on the partition of Δ so that both directions of the Modularity theorem hold.

Recall from Example 5.13 that the other direction does not hold in general. In particular, $\{P, Q\}$ satisfies $\{P \leftarrow Q\}$ and $\{Q \leftarrow P\}$ but not $\{P \leftarrow Q, Q \leftarrow P\}$. In this example, splitting the definition breaks the circular dependency between P and Q . This causes the broken equivalence between Δ and $\Delta_1 \wedge \Delta_2$. The example suggests that splitting a definition will be equivalence preserving if the splitting does not break circular dependencies between atoms. Below we will formalise this notion using the notion of *reduction relation* defined in Section 5.1.

Recall that a pre-well-order is a reflexive and transitive relation such that every non-empty subset contains a minimal element. Let K_o be a structure such that $\tau_{K_o} \subseteq \tau_{\Delta}^o$.

Definition 5.16 reduction partition. Call partition $\{\Delta_1, \dots, \Delta_n\}$ of definition Δ a *reduction partition* of Δ in K_o if there is a reduction pre-well-order \prec of Δ in K_o such that for all defined domain atoms $P[\bar{a}], Q[\bar{b}]$, if $Q[\bar{b}] \prec P[\bar{a}]$ and $P[\bar{a}] \prec Q[\bar{b}]$, then P and Q are defined in the same Δ_i .

The intuition underlying this definition is that in a reduction partition, if an atom defined in one module depends on an atom defined in another module, then the latter atom is strictly less in the reduction ordering and hence does not depend on the first atom.

Recall from Section 3.2.2 that each lattice congruence \cong induces a lattice-pre-order \leq_{\cong} . In case of $\cong_{\prec P[\bar{a}]}$, it is easy to verify that $I \sqsubseteq_{\prec P[\bar{a}]} J$ iff for all domain atoms $Q[\bar{b}] \prec P[\bar{a}]$, $Q[\bar{b}]^I \leq Q[\bar{b}]^J$.

LEMMA 5.17. *Let \prec be a transitive reduction relation of a rule $\forall \bar{x}(Q(\bar{x}) \leftarrow \varphi[\bar{x}])$ in K_o . Let $P[\bar{a}]$ be a domain atom and I, I', J, J' be τ -extensions of K_o such that $I \sqsubseteq_{\prec P[\bar{a}]} I'$ and $J' \sqsubseteq_{\prec P[\bar{a}]} J$. If $Q[\bar{b}] \prec P[\bar{a}]$ then $\varphi'[\bar{b}]^{I,J} \leq \varphi'[\bar{b}]^{I',J'}$.*

PROOF. Let I, I', J, J' and $Q[\bar{b}]$ be as in the lemma. Construct structure I_1 from I and I' such that for each domain atom $R(\bar{c})$, $R(\bar{c})^{I_1} := R(\bar{c})^{I'}$ if $R(\bar{c}) \prec P[\bar{a}]$, and $R(\bar{c})^{I_1} := R(\bar{c})^I$ otherwise. Similarly construct structure J_1 such that for each atom $R(\bar{c})$, $R(\bar{c})^{J_1} := R(\bar{c})^{J'}$ if $R(\bar{c}) \prec P[\bar{a}]$, and $R(\bar{c})^{J_1} := R(\bar{c})^J$ otherwise. By construction, $I \sqsubseteq I_1 \cong_{\prec P[\bar{a}]} I'$ and $J' \cong_{\prec P[\bar{a}]} J_1 \sqsubseteq J$. By transitivity of \prec , $I_1 \cong_{Q[\bar{b}]} I'$ and $J' \cong_{Q[\bar{b}]} J_1$. We have

$$\varphi'[\bar{b}]^{I,J} \leq \varphi'[\bar{b}]^{I_1,J_1} = \varphi'[\bar{b}]^{I',J'}.$$

□

PROPOSITION 5.18. *Let \prec be a transitive reflexive reduction relation of Δ in a τ_{Δ}^o -structure I_o and let Δ_i be an element of some partition of Δ . Denote $I_{oi} := I_o^{\Delta_i \downarrow} |_{\tau_{\Delta_i}^o}$ and $I_{oi}' := I_o^{\Delta_i \uparrow} |_{\tau_{\Delta_i}^o}$.*

If $I_{oi} \cong_{\prec P[\bar{a}]} I_{oi}'$ then $I_{oi}^{\Delta_i \downarrow} \sqsubseteq_{\prec P[\bar{a}]} I_o^{\Delta_i \downarrow} \sqsubseteq I_o^{\Delta_i \uparrow} \sqsubseteq_{\prec P[\bar{a}]} I_{oi}^{\Delta_i \uparrow}$.

PROOF. The proof follows the line of argument of the proof of Theorem 5.12. For convenience, let us denote $I := I_o^{\Delta_i \downarrow}$ and $I' := I_o^{\Delta_i \uparrow}$. Assume that $I_{oi} \cong_{\prec P[\bar{a}]} I_{oi}'$. Recall that for any structure $J \in \mathcal{S}_{I_o}^{\tau}$, $|J|_{\prec P[\bar{a}]}$ denotes the equivalence class of J in the lattice congruence $\cong_{\prec P[\bar{a}]}$. The statement to be proven is equivalent to

$$|I_{oi}^{\Delta_i \downarrow}|_{\prec P[\bar{a}]} \sqsubseteq |I_o^{\Delta_i \downarrow}|_{\prec P[\bar{a}]} \sqsubseteq |I_o^{\Delta_i \uparrow}|_{\prec P[\bar{a}]} \sqsubseteq |I_{oi}^{\Delta_i \uparrow}|_{\prec P[\bar{a}]}.$$
 (14)

It follows from Proposition 5.10 and Proposition 3.4 that

$$(|I|_{\prec P[\bar{a}]}, |I'|_{\prec P[\bar{a}]}) = \text{OSC}(|ST_{\Delta}|_{\prec P[\bar{a}]})$$

computed in the lattice $|\mathcal{S}_{I_o}^{\tau}|_{\prec P[\bar{a}]}$, and

$$(|I_{oi}^{\Delta_i \downarrow}|_{\prec P[\bar{a}]}, |I_{oi}^{\Delta_i \uparrow}|_{\prec P[\bar{a}]}) = \text{OSC}(|ST_{\Delta_i}|_{\prec P[\bar{a}]})$$

computed in the lattice $|\mathcal{S}_{I_{oi}}^{\tau}|_{\prec P[\bar{a}]}$. Thus, we need to compare the maximal oscillating pairs of the two quotient operators.

The first step is to extend the operator $|ST_{\Delta_i}|_{\prec P[\bar{a}]}$ to the larger lattice $|\mathcal{S}_{I_o}^{\tau}|_{\prec P[\bar{a}]}$. Consider again the operator Γ defined in the proof of Theorem 5.12. This operator

maps any $K \in \mathcal{S}_{I_o}^{\tau}$ to $ST_{\Delta_i}(I_{oi}[\tau_{\Delta_i}^d : K])$. When $K \cong_{\prec P[\bar{a}]} K'$, then clearly $I_{oi}[\tau_{\Delta_i}^d : K] \cong_{\prec P[\bar{a}]} I_{oi}[\tau_{\Delta_i}^d : K']$. Also, \prec is a reduction relation of Δ_i in I_{oi} . It follows that Γ preserves $\cong_{\prec P[\bar{a}]}$. Therefore, its quotient $|\Gamma|_{\prec P[\bar{a}]}$ is a well-defined anti-monotone operator such that

$$\text{OSC}(|ST_{\Delta_i}|_{\prec P[\bar{a}]}) = \text{OSC}(|\Gamma|_{\prec P[\bar{a}]}) .$$

We will show that for each $\tilde{K} \in |\mathcal{S}_{I_o}^{\tau}|_{\prec P[\bar{a}]}$,

$$\tilde{K} \sqsubseteq |I|_{\prec P[\bar{a}]} \text{ implies } |I'|_{\prec P[\bar{a}]} \sqsubseteq |\Gamma|_{\prec P[\bar{a}]}(\tilde{K}), \text{ and} \quad (15)$$

$$|I'|_{\prec P[\bar{a}]} \sqsubseteq \tilde{K} \text{ implies } |\Gamma|_{\prec P[\bar{a}]}(\tilde{K}) \sqsubseteq |I|_{\prec P[\bar{a}]} . \quad (16)$$

Application of Proposition 5.15(b) will then yield the desired result.

We first prove (15). Select an arbitrary $\tilde{K} \in |\mathcal{S}_{I_o}^{\tau}|_{\prec P[\bar{a}]}$ and a witness K of \tilde{K} . The structures I' and $\Gamma(K)$ are the least fixpoints of the following monotone operators in $\mathcal{S}_{I_o}^{\tau}$:

$$I' = \text{lfp}(\mathbb{T}_{\Delta}(\cdot, I)), \text{ and}$$

$$\Gamma(K) = \text{lfp}(\mathbb{T}_{\Delta_i}(\cdot, I_{oi}[\tau_{\Delta_i}^d : K])) .$$

Let \tilde{J} be an arbitrary element of $|\mathcal{S}_{I_o}^{\tau}|_{\prec P[\bar{a}]}$, and let J be a witness of \tilde{J} . Denote $J_1 := \mathbb{T}_{\Delta}(J, I)$ and $J_2 := \mathbb{T}_{\Delta_i}(J, I_{oi}[\tau_{\Delta_i}^d : K])$.

For every domain atom $Q[\bar{b}] \prec P[\bar{a}]$ such that $Q \in \tau_{\Delta_i}^o \cap \tau_{\Delta}^d$, it holds that:

$$Q[\bar{b}]^{J_1} = \varphi'_Q[\bar{b}]^{J_I},$$

$$Q[\bar{b}]^{J_2} = Q[\bar{b}]^{I_{oi}} = Q[\bar{b}]^{I_{oi}'} = Q[\bar{b}]^{I'} = \varphi'_Q[\bar{b}]^{I'}$$

where the second equality holds because $I_{oi} \cong_{\prec P[\bar{a}]} I_{oi}'$ and $Q[\bar{b}] \prec P[\bar{a}]$, respectively $I' = \mathbb{T}_{\Delta}(I', I)$. For every domain atom $R[\bar{c}] \prec P[\bar{a}]$ such that $R \in \tau_{\Delta_i}^d$, it holds that:

$$R[\bar{c}]^{J_1} = \varphi'_R[\bar{c}]^{J_I},$$

$$R[\bar{c}]^{J_2} = \varphi'_R[\bar{c}]^{J_{I_{oi}[\tau_{\Delta_i}^d : K]}} .$$

Assume that $\tilde{K} \sqsubseteq |I|_{\prec P[\bar{a}]}$ or equivalently, $K \sqsubseteq_{\prec P[\bar{a}]} I$. We shall prove that $\tilde{J} \sqsubseteq |I'|_{\prec P[\bar{a}]}$ implies $J_1 \sqsubseteq_{\prec P[\bar{a}]} J_2$, or equivalently,

$$|\mathbb{T}_{\Delta}|_{\prec P[\bar{a}]}(\tilde{J}, |I|_{\prec P[\bar{a}]}) \sqsubseteq |\mathbb{T}_{\Delta_i}|_{\prec P[\bar{a}]}(\tilde{J}, |I_{oi}[\tau_{\Delta_i}^d : K]|_{\prec P[\bar{a}]}) .$$

By Proposition 5.14(a), it will follow then that $|I'|_{\prec P[\bar{a}]} \sqsubseteq |\Gamma|_{\prec P[\bar{a}]}(\tilde{K})$.

Assume that $\tilde{J} \sqsubseteq |I'|_{\prec P[\bar{a}]}$, i.e., $J \sqsubseteq_{\prec P[\bar{a}]} I'$. From Lemma 5.17, it follows that $Q[\bar{b}]^{J_1} = \varphi'_Q[\bar{b}]^{J_I} \leq \varphi'_Q[\bar{b}]^{I'} = Q[\bar{b}]^{J_2}$. Since $K \sqsubseteq_{\prec P[\bar{a}]} I$, it holds that $I_{oi}[\tau_{\Delta_i}^d : K] \sqsubseteq_{\prec P[\bar{a}]} I$. Again using Lemma 5.17, we find that $R[\bar{c}]^{J_1} \leq R[\bar{c}]^{J_2}$. Hence, $J_1 \sqsubseteq_{\prec P[\bar{a}]} J_2$. This concludes the proof of (15).

The proof of (16) differs from the one of (15) in subtle ways. Now assume that $|I'|_{\prec P[\bar{a}]} \sqsubseteq \tilde{K}$, or $I' \sqsubseteq_{\prec P[\bar{a}]} K$. This time, we use:

$$I = \text{lfp}(\mathbb{T}_{\Delta}(\cdot, I')),$$

$$\Gamma(K) = \text{Ifp}(\mathsf{T}_{\Delta_i}(\cdot, I_{\text{oi}}[\tau_{\Delta_i}^{\text{d}} : K])).$$

For arbitrary $\tilde{J} \in |\mathcal{S}_{I_o}^{\tau}|_{\prec P[\bar{a}]}$ and witness J of \tilde{J} , denote $J_1 := \mathsf{T}_{\Delta}(J, I')$ and $J_2 := \mathsf{T}_{\Delta_i}(J, I_{\text{oi}}[\tau_{\Delta_i}^{\text{d}} : K])$. We shall prove that $I \sqsubseteq_{\prec P[\bar{a}]} J$ implies $J_2 \sqsubseteq_{\prec P[\bar{a}]} J_1$. Under these conditions, application of Proposition 5.14(b) yields the desired conclusion that $|\Gamma|_{\prec P[\bar{a}]}(\tilde{K}) \sqsubseteq |I|_{\prec P[\bar{a}]}$.

For every domain atom $Q[\bar{b}] \prec P[\bar{a}]$ such that $Q \in \tau_{\Delta_i}^{\circ} \cap \tau_{\Delta_i}^{\text{d}}$, it holds that:

$$Q[\bar{b}]^{J_1} = \varphi'_Q[\bar{b}]^{J_{I'}} \text{ and } Q[\bar{b}]^{J_2} = Q[\bar{b}]^I = \varphi'_Q[\bar{b}]^{I_{I'}},$$

where the last equation follows from $I = \mathsf{T}_{\Delta}(I, I')$.

For every domain atom $R[\bar{c}] \prec P[\bar{a}]$ such that $R \in \tau_{\Delta_i}^{\text{d}}$, it holds that:

$$R[\bar{c}]^{J_1} = \varphi'_R[\bar{c}]^{J_{I'}} \text{ and } R[\bar{c}]^{J_2} = \varphi'_R[\bar{c}]^{J_{I_{\text{oi}}[\tau_{\Delta_i}^{\text{d}} : K]}}.$$

When $I \sqsubseteq_{\prec P[\bar{a}]} J$, Lemma 5.17 implies $Q[\bar{b}]^{J_2} \leq Q[\bar{b}]^{J_1}$. Since $I' \sqsubseteq_{\prec P[\bar{a}]} K$ and $I_{\text{oi}} \cong_{\prec P[\bar{a}]} I_{\text{oi}}'$, it holds that $I' \sqsubseteq_{\prec P[\bar{a}]} I_{\text{oi}}'[\tau_{\Delta_i}^{\text{d}} : K] \cong_{\prec P[\bar{a}]} I_{\text{oi}}[\tau_{\Delta_i}^{\text{d}} : K]$. Again using Lemma 5.17, we find that $R[\bar{c}]^{J_2} \leq R[\bar{c}]^{J_1}$. It follows that $J_2 \sqsubseteq_{\prec P[\bar{a}]} J_1$. \square

Now, we are in a position to prove the second direction of the modularity theorem. Let $\tau^{\circ} \subseteq \tau_{\Delta}^{\circ}$.

THEOREM 5.19. *If Δ has a reduction partition $\{\Delta_1, \dots, \Delta_n\}$ in the τ° -structure K_o , then for any τ -structure I extending K_o , if $I \models \Delta_1 \wedge \dots \wedge \Delta_n$ then $I \models \Delta$.*

PROOF. Let I be a model of $\Delta_1 \wedge \dots \wedge \Delta_n$ extending K_o and let $I_o = I|_{\tau_{\Delta}^{\circ}}$. Assume, towards contradiction, that $I_o^{\Delta \downarrow} \neq I$ or $I_o^{\Delta \uparrow} \neq I$. Let $P[\bar{a}]$ be a minimal domain atom in the reduction pre-well order of the partition such that $P[\bar{a}]^{I_o^{\Delta \downarrow}} \neq P[\bar{a}]^I$ or $P[\bar{a}]^{I_o^{\Delta \uparrow}} \neq P[\bar{a}]^I$ and suppose P is defined in Δ_i . By our choice of $P[\bar{a}]$, $I|_{\tau_{\Delta_i}^{\circ}} \cong_{\prec P[\bar{a}]} I_o^{\Delta \downarrow}|_{\tau_{\Delta_i}^{\circ}} \cong_{\prec P[\bar{a}]} I_o^{\Delta \uparrow}|_{\tau_{\Delta_i}^{\circ}}$. Let us denote $I_o^{\Delta \downarrow}|_{\tau_{\Delta_i}^{\circ}}$ by I_{oi} . Proposition 5.10 guarantees that $I = (I|_{\tau_{\Delta_i}^{\circ}})^{\Delta_i} \cong_{\prec P[\bar{a}]} I_{\text{oi}}^{\Delta_i \downarrow} \cong_{\prec P[\bar{a}]} I_{\text{oi}}^{\Delta_i \uparrow}$. Since \prec is reflexive, $P[\bar{a}]^I = P[\bar{a}]^{I_{\text{oi}}^{\Delta_i \downarrow}} = P[\bar{a}]^{I_{\text{oi}}^{\Delta_i \uparrow}}$. But Proposition 5.18 implies that $I_{\text{oi}}^{\Delta_i \downarrow} \sqsubseteq_{\prec P[\bar{a}]} I_o^{\Delta \downarrow} \sqsubseteq I_o^{\Delta \uparrow} \sqsubseteq_{\prec P[\bar{a}]} I_{\text{oi}}^{\Delta_i \uparrow}$. We obtain the desired contradiction. \square

THEOREM 5.20 MODULARITY. *If $\{\Delta_1, \dots, \Delta_n\}$ is a reduction partition of Δ in τ° -structure K_o , then for any τ -structure I extending K_o ,*

$$I \models \Delta \text{ iff } I \models \Delta_1 \wedge \dots \wedge \Delta_n.$$

PROOF. Combine Theorems 5.12 and 5.19. \square

An immediate consequence is the following corollary.

COROLLARY 5.21. *Let T_o be a theory over τ° such that for any τ° -model I_o of T_o , $\{\Delta_1, \dots, \Delta_n\}$ is a reduction partition of Δ in I_o .*

Then $T_o \wedge \Delta$ and $T_o \wedge \Delta_1 \wedge \dots \wedge \Delta_n$ are logically equivalent.

Example 5.22. Consider the partition of the definition from Example 5.6:

$$\Delta_1 := \left\{ \begin{array}{l} \forall x (E(x) \leftarrow x = 0), \\ \forall x (E(s(x)) \leftarrow O(x)) \end{array} \right\}, \quad \Delta_2 := \left\{ \forall x (O(s(x)) \leftarrow E(x)) \right\}.$$

The transitive reflexive closure \prec^{**} of the reduction of Δ presented in Example 5.6 is a well-founded partial order. It holds that $E[n] \prec^{**} O[m]$ and $O[n] \prec^{**} E[m]$ iff $n < m$. Consequently, $\{\Delta_1, \Delta_2\}$ is a reduction partition. The conjunction of Δ_1 and Δ_2 has one model extending the natural numbers. In ID-logic, the natural numbers are formalized by $T_{\mathbb{N}}$ (Examples 4.3 and 4.20). By Corollary 5.21, the theories $T_{\mathbb{N}} \cup \{\Delta_1 \cup \Delta_2\}$ and $T_{\mathbb{N}} \cup \{\Delta_1 \wedge \Delta_2\}$ are equivalent.

Since a definition has at most one extension in an τ_{Δ}° -structure I_o , another corollary of Theorem 5.20 is the following.

COROLLARY 5.23. *If Δ has a reduction partition $\{\Delta_1, \dots, \Delta_n\}$ in a τ_{Δ}° -structure I_o then $\Delta_1 \wedge \dots \wedge \Delta_n$ has at most one model extending I_o .*

The modularity problem has been studied quite extensively in the context of different semantics and for several extensions of logic programming, e.g. in [Lifschitz and Turner 1994; Schlipf 1995b; Verbaeten et al. 2000], but our results are uniformly stronger in the sense that they are proven for a more expressive logic and under more general conditions. An algebraic version of our modularity theorem was proven in [Vennekens et al. 2005] in the context of approximation theory.

5.3 Proving totality of a definition

As a final topic of this section, we also study under what conditions the totality of the sub-definitions $\Delta_1, \dots, \Delta_n$ guarantees the totality of Δ . This is investigated in the totality theorem. Let $\tau^{\circ} \subseteq \tau_{\Delta}^{\circ}$.

THEOREM 5.24 TOTALITY. *If $\{\Delta_1, \dots, \Delta_n\}$ is a reduction partition of Δ in τ° -structure K_o and, for every $i \in \{1, \dots, n\}$, Δ_i is total in K_o then Δ is total in K_o .*

Thus, one way to prove that Δ is total in K_o is to prove that it has a reduction partition, and that each definition Δ_i in the partition is total in K_o .

PROOF. Take an arbitrary τ_{Δ}° -structure I_o extending K_o . Assume, towards contradiction, that $I_o^{\Delta \downarrow} \neq I_o^{\Delta \uparrow}$ and let $P[\bar{a}]$ be a minimal domain atom in the reduction pre-well order of the partition such that $P[\bar{a}]^{I_o^{\Delta \downarrow}} < P[\bar{a}]^{I_o^{\Delta \uparrow}}$. Let P be defined in Δ_i . By our choice of $P[\bar{a}]$, $I_o^{\Delta \downarrow}|_{\tau_{\Delta_i}^{\circ}} \cong_{\prec P[\bar{a}]} I_o^{\Delta \uparrow}|_{\tau_{\Delta_i}^{\circ}}$. Let us denote $I_o^{\Delta \downarrow}|_{\tau_{\Delta_i}^{\circ}}$ by I_{oi} . By totality of Δ_i , $I_{oi}^{\Delta_i \downarrow} = I_{oi}^{\Delta_i \uparrow}$. It follows from Proposition 5.18 that $I_{oi}^{\Delta_i \downarrow} \sqsubseteq_{\prec P[\bar{a}]} I_o^{\Delta \downarrow} \sqsubseteq I_o^{\Delta \uparrow} \sqsubseteq_{\prec P[\bar{a}]} I_{oi}^{\Delta_i \uparrow}$. By reflexivity of \prec , it follows that $P[\bar{a}]$ has the same value in all these structures. This is a contradiction. \square

COROLLARY 5.25. *Let T_o be a theory over τ° such that for any τ° -model I_o of T_o , $\{\Delta_1, \dots, \Delta_n\}$ is a reduction partition of Δ in I_o and each Δ_i is total in I_o . Then Δ is total in T_o .*

COROLLARY 5.26 SATISFIABILITY. *If Δ has a reduction partition $\{\Delta_1, \dots, \Delta_n\}$ in a τ_{Δ}° -structure I_o such that $\Delta_1, \dots, \Delta_n$ are total in I_o , then Δ and $\Delta_1 \wedge \dots \wedge \Delta_n$ are satisfiable and have exactly one model that extends I_o .*

6. SOME FAMILIAR TYPES OF DEFINITIONS

This section reconsiders the four different types of informal inductive definitions discussed in Section 2: non-recursive definitions, positive definitions, definitions

over well-founded sets and iterated inductive definitions. We demonstrate that these types of definitions can be correctly and uniformly represented in ID-logic. To this end, we define four formal subclasses of definitions of ID-logic that naturally correspond to the four informal types of inductive definitions and prove theorems to show that the well-founded semantics correctly formalizes the meaning of these types of definitions.

6.1 Non-Recursive Definitions.

A first case is that of non-recursive definitions. A definition Δ is non-recursive if the bodies of the rules do not contain defined predicates.

Definition 6.1 completion of Δ [Clark 1978]. Define the *completion* of Δ , denoted $\text{comp}(\Delta)$, as the conjunction, for each defined symbol X of Δ , of formulas

$$\forall \bar{x}(X(\bar{x}) \leftrightarrow \varphi_X[\bar{x}]).$$

The equivalence $\forall \bar{x}(X(\bar{x}) \leftrightarrow \varphi_X[\bar{x}])$ is called the *completed definition* of X .

THEOREM 6.2. *Let Δ be a non-recursive definition over τ . Then Δ is total and a τ -structure I satisfies Δ iff I satisfies $\text{comp}(\Delta)$.*

PROOF. It is straightforward to show that if Δ is non-recursive, then for each τ_Δ° -structure I_o , the operator T_Δ is constant in the lattice $\mathcal{S}_{I_o}^\tau$ and it maps each pair of τ -structures to the unique structure I such that, for each defined symbol X ,

$$X^I = \{\bar{d} \mid I_o \models \varphi_X[\bar{d}]\}.$$

This I is the unique model of Δ and the unique model of $\text{comp}(\Delta)$ in $\mathcal{S}_{I_o}^\tau$. \square

6.2 Positive Definitions.

Let Δ be a positive definition, defining the symbols \bar{P} . Let \bar{X} be a set of new predicate symbols such that for each defined symbol P_i , X_i and P_i have the same arity. Define the following formula

$$PID(\Delta) := \bigwedge \Delta \wedge \forall \bar{X} (\bigwedge \Delta[\bar{P}/\bar{X}] \supset (\bar{P} \subseteq \bar{X})).$$

Here, $\bigwedge \Delta$ is the conjunction of formulas obtained by replacing definitional rules with material implications in Δ ; $\Delta[\bar{P}/\bar{X}]$ is the definition obtained by substituting X_i for each defined symbol P_i and $\bar{P} \subseteq \bar{X}$ is a shorthand for the formula $\forall \bar{x}(P_1(\bar{x}) \supset X_1(\bar{x})) \wedge \dots \wedge \forall \bar{x}(P_n(\bar{x}) \supset X_n(\bar{x}))$. The formula $PID(\Delta)$ is the standard second-order formula to express that predicates \bar{P} satisfy the positive inductive definition Δ .

Define also

$$\text{Circ}(\Delta; \bar{P}) := \bigwedge \Delta \wedge \forall \bar{X} (\bigwedge \Delta[\bar{P}/\bar{X}] \wedge \bar{X} \subseteq \bar{P} \supset \bar{P} \subseteq \bar{X}).$$

This formula is the standard circumscription of $\bigwedge \Delta$ with respect to the defined predicates \bar{P} [Lifschitz 1994].

THEOREM 6.3. *Let Δ be a positive definition over τ . Then Δ is total and for all τ -structures I , the following are equivalent:*

- (a) I is a model of Δ ;
- (b) I is the least fixpoint of Γ_Δ in the lattice $\mathcal{S}_{I_o}^T$;
- (c) I is a model of $PID(\Delta)$;
- (d) I is a model of $Circ(\Delta; \bar{P})$.

PROOF. In case Δ is a positive definition, defined symbols have no negative occurrences, so Δ and Δ' are identical. Consequently, for any pair of structures I, J in the lattice $\mathcal{S}_{I_o}^T$, it holds that $\top_\Delta(I, J) = \Gamma_\Delta(I)$ which does not depend on J . Thus, the stable operator ST_Δ is a constant operator in this lattice and maps any structure J to the least fixpoint of Γ_Δ . Thus, it follows that $I_o^{\Delta\downarrow}$ and $I_o^{\Delta\uparrow}$ are identical to the least fixpoint of Γ_Δ in $\mathcal{S}_{I_o}^T$. This proves the equivalence of (a) and (b).

The equivalence of (b) and (c) in case of a positive definition is well-known (see e.g. [Aczel 1977]). Finally, the axiom $PID(\Delta)$ expresses that \bar{P} should be the least relations satisfying $\bigwedge \Delta$, while $Circ(\Delta; \bar{P})$ expresses that \bar{P} should be minimal relations satisfying $\bigwedge \Delta$. Both axioms are equivalent, since there is a set of least relations satisfying $\bigwedge \Delta$, and it is the unique set of minimal relations satisfying this formula. \square

The theorem is significant since it shows that for positive definitions, the semantics defined here coincides with standard monotone induction. It implies that if $I \models \Delta$ then I is the least structure extending I_o that satisfies the rules of Δ viewed as a set of first-order implications.

Example 6.4. Consider the formulation of the induction axiom in ID-logic in Example 4.3:

$$\exists N \left[\left\{ \begin{array}{l} \forall x (N(x) \leftarrow x = 0), \\ \forall x (N(s(x)) \leftarrow N(x)) \end{array} \right\} \wedge \forall x N(x) \right].$$

By Theorem 6.3, it is equivalent to the second-order axiom

$$\exists N \left[\begin{array}{l} \forall x (N(x) \subset x = 0) \wedge \\ \forall x (N(s(x)) \subset N(x)) \wedge \\ \forall X [\forall x (X(x) \subset x = 0) \wedge \forall x (X(s(x)) \subset X(x)) \supset \forall x (N(x) \supset X(x))] \wedge \\ \forall x N(x) \end{array} \right].$$

We show that this formula is logically equivalent to the standard induction axiom. The first two conjuncts follow from the last and may be deleted. Using the last conjunct, the third conjunct can be simplified as follows:

$$\exists N \left[\begin{array}{l} \forall X [\forall x (X(x) \subset x = 0) \wedge \forall x (X(s(x)) \subset X(x)) \supset \forall x X(x)] \wedge \\ \forall x N(x) \end{array} \right].$$

Notice that the first element of the conjunction does not depend of N , so the outer existential quantifier can be moved inwards, and the tautological $\exists N \forall x N(x)$ can be removed. We obtain the standard induction axiom:

$$\forall X [\forall x (X(x) \subset x = 0) \wedge \forall x (X(s(x)) \subset X(x)) \supset \forall x X(x)].$$

6.3 Iterated Inductive Definitions

Recall from Section 2 that an iterated inductive definition constructs a set as the limit of a sequence of subsidiary computations, each of which itself is a monotone

induction. Here, we formalize that intuition, and make a connection between this new “formalism” and the representation of iterated inductive definitions in ID-logic.

Let $(\Delta_1, \dots, \Delta_n)$ be a finite sequence of positive definitions over a vocabulary τ such that:

- all definitions define disjoint sets of relation symbols, i.e., $\tau_{\Delta_i}^d \cap \tau_{\Delta_j}^d = \emptyset$ for $i \neq j$;
- if a relation symbol is defined in some Δ_i , then it does not occur as an open symbol in Δ_j , for any $j < i$.

We call such a sequence an *iterated inductive definition* and we interpret it as a simple, finite case of an iterated inductive definition.

Let \bar{X} be the set $\tau_{\Delta_1}^d \cup \dots \cup \tau_{\Delta_n}^d$, i.e., the collection of all symbols defined in at least one definition Δ_i , $1 \leq i \leq n$, and let τ° be the vocabulary $\tau \setminus \bar{X}$. Select an arbitrary τ° -structure I_o .

We define $I_o^{(\Delta_1, \dots, \Delta_n)}$ by induction on i : $I_o^{(0)} := I_o$ and for each i , $1 \leq i \leq n$, $I_o^{(\Delta_1, \dots, \Delta_i)} := (I_o^{(\Delta_1, \dots, \Delta_{i-1})})^{\Delta_i}$. Note that by Theorem 6.3, $I_o^{(\Delta_1, \dots, \Delta_i)}$ is the least fixpoint of the positive definition Δ_i extending $I_o^{(\Delta_1, \dots, \Delta_{i-1})}$. The above definition models precisely the process of iterated induction as explained in Section 2. We say that the τ -structure $I_o^{(\Delta_1, \dots, \Delta_n)}$ is the *structure defined by the iterated inductive definition* $(\Delta_1, \dots, \Delta_n)$ in I_o .

Consider the iterated inductive definition $(\Delta_1, \dots, \Delta_n)$ and the new definition $\Delta = \Delta_1 \cup \dots \cup \Delta_n$. It is obvious that τ_Δ° is equal to τ° .

THEOREM 6.5 ITERATED INDUCTION. *Let $(\Delta_1, \dots, \Delta_n)$ be an iterated inductive definition over vocabulary τ . Definition $\Delta := \Delta_1 \cup \dots \cup \Delta_n$ is a total definition, and for all τ -structures I extending a τ° -structure I_o , the following are equivalent:*

- (a) I is a model of Δ ;
- (b) I is the structure defined by $(\Delta_1, \dots, \Delta_n)$ in I_o , i.e., $I = I_o^{(\Delta_1, \dots, \Delta_n)}$;
- (c) I satisfies $PID(\Delta_1) \wedge \dots \wedge PID(\Delta_n)$.

The above notion of *iterated inductive definition* extends that of a *stratified logic program* and its perfect model [Apt et al. 1988]. Theorem 6.5 generalizes theorem 6.1 in [Van Gelder et al. 1991] to rules with FO bodies and arbitrary structures. The theorem’s significance and the reason to include it in this paper is that it shows that definitions in ID-logic correctly formalize iterated induction, a point that was not addressed in [Apt et al. 1988] and [Van Gelder et al. 1991].

An iterated induction can be viewed as a repeated application of the principle of definitional extension. The proof of Theorem 6.5 indeed relies on the following formalization of this principle.

PROPOSITION 6.6. *Let Δ be a definition over τ° , Δ_1 a definition over τ such that $\tau_{\Delta_1}^\circ = \tau^\circ$, i.e., Δ_1 defines new symbols in terms of τ° . Let K_o be a structure such that $\tau_{\text{fn}} \subseteq \tau_{K_o} \subseteq \tau^\circ$.*

If Δ_1 is total in K_o then there is a one-to-one mapping between τ -models M of $\Delta \cup \Delta_1$ extending K_o and τ° -models N of Δ extending K_o such that $M = N^{\Delta_1}$.

Consequently, extending a definition Δ with a total set of rules defining new symbols, has no effect on the entailed properties expressible in the vocabulary of Δ .

PROOF. Consider the binary relation \prec on At_A^τ such that $P[\bar{a}] \prec Q[\bar{b}]$ if $P \in \tau^\circ$ and $Q \in \tau$, or, if $P, Q \in \tau_{\Delta_1}^d$. Since the rules of Δ_1 do not mention symbols of τ_Δ^d ,

this is a reduction relation of $\Delta \cup \Delta_1$ in K_o . Clearly, $\{\Delta, \Delta_1\}$ is a reduction partition of $\Delta \cup \Delta_1$ in K_o . Hence, for any τ -structure M extending K_o , $M \models \Delta \cup \Delta_1$ iff $M \models \Delta$ and $M \models \Delta_1$ iff $M|_{\tau^o} \models \Delta$ and $M = (M|_{\tau^o})^{\Delta_1 \downarrow} = (M|_{\tau^o})^{\Delta_1 \uparrow}$. By totalness of Δ_1 in K_o , the proposition follows. \square

PROOF. (of Theorem 6.5.) The proof is by induction on n . Assume that the theorem holds for iterated inductive definitions of length $n - 1$. By Theorem 6.3, Δ_n is total in any structure. By Proposition 6.6, for every τ -structure I , $I \models \Delta_1 \cup \dots \cup \Delta_n$ iff $I|_{\tau_{\Delta_n}^o} \models \Delta_1 \cup \dots \cup \Delta_{n-1}$ and $I = (I|_{\tau_{\Delta_n}^o})^{\Delta_n}$. Again by Theorem 6.3, the latter condition is equivalent to $I \models PID(\Delta_n)$. By the induction hypothesis, the first condition is equivalent to $I|_{\tau_{\Delta_n}^o} = I_o^{(\Delta_1, \dots, \Delta_{n-1})}$ and with $I|_{\tau_{\Delta_n}^o} \models PID(\Delta_1) \wedge \dots \wedge PID(\Delta_{n-1})$. The theorem follows directly. \square

6.4 Definitions over Well-Founded Order.

We now present a formalization of the informal concept of a definition over a well-founded order (see section 2) in the framework of ID-logic. Let Δ be a definition over τ and K_o a structure such that $\tau_{K_o} \subseteq \tau_{\Delta}^o$.

Definition 6.7 strict reduction relation. A reduction relation \prec of Δ in K_o is *strict* if it is a strict well-founded partial order (i.e., an anti-symmetric, transitive binary relation without infinite descending chains).

Hence, a strict reduction \prec has no cycles. If Δ allows a strict reduction then there are no atoms that depend on themselves.

Definition 6.8 definition over a well-founded order. We say that Δ is a definition over the (strict) well-founded order \prec in K_o if \prec is a strict reduction relation of Δ in K_o .

THEOREM 6.9 COMPLETION. *Suppose \prec is a strict reduction relation of Δ in K_o . The definition Δ is total in K_o and for any τ -structure I extending K_o , $I \models \Delta$ iff $I \models comp(\Delta)$.*

PROOF. Fix an arbitrary τ_{Δ}^o -structure I_o extending K_o . We will show that the equality $I_o^{\Delta \downarrow} = I_o^{\Delta \uparrow} = I_o^{\Delta}$ holds, and moreover that for any τ -structure I extending I_o , $I \models comp(\Delta)$ iff $I = I_o^{\Delta}$. Since I_o is arbitrary, we will obtain the proof of the theorem.

We start by showing that there is at most one pair (I, J) in $\mathcal{S}_{I_o}^{\tau}$ satisfying $\Upsilon_{\Delta}(I, J) = I$ and $\Upsilon_{\Delta}(J, I) = J$, moreover if such a pair exists then $I = J$. Suppose that there are two such pairs; i.e., there exist $I, J, I', J' \in \mathcal{S}_{I_o}^{\tau}$ such that $\Upsilon_{\Delta}(I, J) = I$, $\Upsilon_{\Delta}(J, I) = J$, $\Upsilon_{\Delta}(I', J') = I'$ and $\Upsilon_{\Delta}(J', I') = J'$. Let $P[\bar{a}]$ be a minimal atom in \prec such that $P[\bar{a}]^I \neq P[\bar{a}]^{I'}$ or $P[\bar{a}]^J \neq P[\bar{a}]^{J'}$. Since \prec is irreflexive, it holds that $I \cong_{\prec P[\bar{a}]} I'$ and $J \cong_{\prec P[\bar{a}]} J'$. Hence by Proposition 5.9,

$$P[\bar{a}]^I = P[\bar{a}]^{\Upsilon_{\Delta}(I, J)} = P[\bar{a}]^{\Upsilon_{\Delta}(I', J')} = P[\bar{a}]^{I'}, \text{ and}$$

$$P[\bar{a}]^J = P[\bar{a}]^{\Upsilon_{\Delta}(J, I)} = P[\bar{a}]^{\Upsilon_{\Delta}(J', I')} = P[\bar{a}]^{J'}.$$

We obtain a contradiction.

It follows that there can be at most one pair (I, J) satisfying this condition. Moreover, if such a pair, say (I, J) , exists then also the symmetric pair (J, I) satisfies the condition and consequently, I and J have to be identical.

Now, the proof of totality follows easily. The pair $(I_o^{\Delta\downarrow}, I_o^{\Delta\uparrow})$ is the maximal oscillating pair of the stable operator. Every oscillating pair (I, J) of the stable operator satisfies $\mathsf{T}_\Delta(I, J) = I$ and $\mathsf{T}_\Delta(J, I) = J$. By the previous paragraph, it follows that $I_o^{\Delta\downarrow} = I_o^{\Delta\uparrow} = I_o^\Delta$.

We also just proved that I_o^Δ is the unique structure that extends I_o and satisfies the fixpoint equation $\mathsf{T}_\Delta(I, I) = I$. We derive for all I extending I_o :

$$\begin{aligned} I = I_o^\Delta & \text{ iff } I = \mathsf{T}_\Delta(I, I) \\ & \text{ iff } I = \Gamma_\Delta(I) \text{ (Proposition 4.8)} \\ & \text{ iff for each defined domain atom } P[\bar{a}], P[\bar{a}]^I = \varphi_P[\bar{a}]^I \\ & \text{ iff } I \models \text{comp}(\Delta). \end{aligned}$$

□

We obtain the following corollary.

COROLLARY 6.10. *Suppose a definition Δ over τ and a theory T_o over $\tau^o \subseteq \tau_\Delta^o$ such that for any model K_o of T_o , Δ is a definition over some well-founded order \prec in K_o . Then $T_o \wedge \Delta$ and $T_o \wedge \text{comp}(\Delta)$ are logically equivalent.*

Example 6.11. Consider the definition Δ of Example 4.17:

$$\Delta_{\text{even}} := \left\{ \begin{array}{l} \forall x (E(x) \leftarrow x = 0) , \\ \forall x (E(s(x)) \leftarrow \neg E(x)) \end{array} \right\}.$$

The transitive closure of the reduction $\{(E[n], E[n+1]) \mid n \in \mathbb{N}\}$ is a strict reduction of Δ_{even} in the natural numbers. Consequently, $T_{\mathbb{N}} \wedge \Delta_{\text{even}}$ and $T_{\mathbb{N}} \wedge \text{comp}(\Delta_{\text{even}})$ are equivalent.

Note that in the natural numbers, $\text{PID}(\Delta_{\text{even}})$ is inconsistent. Indeed, the sets $\{0, 2, 4, 6, \dots\}$ and $\{0, 1, 3, 5, \dots\}$ are both minimal sets containing 0 and containing $n+1$ if n is not contained. Consequently, there is no least such set.

7. KNOWLEDGE REPRESENTATION

Mathematicians use inductive definitions to construct exact, precise mathematical objects. Unfortunately, in practical settings of knowledge representation, precise, clear-cut information is usually lacking and the precision of mathematics is not attainable. This phenomenon has been the motivation for the research areas of non-monotonic and common-sense reasoning. We demonstrate here that although ID-logic is founded on a principle from mathematics, this principle is also deeply linked to some of the most important common-sense knowledge principles.

7.1 Definitions in KR

In [Brachman and Levesque 1982], it was observed that definitional knowledge is an important form of human expert knowledge. This was the motivation for extending the KR-one system with a Tbox to represent (non-recursive) definitions. This work laid the foundations for the area of Description logics [Baader et al. 2002]. The current generation of description logics were extended with fixpoint constructs to

represent monotone induction and Co-induction. ID-logic fits into the paradigm of description logics, in the sense of its separation of definitional and assertional knowledge, and extends it by allowing definitions for n-ary predicates and non-monotone inductive definitions.

Inductive definitions may be useful to express complex properties of relations, e.g. properties involving reachability. As an example, consider the concept of a Hamiltonian cycle in a graph G : a path in G that reaches every vertex of the graph exactly once and ends in its first vertex. A Hamiltonian path is characterized by its successor relation. This successor function is a subgraph of G , each successor has at most one successor and is the successor of at most one vertex, and each vertex is reachable from some initial vertex a :

$$\left. \begin{array}{l} \forall x \forall y (Suc(x, y) \supset G(x, y)) \\ \forall x \forall y \forall z (Suc(x, y) \wedge Suc(x, z) \supset y = z) \\ \forall x \forall y \forall z (Suc(y, x) \wedge Suc(z, x) \supset y = z) \\ \forall x Reached(x) \\ \left\{ \begin{array}{l} \forall x (Reached(x) \leftarrow Suc(a, x)), \\ \forall x \forall y (Reached(x) \leftarrow Reached(y) \wedge Suc(y, x)) \end{array} \right\} \end{array} \right\}$$

This theory illustrates the use of definitions in terms of open predicates, and the combined use of classical logic and definitions. This example is a benchmark example of Answer Set Programming [Marek and Truszczyński 1999; Niemelä 1999]. In the model expansion paradigm [Mitchell and Ternovska 2005], Hamiltonian paths are computed by extending a given input interpretation K_0 interpreting the symbols G and a , into a model of the above theory.

7.2 Domain Closure Axioms and Unique Names Axioms

One of the first non-monotonic reasoning principles that was investigated is the *Domain Closure Assumption* [Reiter 1980]. It is the assumption that the domain of discourse contains no other objects than those named by ground terms. For a vocabulary τ , it can be formalized by the following *Domain Closure Axiom* ($DCA(\tau)$) in ID-logic:

$$\exists U \left(\left\{ \begin{array}{l} \dots \\ \forall \bar{x} U(f(\bar{x})) \leftarrow U(x_1) \wedge \dots \wedge U(x_n) \\ \dots \end{array} \right\} \wedge \forall x U(x) \right),$$

where the definition contains one rule for every n-ary object or function symbol f ($n \geq 0$) of τ . Base cases of the definition are the rules $U(c) \leftarrow \mathbf{t}$ for object symbols $c \in \tau$. The inductive rules close the domain elements denoted by these constants under application of the functions. Consequently, the definition defines U as the set of all objects named by ground terms. The assertion $\forall x U(x)$ states that this set coincides with the domain of discourse. This formula is SO[ID], but skolemization of U turns it into FO[ID]. When τ contains no function symbols of arity > 0 , the definition is non-recursive and the formula is equivalent to:

$$\forall x (x = c_1 \vee \dots \vee x = c_n)$$

where c_1, \dots, c_n are the object symbols in τ .

The Domain Closure Axiom is often applied in combination with the *Unique Name Axioms*, $UNA(\tau)$, which express that different terms represent different objects:

$$\begin{aligned} & \forall \bar{x} \forall \bar{y} (f(\bar{x}) \neq g(\bar{y})) \\ & \forall \bar{x} \forall \bar{y} (f(\bar{x}) = f(\bar{y}) \supset x_1 = y_1 \wedge \dots \wedge x_n = y_n), \end{aligned}$$

for every pair of constant or function symbols $f, g \in \tau$. The combination of $DCA(\tau)$ and $UNA(\tau)$ is implicitly present in logics with Herbrand model semantics, such as logic programming. Observe that these principles are found also in Peano's axioms, where of course they do not model default assumptions but solid mathematical truth. In fact, the theory $T_{\mathbb{N}}$ of Example 4.3 consists of $DCA(\{0, S/1\})$ and $UNA(\{0, S/1\})$. This shows that $DCA(\tau)$ is a generalized induction axiom.

7.3 Closed World Assumption and Default Inheritance

When a definition Δ contains no open predicates (and only then!), an interesting alternative interpretation of Δ is possible: as a set of material implications augmented with the *Closed World Assumption* [Reiter 1978]: the assumption that an atom is false unless it can be proven from the material implications. As a consequence, ID-logic is suitable to model certain forms of default reasoning and default inheritance. Below is an ID-logic representation of a well-known toy-example:

$$\begin{aligned} & \forall x (Bird(x) \wedge \neg Ab(x) \supset Fly(x)) \\ & \forall x (Penguin(x) \supset \neg Fly(x)) \\ & \left. \begin{array}{l} \forall x (Bird(x) \leftarrow Penguin(x)) \\ \forall x (Ab(x) \leftarrow Penguin(x)) \\ Bird(Tweety) \leftarrow \mathbf{t} \\ Penguin(Clyde) \leftarrow \mathbf{t} \end{array} \right\} \end{aligned}$$

This theory, augmented with $DCA(\tau) + UNA(\tau)$, where τ consists of all free symbols in the above theory, can be viewed as a representation of the state of the world in which all defaults hold. For instance, it entails the desired default properties $fly(Tweety)$ and $\neg Penguin(Tweety)$ and $\neg fly(Clyde)$. Adding the atomic rule $Penguin(Tweety) \leftarrow \mathbf{t}$ to the definition has the non-monotonic effect of deriving $\neg Fly(Tweety)$. In general, complex inheritance hierarchies can be modelled in this way, using the methodologies for representing default inheritance developed in the context of Logic Programming, e.g. in [You et al. 1999].

7.4 Temporal Reasoning

Temporal reasoning has always been a major test case for knowledge representation formalisms. In this section, we illustrate the use of iterated inductive definitions and the tools introduced in the previous sections in the context of a temporal reasoning example. A more general study of this topic was made in [Denecker and Ternovska 2004a; 2007].

The problem domain we want to axiomatise is a variant of the well-known blocks world planning domain. Given is a set of blocks on a table. A robot can execute action *pickup* to pick up a block and action *put* to put the block on the table or

another block. When a block is picked up on top of which other blocks are piled up, the effect of this action depends on the size of the pile: if the size exceeds a given threshold, then the pile collapses and all blocks fall on the table with exception of the picked one. Otherwise, the robot can hold and move the block and all blocks piled on it. We assume that initially all blocks are on the table and the robot's hand is empty.

We model this problem in a variant of situation calculus, using a *sorted* version of ID-logic. Sorts are *sit*, *action*, *block* and *nat*. The symbols of sort *block* are the constant *Table* and one constant for each block in the problem domain. We use an object symbol S_0 and a binary function symbol *do* of sort *sit*, where *do* has arguments of sorts *action* and *sit*, a unary symbol *pickup* and binary symbol *put* of sort *action*, both with arguments of sort *block*, and, finally, a constant n of sort *nat* to represent the threshold. To formalize the domain of each sort, we use a typed version of the domain closure and unique name axioms. For example, for the situation sort *sit*, the theory $DCA(\tau_{sit}) \wedge UNA(\tau_{sit})$ is:

$$\begin{aligned} & \forall a \forall s S_0 \neq do(a, s) \\ & \forall a \forall s \forall a' \forall s' (do(a, s) = do(a', s') \supset a = a' \wedge s = s') \\ & \exists P(\{ P(S_0) \leftarrow \mathbf{t}, \forall a \forall s (P(do(a, s)) \leftarrow P(s)) \} \wedge \forall s P(s)) \end{aligned}$$

Similar theories can be defined for the sorts *action* and *block*. Since these sorts do not have recursively applicable function symbols (e.g. action functions with an action argument), the definitions in $DCA(\tau_{action})$ and $DCA(\tau_{block})$ are non-recursive. We denote the union of these theories by \mathcal{F} . This theory has only one model \mathcal{A} , modulo isomorphism.

The definition in Figure 1 describes the effects of all actions. Predicate *Above* represents the transitive closure of the predicate *On*. The first rule describes the

$$\left\{ \begin{array}{l} \forall s \forall b (On(b, Table, S_0) \leftarrow b \neq Table), \\ \forall a \forall s \forall b \forall b' (On(b, b', a, s) \leftarrow a = do(put(b, b'), s), \\ \forall a \forall s \forall b \forall b' (On(b, Table, do(pickup(b'), s)) \leftarrow Above(b, b', s) \wedge Pilesize(b', n, s) \wedge \\ \quad n > Tresh), \\ \forall a \forall s \forall b, b' (On(b, b', do(a, s)) \leftarrow On(b, b', s) \wedge \\ \quad \neg(a = pickup(b) \vee \exists c \exists n (a = pickup(c) \wedge Above(b', c, s) \\ \quad \wedge Pilesize(c, n, s) \wedge n > Tresh))), \\ \forall a \forall s \forall b (Clasped(b, do(a, s)) \leftarrow a = pickup(b)), \\ \forall a \forall s \forall b (Clasped(b, do(a, s)) \leftarrow Clasped(b, s) \wedge \neg \exists b' a = put(b, b')), \\ \forall b \forall b' \forall s (Above(b, b', s) \leftarrow On(b, b', s)), \\ \forall b \forall b' \forall s (Above(b, b', s) \leftarrow \exists c (On(b, c, s) \wedge Above(c, b', s))), \\ \forall b \forall s (Pilesize(b, 0, s) \leftarrow \neg \exists b' (On(b', b, s))), \\ \forall b \forall n \forall s (Pilesize(b, n+1, s) \leftarrow \exists b' (On(b', b, s) \wedge Pilesize(b', n, s))) \end{array} \right.$$

Fig. 1. Inductive definitions in situation calculus

initial situation, the second expresses the effect of putting down a block, the third of picking up a pile of blocks which is too large, and the fourth expresses what

actions leave two blocks on each other. The next two rules express when a block is clasped. The four final rules are inductive definitions of *Above* and *Pilesize*.

The example illustrates nicely the concept of an iterated inductive definition. The fluents *On* and *Clasped* are defined in terms of the state of the fluents in the previous situation through both positive and negative inductive rules. In particular, the fourth rule defining *On* contains negative occurrences of *Above* and *Pilesize*, which in turn are defined in terms of *On* through monotone induction. The result is an iterated inductive definition on the well-founded set of situations with alternating phases of monotone and non-monotone induction.

In the propositions below, we analyse the above definition. In the context of the structure \mathcal{A} satisfying \mathcal{F} , we define the relation \prec as the set of all tuples:

$$\begin{aligned} & (\textit{Above}[c, c', s], \textit{On}[b, b', do^A(a, s)]), (\textit{Pilesize}[c, n, s], \textit{On}[b, b', do^A(a, s)]), \\ & (\textit{On}[b, b', s], \textit{On}[b, b', do^A(a, s)]), \\ & (\textit{Clasped}[b, s], \textit{Clasped}[b, do^A(a, s)]), \\ & (\textit{On}[c, c', s], \textit{Above}[b, b', s]), \\ & (\textit{Above}[c, c', s], \textit{Above}[b, b', s]), \\ & (\textit{On}[b, b', s], \textit{Pilesize}[c, n, s]), \\ & (\textit{Pilesize}[b, n, s], \textit{Pilesize}[b', m, s]), \end{aligned}$$

for all blocks b, b', c, c' , for all natural numbers n, m , for all actions a and for all situations s .

PROPOSITION 7.1. *The relation \prec is a reduction relation of Δ in \mathcal{A} . Its reflexive and transitive closure is a pre-well-order.*

PROOF. Each of the seven rows in the definition of \prec corresponds to one of the seven recursive rules in Δ with defined predicates in the body. It is straightforward to verify that the tuples on each row specify a reduction relation of the corresponding rule. Consequently, \prec is a reduction relation of Δ . \square

For every fluent symbol F in $\{\textit{On}, \textit{Clasped}, \textit{Above}, \textit{Pilesize}\}$, let Δ_F be the set of rules defining F .

PROPOSITION 7.2. *$\{\Delta_{\textit{On}}, \Delta_{\textit{Clasped}}, \Delta_{\textit{Above}}, \Delta_{\textit{Pilesize}}\}$ is a reduction partition in \mathcal{A} , and each of these definitions is total.*

PROOF. It is straightforward to see that $\{\Delta_{\textit{On}}, \Delta_{\textit{Clasped}}, \Delta_{\textit{Above}}, \Delta_{\textit{Pilesize}}\}$ is a reduction partition under the transitive closure \prec^* of \prec . The totality of the partition follows from the fact that all component definitions are positive. \square

PROPOSITION 7.3. *$\Delta_{\textit{On}}$ and $\Delta_{\textit{Clasped}}$ are definitions by well-founded induction over the successor relation between situations in \mathcal{A} .*

PROOF. By Proposition 5.5, the restriction $\prec_{\mathcal{A}}^{\Delta_{\textit{On}}}$ of \prec to tuples with an atom $\textit{On}(b, b', s)$ in the second argument is a reduction relation of $\Delta_{\textit{On}}$ in \mathcal{A} . Its transitive closure is also a reduction relation and moreover, it is a strict well-founded order. A similar argument holds for $\Delta_{\textit{Clasped}}$ and $\prec_{\mathcal{A}}^{\Delta_{\textit{Clasped}}}$, the restriction of \prec to tuples with a *Clasped* domain atom in the second argument. \square

We obtain as a corollary the following equivalence result.

PROPOSITION 7.4. *The theories $\mathcal{F} \cup \{\Delta\}$ and $\mathcal{F} \cup \{comp(\Delta_{On}) \wedge comp(\Delta_{Clasped}) \wedge PID(\Delta_{Above}) \wedge PID(\Delta_{Pilesize})\}$ are logically equivalent.*

The non-monotone forms of inductive definitions in ID-logic, have interesting applications in *causal and temporal reasoning*. In [Denecker et al. 1998], it was observed that the construction process of an inductive definition formally mimics the physical process of the propagation of causes and effects in a dynamic system. Based on this idea, a general solution to model ramifications using inductive definitions was proposed. In [Denecker and Ternovska 2004a; 2007], this solution was integrated in situation calculus. This solution is currently the most general solution for the ramification problem in the situation calculus.

8. CONCLUSION

Extending work of the first author in [Denecker 2000; Denecker et al. 2001], we presented a logical formalization of several *informal* forms of inductive definitions in mathematics, allowing a uniform, rule-based representation of non-recursive definitions, monotone inductive definitions and non-monotone forms such as induction over a well-founded order and iterated inductive definitions. In order to compensate for classical first-order logic's representational weakness on inductive definability, we extended it with definition formulas. The main technical theorems here are the Modularity and Totality theorems and the theorems analysing specific subclasses of definitions and translating them into classical logic.

ID-logic is inspired by semantical studies of logic programming, and formally extends logic programming and variants such as Datalog and Abductive Logic Programming under the well-founded semantics. Thus, we succeeded in integrating classical logic and logic programming in a coherent and conceptually clean way. We informally discussed the strong relationship with IID, a mathematical formalism for representing iterated inductive definitions. As an extension of classical logic with an alternating fixpoint construct, ID-logic fits into the paradigm of fixpoint logics. It also fits in the paradigm of Description logics, well-known in Knowledge Representation, in the sense of its separation of definitional and assertional knowledge. We also argued that despite its mathematical origin, the concept of inductive definition is closely related to principles of common-sense reasoning, including Domain Closure, Closed World Assumption, defaults and temporal and causal reasoning. We also showed a natural example of an iterated inductive definition in the context of situation calculus for temporal reasoning and applied the technical apparatus developed in this paper to demonstrate the equivalence of this definition to a situation calculus axiomatization in classical logic based on completion and circumscription.

Thus, it appears that the notion of definition and its inductive generalizations emerges as a unifying theme in many areas of mathematics and computational logic. Hence, its study could improve insight in the interrelations between these areas and lead to synergy.

REFERENCES

- ACZEL, P. 1977. An introduction to inductive definitions. In *Handbook of Mathematical Logic*, J. Barwise, Ed. North-Holland Publishing Company, 739–782.
- APT, K., BLAIR, H., AND WALKER, A. 1988. Towards a theory of Declarative Knowledge. In *Foundations of Artificial Intelligence*, Vol. 1, pp. 1–21. North-Holland.
- ACM Transactions on Computational Logic, Vol. V, No. N, October 2006.

- dations of Deductive Databases and Logic Programming*, J. Minker, Ed. Morgan Kaufmann, 89–148.
- BAADER, F., CALVANESE, D., MCGUINNESS, D., NARDI, D., AND PATEL-SCHNEIDER, P., Eds. 2002. *The Description Logic Handbook. Theory, Implementation and Applications*. Cambridge University Press.
- BRACHMAN, R. J. AND LEVESQUE, H. J. 1982. Competence in Knowledge Representation. In *National Conference on Artificial Intelligence (AAAI'82)*. 189–192.
- BUCHHOLZ, W., FEFERMAN, S., AND SIEG, W. P. W. 1981. *Iterated Inductive Definitions and Subsystems of Analysis: Recent Proof-Theoretical Studies*. Lecture Notes in Mathematics, vol. 897. Springer-Verlag.
- CLARK, K. L. 1978. Negation as failure. In *Logic and Databases*, H. Gallaire and J. Minker, Eds. Plenum Press, 293–322.
- COMPTON, K. 1993. A Deductive System for Existential Least Fixpoint Logic. *Journal of Logic and Computation* 3, 2, 197–213.
- DENECKER, M. 1998. The well-founded semantics is the principle of inductive definition. In *Logics in Artificial Intelligence (JELIA'98)*, J. Dix, L. Fariñas del Cerro, and U. Furbach, Eds. Lecture Notes in Artificial Intelligence, vol. 1489. Springer-Verlag, 1–16.
- DENECKER, M. 2000. Extending classical logic with inductive definitions. In *First International Conference on Computational Logic (CL'2000)*, J. Lloyd et al., Ed. Lecture Notes in Artificial Intelligence, vol. 1861. Springer, 703–717.
- DENECKER, M. 2004. What's in a model? Epistemological analysis of Logic Programming. In *Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR'04)*. 106–113. URL = http://www.cs.kuleuven.ac.be/cgi-bin-dtai/publ_info.pl?id=41086.
- DENECKER, M., BRUYNOOGHE, M., AND MAREK, V. 2001. Logic programming revisited: Logic programs as inductive definitions. *ACM Transactions on Computational Logic* 2, 4 (October), 623–654.
- DENECKER, M., MAREK, V., AND TRUSZCZYŃSKI, M. 2000. Approximating operators, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In *Logic-based Artificial Intelligence*, J. Minker, Ed. Kluwer Academic Publishers, Chapter 6, 127–144.
- DENECKER, M. AND TERNOVSKA, E. 2004a. Inductive Situation Calculus. In *Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR'04)*. 545–553.
- DENECKER, M. AND TERNOVSKA, E. 2004b. A logic of non-monotone inductive definitions and its modularity properties. In *Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'04)*, V. Lifschitz and I. Niemelä, Eds.
- DENECKER, M. AND TERNOVSKA, E. 2007. Inductive situation calculus. *Artificial Intelligence*.
- DENECKER, M., THESEIDER DUPRÉ, D., AND VAN BELLEGHEM, K. 1998. An inductive definition approach to ramifications. *Linköping Electronic Articles in Computer and Information Science* 3, 7, 1–43. URL: <http://www.ep.liu.se/ea/cis/1998/007/>.
- EBBINGHAUS, H. AND FLUM, J. 1999. *Finite Model Theory*, Second ed. Springer-Verlag.
- FEFERMAN, S. 1970. Formal theories for transfinite iterations of generalised inductive definitions and some subsystems of analysis. In *Intuitionism and Proof theory*, A. Kino, J. Myhill, and R. Vesley, Eds. North Holland, 303–326.
- FITTING, M. 1985. A Kripke-Kleene Semantics for Logic Programs. *Journal of Logic Programming* 2, 4, 295–312.
- FITTING, M. 2002. Fixpoint semantics for logic programming - a survey. *Theoretical Computer Science* 278, 25–51.
- GELFOND, M. AND LIFSCHITZ, V. 1988. The stable model semantics for logic programming. In *International Joint Conference and Symposium on Logic Programming (IJCSLP'88)*. MIT Press, 1070–1080.
- GRAEDEL, E. AND KREUTZER, S. 2003. Will deflation lead to depletion? on non-monotone fixed point inductions. In *IEEE Symposium on Logic in Computer Science (LICS'03)*. 158–.
- GUREVICH, Y. AND SHELAH, S. 1986. Fixed-point Extensions of First-Order Logic. *Annals of Pure and Applied Logic* 32, 265–280.
- ACM Transactions on Computational Logic, Vol. V, No. N, October 2006.

- IMMERMAN, N. 1986. Relational queries computable in polynomial time. *Information and Control* 68, 86–104.
- KAKAS, A. C., VAN NUFFELEN, B., AND DENECKER, M. 2001. A-system : Problem solving through abduction. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, B. Nebel, Ed. Vol. 1. Morgan Kaufmann Publishers, Inc., 591–596.
- KREISEL, G. 1963. Generalized inductive definitions. Tech. rep., Section III in the Stanford University report on the Foundations of Analysis.
- LIFSCHITZ, V. 1994. Circumscription. In *Handbook of Logic in AI and Logic Programming*. Vol. 3. Oxford University Press, 298–352.
- LIFSCHITZ, V. AND TURNER, H. 1994. Splitting a logic program. In *International Conference on Logic Programming (ICLP'94)*. 23–37.
- LIVCHAK, A. 1983. The Relational Model for Process Control. *Automatic Documentation and Mathematical Linguistics* 4, 27–29.
- MAREK, V. W. AND TRUSZCZYŃSKI, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25 Years Perspective*, K. Apt, V. Marek, M. Truszczyński, and D. Warren, Eds. Springer-Verlag, pp. 375–398.
- MAREK, W. 1989. Stable theories in autoepistemic logic. *Fundamenta Informaticae* 12, 2, 243–254.
- MARIËN, M., WITTOCK, J., AND DENECKER, M. 2006. The IDP framework for declarative problem solving. In *Search and Logic: Answer Set Programming and SAT*. 19–34. URL = <http://www.cs.kuleuven.ac.be/cgi-bin/dtai/publ.info.pl?id=42379>.
- MARTIN-LÖF, P. 1971. Hauptsatz for the intuitionistic theory of iterated inductive definitions. In *Second Scandinavian Logic Symposium*, J. Fenstad, Ed. 179–216.
- MITCHELL, D. AND TERNOVSKA, E. 2005. A framework for representing and solving np search problems. In *AAAI'05*. AAAI Press/MIT Press, 430–435.
- MOORE, R. 1983. Semantical Considerations on non-monotonic logic. In *International Joint Conference on Artificial Intelligence (IJCAI'83)*. 272–279.
- MOSCHOVAKIS, Y. N. 1974a. *Elementary Induction on Abstract Structures*. North-Holland Publishing Company, Amsterdam- New York.
- MOSCHOVAKIS, Y. N. 1974b. On non-monotone inductive definability. *Fundamenta Mathematica* 82, 39–83.
- NIEMELÄ, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25, 3,4, 241–273.
- PELOV, N. AND TERNOVSKA, E. 2005. Reducing inductive definitions to propositional satisfiability. In *International Conference on Logic Programming (ICLP'05)*. 221–234.
- POHLERS, W. 1989. *Proof Theory: an introduction*. Springer-Verlag.
- POST, E. 1943. Formal reduction of the general combinatorial decision problem. *American Journal of Mathematics* 65, 197–215.
- RAO, P., RAMSKRISHNAN, I., SAGONAS, K., SWIFT, T., WARREN, D., AND FREIRE, J. 1997. XSB: A system for efficiently computing well-founded semantics. In *International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'97)*. Lecture Notes in Computer Science, 1265. Springer-Verlag, 430–440.
- REITER, R. 1978. On Closed World Data bases. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds. Plenum Press, New York, 55–76.
- REITER, R. 1980. Equality and domain closure in first-order databases. *Journal of the ACM* 27, 235–249.
- SCHLIPF, J. 1995a. Complexity and undecidability results in logic programming. *Annals of Mathematics and Artificial Intelligence* 15, 257–288.
- SCHLIPF, J. 1995b. The expressive powers of the logic programming semantics. *Journal of Computer and System Sciences* 51, 64–86.
- SPECTOR, C. 1961. Inductively defined sets of natural numbers. In *Infinitistic Methods (Proc. 1959 Symposium on Foundation of Mathematics in Warsaw)*. Pergamon Press, Oxford, 97–102.

- TARSKI, A. 1955. Lattice-theoretic fixpoint theorem and its applications. *Pacific journal of Mathematics* 5, 285–309.
- TERNOVSKAIA, E. 1998a. Causality via inductive definitions. In *Working Notes of "Prospects for a Commonsense Theory of Causation"*, AAAI Spring Symposium Series, March 23-28.
- TERNOVSKAIA, E. 1998b. Inductive definability and the situation calculus. In *Transaction and Change in Logic Databases*. Lecture Notes in Computer Science, vol. 1472. Springer-Verlag.
- VAN GELDER, A. 1993. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences* 47, 1, 185–221.
- VAN GELDER, A., ROSS, K. A., AND SCHLIPF, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM* 38, 3, 620–650.
- VARDI, M. 1982. The complexity of relational query languages. In *Proc. of the 14th ACM Symposium on the Theory of Computing*. 137–146.
- VENNEKENS, J., GILIS, D., AND DENECKER, M. 2005. Splitting an operator: Algebraic modularity results for logics with fixpoints semantics. *ACM Transactions on Computational Logic*. URL = http://www.cs.kuleuven.ac.be/cgi-bin-dtai/publ_info.pl?id=41545.
- VERBAETEN, S., DENECKER, M., AND DE SCHREYE, D. 2000. Compositionality of normal open logic programs. *Journal of Logic Programming* 41, 3 (Mar.), 151–183.
- YOU, J.-H., WANG, X., AND YUAN, L. Y. 1999. Compiling defeasible inheritance networks to general logic programs. *Artificial Intelligence* 113, 1-2 (September), 247 – 268.