Open access • Book Chapter • DOI:10.1007/3-540-44957-4_47

# Extending Classical Logic with Inductive Definitions — Source link

Marc Denecker

**Institutions:** Katholieke Universiteit Leuven

**Published on:** 01 Jul 2000 - Lecture Notes in Computer Science (Springer, Berlin, Heidelberg)

**Topics:** Philosophy of logic, Classical logic, Deductive reasoning, Intermediate logic and Many-valued logic

# Extending Classical Logic with Inductive Definitions

**Marc Denecker**
Department of Computer Science, K.U.Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium.
Phone: +32 16 327555 — Fax: +32 16 327996
email: marcd@cs.kuleuven.ac.be

**Abstract**

The goal of this paper is to extend classical logic with a generalized notion of inductive definition supporting positive and negative induction, to investigate the properties of this logic, its relationships to other logics in the area of non-monotonic reasoning, logic programming and deductive databases, and to show its application for knowledge representation by giving a typology of definitional knowledge.

## Introduction

Since the early days of Computer Science, when John McCarty started to investigate the role of logic for computing and artificial intelligence, the key attraction of logic was the promise of a natural representation of knowledge. By the end of the seventies, it had become clear that representing common sense knowledge in classical logic, despite its clear and well-understood intuitive reading, was a complex task. Two major problems were seen as the causes: first, very often common sense knowledge turns out to be partial and incomplete. Second, in many applications an unrealistic amount of axioms seemed necessary to represent the human knowledge; the prototypical example in this respect is the frame problem (McCarthy & Hayes 1969). To solve these problems, non-monotonic logics such as circumscription and default logic, incorporate strong logical closure principles that allow to reason about partial and incomplete knowledge, and allow a compact representation of human knowledge.

This paper defines and investigates a conservative extension of classical logic with a less general nonmonotonic closure principle: the principle of definition and inductive definition. In the context of nonmonotonic reasoning, definitions have received little attention so far. This is remarkable for several reasons. The study of definitions in knowledge representation has a long tradition which extends to the origins of Western philosophy. The problem of defining a *natural kind* as studied by the ancient Greek philosophers is in many ways an instance avant la lettre of the general problem of common sense knowledge representation.

Also in the context of modern AI, the study of the role of definitions in common sense knowledge representation has a long tradition. As an outcome of a series of investigations into the semantics of semantic networks[1], Brachman and Levesque (Brachman & Levesque 1982) observed that an important component of expert knowledge is knowledge of the *defining properties* of concepts, and that it is crucial to distinguish between *defining properties* of concepts and *assertional knowledge* on concepts. Description logics are based on this idea, and consist of a Tbox to represent definitional knowledge and an Abox to represent *assertional knowledge*.

Another area related to nonmonotonic reasoning in which definitions have been prominent is in the context of logic programming and abductive (or open) logic programming. One of the original ideas on the declarative semantics of logic programs with negation as failure was to interpret a logic program as a *definition* of its predicates. This view is underlying Clark's completion semantics (Clark 1978). In (Denecker 1995; Denecker & De Schreye 1995; Van Belleghem, Denecker, & De Schreye 1997), this view was extended to abductive logic programs and open logic programs and the relation with description logics was shown.

In the context of non-monotonic reasoning, Reiter (Reiter 1996) and Amati et al (Amati, Carlucci Aiello, & Pirri 1997) observed that an important method for analysis and computation in common sense knowledge representation is to *compile* non-monotonic theories into first order definitions (i.e. Clark completions). They argue that the advantages of this compilation are that it clarifies the meaning of the original theory and that it yields theories that are better suited for computational purposes. Recently (Denecker, Theseider Dupré, & Van Belleghem 1998) and (Ternovskaia 1998) showed a strong correspondence between induction and *causality* and investigated the use of inductive definitions to represent temporal and causal knowledge.

The above suggests that the study of induction and inductive definitions could provide a valuable contribution to the area of common sense knowledge representation, nonmonotonic reasoning and logic programming.

The study of induction could be defined as the study

[1] See (Reichgelt 1991) for a discussion of this topic.

of construction techniques in mathematics. In general, an inductive definition *defines* a relation (or a collection of relations) through a constructive process of iterating a recursive *recipe* that defines new instances of the relation in terms of the presence or absence of other tuples of the same relation or other relations. The recipe naturally defines an operator: this operator maps a relation (or set of relations) to the relation (or set of) that can be obtained by applying the recipe.

Standard work was done by Moschovakis (Moschovakis 1974) and Aczel (Aczel 1977). These treatments study the theoretical expressivity of *positive* or *monotone induction*. One spin-off of this field is fixpoint logic (Gurevich & Shelah 1986), a subarea of databases (Abiteboul, Hull, & Vianu 1995). As shown in (Denecker 1998), the abstract positive inductive definition logic defined in (Aczel 1977) is formally isomorphic with the formalism of propositional Horn programs under least model semantics. In the case of Horn programs, the associated operator corresponds to the immediate consequence operator.

In mathematical applications, also non-monotone forms of induction can be distinguished:

- stratified induction and a generalisation, induction in well-founded sets. In stratified induction the domain of the defined concept(s) can be stratified (possibly in transfinite number of levels) such that higher level instances of the concept are defined positively or negatively in terms of lower level instances of the predicate. Two well-known examples are the following definition of even numbers:

$$even :: \left\{ \begin{array}{l} even(0) \leftarrow \\ even(s(x)) \leftarrow \neg even(x) \end{array} \right\}$$

  and the definition of the ordinal powers of a monotone operator as used in the Tarski-Kleene least fixpoint theory.

- induction in the context of well-founded sets. Here a concept at a higher level of the set is defined in terms of the concept at lower levels in a monotone or nonmonotone way. An example of definition with positive and negative induction in the context of well-founded sets is given in (Denecker 1998): the definition of *depth* or *rank* of an element in a well-founded set: it is the least ordinal strictly larger than the depths of strictly less elements. Also the definition of ordinal powers can be seen as an application of this principle.

Inflationary fixpoint logic is a well-known extension of fixpoint logic for nonmonotone operators. However, as argued in (Denecker 1998), this extension does not match the intuition of non-monotone induction. For example, the semantics of the nonmonotone definition of $even$ in inflationary semantics would be the set of all natural numbers. In general, the inflationary fixpoint is not even a fixpoint of the original operator, and if it is, it is not necessarily a minimal fixpoint.

Formalizations of definitions with positive and negative induction were investigated in the area of Iterated Inductive Definitions (Feferman 1970; Buchholz, Feferman, & Sieg 1981). As argued in (Denecker 1998), the intuition of such formalisms is simple and natural and corresponds with the many notions of stratification in logic programming. Yet, it is also shown that encoding even simple inductive definitions is extremely tedious; knowledge representation in such systems is practically impossible.

(Denecker 1998) presents a knowledge theoretic study of a generalized principle of inductive definition in the abstract setting of an (infinitary) propositional logic. This definition logic extends Aczel's infinitary propositional logic for positive induction(Aczel 1977)[2]. When extending the notion of inductive definition to unrestricted forms of induction, a challenging problem arises of defining a uniform semantical principle that assigns the right semantics to all kinds of inductive definitions. The main contribution of (Denecker 1998) was to show that the principle of well-founded model in logic programming (Van Gelder, Ross, & Schlipf 1991) is the suitable mathematical principle of generalized inductive definition. The well-founded model is obtained as the least fixpoint of the 3-valued stable operator (Przymusinski 1990b). The latter operator is a general and robust implementation of the principle of positive induction; negative induction is dealt with by iterating this positive induction operator in a least fixpoint computation. As a consequence, this semantics generalizes all well-known ways of defining concepts. It coincides with least fixpoint and least model semantics in the context of positive induction; it coincides with iteration of the positive induction in the context of iterated (or stratified) systems of inductive definitions; it coincides with Clark completion semantics in case of non-inductive definitions and inductive definitions on a well-founded set; beyond these classes, it gives the intended meaning to mixtures of positive and negative induction in semi-well-founded sets[3].

The present paper is concerned with the role of induction in common sense knowledge representation and the definition of a knowledge representation logic based on inductive definitions. To this end, the abstract logic of (Denecker 1998) is lifted to a predicate logic suitable for representing this generalized notion of inductive definitions in the context of uncertainty and incomplete knowledge. I investigate formal properties and methodological guidelines of this logic, important from the point of view of knowledge representation. A number of important applications of this logic are sketched. The relationship with many other logics are discussed,

[2]Both logics are *abstract* in the sense that infinitary theories and rules are considered.

[3]A semi-order $\leq$ is a reflexive, transitive relation and defines an equivalence relation $x \equiv y$ iff $x \leq y \wedge y \leq x$). The set of equivalence classes is a poset. A semi-order is well-founded if the poset of equivalence classes is well-founded.

in particular logic programming and some of its extensions.

By lack of space, proofs of all theorems are omitted.

## An abstract logic of inductive definitions

In (Denecker 1998), I proposed an extension of Aczel's logic for general monotone and non-monotone inductive definitions. The result is isomorphic with the formalism of infinitary propositional logic programs (with negation) under well-founded semantics. An abstract inductive definition (ID) $D$ in this logic defines a set $Defined(D)$ of symbols, called the set of defined symbols, by a set of rules of the form

$$p \leftarrow B$$

where $p$ is a defined atom and $B$ a set of positive or negative literals. The other atoms are called *open atoms*; their set is denoted $Open(D)$.

In (Denecker 1998) it was argued that Przymusinski's 3-valued extension (Przymusinski 1990a; Przymusinski 1990b) of Gelfond and Lifschitz' stable model operator (Gelfond & Lifschitz 1988) is a general and robust implementation of the principle of positive induction, and that its least fixpoint, the well-founded model (Van Gelder, Ross, & Schlipf 1991) naturally extends the ideas of Iterated Inductive Definitions and gives the right semantics to generalized inductive definitions.

In general, given an ID $D$ and an interpretation $I$ of the open symbols of $D$, there is a unique well-founded model extending $I$. This model will be denoted $\overline{I}^D$. An interpretation is a model of $D$ iff $M = \overline{M_o}^D$ where $M_o$ is the restriction of $M$ to the open symbols. In general, a model of an inductive definition is a partial (3-valued) interpretation. However, for broad classes of definitions, the well-founded model is known to be total (2-valued). These are the cases that are of interest in this paper.

## ID-logic: classical logic with definitions

This section defines a conservative extension of classical logic with definitions. An ID-logic theory $T$ (based on some logical alphabet $\Sigma$) consists of a set of classical logic sentences and a set of definitions. A definition $D$ is a pair of a set $Defined(D)$ of predicates and a set $Rules(D)$ of rules of the form:

$$p(\overline{t}) \leftarrow F$$

where $p \in Defined(D)$ and $F$ an arbitrary first order formula based on $\Sigma$. Predicates of $Defined(D)$ are called the *defined predicates* of $D$; other predicates are called *open predicates* of $D$. A definition defines the defined predicates in terms of the open predicates. More precisely, given some state of the open predicates, the rule set of the definition gives an exhaustive enumeration of the cases in which the defined predicates are true; any defined atom not covered by a rule is defined as false.

A definition will be formally represented as in the example:

$$even, odd :: \left\{ \begin{array}{l} even(0) \leftarrow \\ even(S(x)) \leftarrow odd(x) \\ odd(S(x)) \leftarrow even(x) \end{array} \right\}$$

This is one definition defining two predicates simultaneously.

In ID-logic, definitions are considered as sentences. An ID-logic theory based on $\Sigma$ consists of sentences and may contain different definitions, even for the same predicates. An $\Sigma$-interpretation is a model of an ID-logic theory iff it is a model of all its sentences. So, it suffices to define what is a model of a definition.

The semantics for propositional definitions of section 3 can be lifted easily to the predicate case by use of the grounding technique: the technique of reducing a predicate definition to an infinitary propositional definition[4]. In the context of ID-logic, this grounding of a definition is constructed using the domain and functions of some (general non-Herbrand) interpretation $I$.

To define the grounding the following terminology is needed. Given an alphabet $\Sigma$ and a $\Sigma$-interpretation $I$, define the alphabet $\Sigma_I$ by adding the domain elements of $I$ as constants to $\Sigma$[5]. $I$ is naturally extended to $\Sigma_I$ by defining $I(x) = x$ for each domain element $x$ of $I$. The evaluation of a ground term $t$ of $\Sigma_I$ (which may contain domain elements of $I$) is defined inductively as usual, and is denoted $|t|^I$. Likewise, truth value of a sentence of $\Sigma_I$ is defined by the usual truth recursion.

Given some partial (3-valued) interpretation $I$ and a definition $D$, $I_o$ denotes the restriction of $I$ to the constant, functor and open predicate symbols of $D$. $At_I$ denotes the set of all atoms $p(\overline{d})$ where $p$ is a defined predicate of $D$ and $\overline{d}$ is a tuple of domain elements of $I$. A ground instance of a rule $p(\overline{t}[\overline{x}]) \leftarrow F[\overline{x}]$ with $\overline{x}$ the tuple of all its free variables, is a rule $p(\overline{t}[\overline{d}]) \leftarrow F[\overline{d}]$ obtained by substituting domain elements $\overline{d}$ for $\overline{x}$.

The intuition to get the grounding is as follows. We must replace a rule-instance $p(\overline{t}[\overline{d}]) \leftarrow F[\overline{d}]$ by some logically equivalent set of propositional rules $A \leftarrow B$ where $B$ is a set of literals. We will select $A = p(|\overline{t}[\overline{d}]|^I)$ and $B$ any set of literals of $At_I$ such that if all elements of $B$ were true, then $F[\overline{d}]$ would be true. Or, the bodies of the grounding correspond to the partial models of the rule body $F[\overline{d}]$.

Define a consistent set $S$ of literals of $At_I$ as a set of positive or negative literals based on $At_I$ that does not contain a pair of complementary literals $p(\overline{d}), \neg p(\overline{d})$.

[4] In (Gelder 1993), an alternative way of defining the well-founded semantics of predicate rules is proposed; it is based on a different treatment of positive and negative occurrences of predicates in the body of rules. I believe both techniques are equivalent but haven't proven this.

[5] Note that $\Sigma_I$ may be infinite, even non-countable. This is mathematically and philosophically non-problematic because $\Sigma_I$ is purely used as a semantic device, namely to define the grounding.

Note that there is a one-to-one correspondence between partial interpretations extending $I_o$ and consistent sets of $At_I$-literals. Each partial interpretation $J$ extending $I_o$ defines a unique consistent set $S_J$ of all literals $l$ of $At_I$ that are true in $J$. Vice versa, each consistent set $S$ defines a unique partial interpretation $J_S$ extending $I_o$ such that $J_S(l) = \mathbf{t}$ iff $l \in S$. Moreover, $J_{S_J} = J$.

**Definition 1** *Given an interpretation $I$, the grounding of a definition $D$ w.r.t. $I$, denoted $I$-grounding($D$), is the propositional definition defining all atoms of $At_I$ and consisting of all rules*

$$p(|\overline{t}[\overline{d}]|^I) \leftarrow S_J$$

*such that $p(\overline{t}[\overline{d}]) \leftarrow F[\overline{d}]$ is a ground instance of a rule of $D$ and $J$ is a partial model of $F[\overline{d}]$ extending $I_o$.*

**Definition 2** *A 3-valued interpretation $I$ is a justified interpretation of $D$ iff $S_I$ is the 3-valued (well-founded) model of the grounding of $D$ w.r.t. $I$. $I$ is a justified interpretation of a theory $T$ iff it is a justified interpretation of all its definitions and a (3-valued) model of the classical logic sentences of $T$.*

*An interpretation $I$ is a model of a definition $D$, resp. theory $T$, iff it is a total (i.e. 2-valued) justified interpretation of $D$, resp. $T$.*

The above model theory is based on total, general non-Herbrand models. As a consequence, ID-logic is an extension of classical logic. The restriction to total models is not only necessary to get a extension of classical logic, but also because of methodological constraints on the use of definitions, as explained in the next section.

**Example 1** *The first example shows that different definitions are independent and interact in a monotonic way. Consider the theory consisting of three definitions.*

$$\left\{ \begin{array}{l} father :: \left\{ \begin{array}{l} father(x,y) \leftarrow parent(x,y) \wedge \\ \qquad\qquad\quad male(x) \end{array} \right\} \\ mother :: \left\{ \begin{array}{l} mother(x,y) \leftarrow parent(x,y) \wedge \\ \qquad\qquad\quad female(x) \end{array} \right\} \\ parent :: \left\{ \begin{array}{l} parent(x,y) \leftarrow father(x,y) \\ parent(x,y) \leftarrow mother(x,y) \end{array} \right\} \end{array} \right\}$$

*Note that in the first definition, father depends on parent, while in the third, parent depends on father. However, the semantics of a set of definitions is monotonically composed of the semantics of its definitions. Since none of these definitions is recursive, each is equivalent with its completed definition. Consequently, this triple of definitions is equivalent with the FOL theory:*

$$\left\{ \begin{array}{l} father(x,y) \leftrightarrow parent(x,y) \wedge male(x) \\ mother(x,y) \leftrightarrow parent(x,y) \wedge female(x) \\ parent(x,y) \leftrightarrow father(x,y) \vee mother(x,y) \end{array} \right\}$$

*One can observe that if $male(x) \leftrightarrow \neg female(x)$ holds, then the definition of parent is redundant.*

*Compare this theory with the simultaneous definition obtained by merging the three definitions in one:*

$$\left\{ \begin{array}{l} father, mother, parent :: \\ \quad \left\{ \begin{array}{l} father(x,y) \leftarrow parent(x,y) \wedge male(x) \\ mother(x,y) \leftarrow parent(x,y) \wedge female(x) \\ parent(x,y) \leftarrow father(x,y) \\ parent(x,y) \leftarrow mother(x,y) \end{array} \right\} \end{array} \right\}$$

*This new definition is positive recursive. This has the unintended effect that in each model, father, mother and parent are interpreted as the empty relationships.*

**Example 2** *A theory may contain more than one definition for the same concept. E.g.*

$$\left\{ \begin{array}{l} even :: \{ \; even(0) \leftarrow \neg odd(x) \; \} \\ even :: \left\{ \begin{array}{l} even(0) \\ even(s(s(x))) \leftarrow even(x) \end{array} \right\} \end{array} \right\}$$

*The first definition defines even as the complement of odd. In itself, it does not define even nor odd. The second definition defines even by the usual positive recursion. The combination of both definitions defines both even and odd.*

**Definition 3** *A definition is recursive iff a defined predicate appears in the body of a rule. A definition is positive recursive iff all occurrences of the defined predicates in the body of the rules are positive (i.e. occur in the scope of an even number of negations). A simultaneous definition defines more than one predicate. A stratified definition is one in which the defined predicates can be semi-ordered such that each defined predicate occurring positively, resp. negatively, in the body of a rule is less, resp. strictly less, than the predicate in the head.*

*A definition hierarchy is a set $\mathcal{D}$ of definitions such that each predicate is defined in at most one definition of $\mathcal{D}$ and $\mathcal{D}$ can be ordered such that each open predicate appearing in a definition is not defined in a later definition.*

Below, I define the concept of a well-founded definition. This concept generalizes the principle of definition in a well-founded set.

**Definition 4** *A definition $D$ is well-founded in some collection $\mathcal{I}$ of total interpretations of the open predicates of $D$ iff for each $I \in \mathcal{I}$, there exists a well-founded order on the atoms of $At_I$ such that for each ground instance $p(\overline{t}[\overline{d}]) \leftarrow F[\overline{d}]$, the body $F[\overline{d}]$ has the same truth value in all partial interpretations that extend $I$ and are identical on all atoms less than $p(|\overline{t}[\overline{d}]|^I)$.*

The following theorem states an interesting property of well-founded definitions.

**Theorem 1** *If $D$ is well-founded in $\mathcal{I}$, then each justified interpretation $M$ of $D$ extending an element $I$ of $\mathcal{I}$ is total (and hence a model) and coincides with the least model of the 3-valued completion of $D$ (Fitting 1985) extending $I$. $M$ is the unique model of the Clark completion (Clark 1978) of $D$ extending $I$.*

**Example 3** *Consider the definition of even numbers:*

$$even :: \left\{ \begin{array}{l} even(0) \leftarrow \\ even(S(x)) \leftarrow \neg even(x) \end{array} \right\}$$

*In the context of the natural numbers, this definition is well-founded and the justified interpretation is total. However, in any interpretation where the successor function contains cycles, the justified interpretation is partial.*

## Properties of definitions

### Consistency of definitions

The aim of an inductive definition is to *define* its defined predicates. Therefore, a natural quality requirement is that those justified interpretations that are total in the open predicates, should define truth of all defined predicates, i.e. they should be total in all predicates. As shown by Example 3 the property of having total justified interpretations is context dependent.

**Definition 5** *A definition $\mathcal{D}$ is well-defining in a collection $\mathcal{I}$ of total interpretations of its open predicates iff each justified interpretation of $\mathcal{D}$ extending an element of $\mathcal{I}$ is total. Otherwise, $\mathcal{D}$ is called an unfounded definition in $\mathcal{I}$.*

Part of the knowledge representation methodology for representing definitions is to show that each definition in the theory is well-defining in the collection of relevant interpretations of its open predicates. For this purpose, practical mathematical techniques must be developed.

**Theorem 2** *Assume that a theory $T$ can be split up in a sequence of theories $T_1, .., T_n$ such that for each $i$, the predicates with a definition in $T_i$ do not appear in $T_1, .., T_{i-1}$ and for each model $I$ of $T_1 \cup .. \cup T_{i-1}$, the definitions in $T_i$ are well-defining in $I$.*

*Then each justified interpretation of $T$, total for the subset of predicates without definition in $T$, is total.*

The proof of this theorem is omitted.

Some syntactic properties that guarantee that a definition is well-defining in every context are well-known from the logic programming literature:

- non-recursive definitions
- positive recursive definitions
- stratified definitions

Other properties guarantee well-defining definitions in some specific context. Inductive definitions corresponding to acyclic (Apt & Bezem 1990) or locally stratified logic programs (Przymusinski 1988) are well-defining in the context of Herbrand interpretations. It follows from theorem 1 that a well-founded definition in context $\mathcal{I}$ is also well-defining in $\mathcal{I}$.

A syntactical criterion that guarantees well-foundedness and hence well-defining-ness is the following. Define a relativized definition w.r.t. some strict order $<$ as a definition of a predicate $p(x, \overline{y})$ that consists of rules:

$$p(x, \overline{t}) \leftarrow F[x]$$

such that each $p$-atom in $F$ is of the form $p(z, \overline{t'})$ and appears in the scope of a subformula of $F[x]$ of the form $\forall z.z < x \rightarrow G$ or $\exists z.z < x \wedge G$.

When $<$ represents a well-founded order, relativized definitions are well-founded and hence well-defining, by theorem 1. The following theorem was proven.

**Theorem 3** *A relativized definition (w.r.t. to $<$) is well-founded in each interpretation that interprets $<$ as a strict well-founded order.*

### Equivalence of definitions

In a logic for knowledge representation, there should be a well-understood notion of equivalence. The following example shows that one cannot simply replace bodies of cases by equivalent bodies (w.r.t. 2-valued semantics).

**Example 4** *The definitions $p :: \{ \; p \leftarrow \mathbf{t} \; \}$ and $p :: \{ \; p \leftarrow p \vee \neg p \; \}$ have different justified interpretations, respectively the interpretations (represented as literal sets) $\{p\}$ and $\{\}$. Note that their bodies are equivalent w.r.t. 2-valued semantics but not w.r.t. 3-valued semantics.*

Some important cases of equivalence preserving rules are sketched below:

- A case $p(\overline{t}[\overline{x}]) \leftarrow F$ can be replaced by $p(\overline{y}) \leftarrow \exists \overline{x}.\overline{y} = \overline{t}[\overline{x}] \wedge F$.
- In a definition, two cases $p(\overline{t}) \leftarrow F_1$ and $p(\overline{t}) \leftarrow F_2$ can be replaced by one case $p(\overline{t}) \leftarrow F_1 \vee F_2$. Together with the first rule, it follows that replacing a set of rules by its Clark completion is equivalence preserving.
- The substitution of a sub-formula $F[\overline{x}]$ in the body of a case of a formula by a formula $G[\overline{x}]$ is equivalence preserving if $F[\overline{x}]$ and $G[\overline{x}]$ are equivalent in 3-valued logic, i.e. if $\forall \overline{x}.F[\overline{x}] \leftrightarrow G[\overline{x}]$ is a tautology in 3-valued logic[6].
- Define the composition of two definitions $Pred_1 :: \{ \; C_1 \; \}$ and $Pred_2 :: \{ \; C_2 \; \}$ as the definition $Pred_1 \cup Pred_2 :: \{ \; C_1 \cup C_2 \; \}$. In general, substituting a pair of definitions by their composition is not equivalence preserving. (Verbaeten, Denecker, & De Schreye 2000) presents an extensive study of when merging definitions is equivalence preserving in the context of open logic programming, a sub-formalism of the logic defined here. One important example is that a definition hierarchy (Definition 3) is equivalent with its composition. Note that the composition of a definition hierarchy of positive recursive definitions is a stratified definition.

[6] Here the strong Kleene truth table for $\leftrightarrow$ must be used.

## Monotonicity, non-monotonicity and modularity

Non-monotonicity is a necessary property of elaboration tolerant logic descriptions (McCarthy 1998). Non-monotonicity is a natural consequence of any sort of closure principle. However, also a degree of monotonicity is important in knowledge representation. Indeed, a highly desirable feature of a knowledge representation logic is that independent properties of the problem domain can be represented in a modular way, and that adding the modules together in one theory preserves the semantics of each module. Modular composition is guaranteed if the models of the composition of the modules are models of the independent modules. However, this property guarantees also that the extension of one module with another is a monotonic operation.

Both properties are present in the logic defined here:

- Extending a definition with one or more new cases is in general a non-monotonic operation.
- Adding new definitions or new axioms to a theory is a monotonic operation. This follows trivially from the definition of model.

## Applications of definitions

Below some applications of ID-logic are given.

**Terminological knowledge.**

According to (Brachman & Levesque 1982), definitions of terminology constitutes an important part of expert knowledge. Terminological knowledge is about the *defining properties* of a concept, i.e. the necessary and sufficient conditions to belong to the concept. To see the crucial difference between terminological and assertional knowledge, consider the atomic statement $c(O)$ that some object $O$ belongs to a concept $c$. When $c(O)$ represents an assertional statement, it asserts that $O$ belongs to $c$ and, hence, satisfies the defining property of $c$. On the other hand, if the atom is added as a new *case* to the definition of $c$, then the defining property of $c$ is modified such that $O$ belongs to $c$ by definition and not by virtue of its properties.

**Temporal Reasoning.**

In (Ternovskaia 1998), it was shown that Reiter's situation calculus (Reiter 1991) has an equivalent formalization by a set of positive recursive definitions of the fluent predicates and of the effects of actions. Using general inductive definitions (with positive and negative induction), the formalization can be further simplified in ID-logic. Below, I sketch how to do this.

The definition defines all fluent symbols and all causal predicates by simultaneous induction on the poset of situations. I introduce for each fluent $f$ three new predicates: $initially_f$ to represent the initial state of $f$, and $cause_f$ and $cause_{\neg f}$, representing initiating and terminating causes for $f$. For each fluent symbol $f$, the definition contains three cases[7]:

$$f(\overline{x}, S_0) \leftarrow initially_f(\overline{x})$$
$$f(\overline{x}, do(a,s)) \leftarrow cause_f(a,s,\overline{x})$$
$$f(\overline{x}, do(a,s)) \leftarrow f(\overline{x},s) \wedge \neg cause_{\neg f}(a,s,\overline{x})$$

Note that in contrast to Reiter's situation calculus, this rule set does not contain a rule of the form:

$$\neg f(\overline{x}, do(a,s)) \leftarrow cause_{\neg f}(a,s,\overline{x})$$

However, it is easy to show that the completion of the above 3 rules entails the formula:

$$\neg f(do(a,s)) \leftarrow \neg cause_f(a,s,\overline{x}) \wedge cause_{\neg f}(a,s,\overline{x})$$

which reduces to the causal rule for $\neg f$ if the natural requirement is added that an action cannot cause $f$ and $\neg f$ in the same situation. This requirement is formalised by the clause:

$$\leftarrow cause_{\neg f}(a,s,\overline{x}), cause_f(a,s,\overline{x})$$

This illustrates a general methodological principle of using inductive definitions. In an inductive definition, one defines a concept by enumerating the positive cases; given such an enumeration, the closure mechanism of the semantics yields the negative cases.

In addition, per initiating effect of some action represented by an action term $A[\overline{y}]$, there is a case:

$$cause_f(A[\overline{y}], s, \overline{x}) \leftarrow \Psi[\overline{y}, s, \overline{x}]$$

such that the only term in $\Psi$ of the situation sort is $s$ and it appears purely in fluent symbols. Likewise, for each terminating effect there is a case:

$$cause_{\neg f}(A[\overline{y}], s, \overline{x}) \leftarrow \Psi[\overline{y}, s, \overline{x}]$$

**Theorem 4** *A definition consisting of the above rules is well-founded in the collection of all interpretations that satisfy the Unique Names Axioms (UNA) axioms (Reiter 1980) and the second order induction axiom for the situation sort*[8].

It follows from theorem 1 that the semantics of this inductive definition coincides with its Clark completion. Note that the completion of this definition is very similar to Reiter's state successor axioms [9].

[7] We assume a many-sorted version of ID-logic, with situation, action and user defined sorts.

[8] The order of the atoms, needed to establish the well-foundedness of the definition is the order generated by the atoms $f(..,do(a,s)) > cause_f(a,s,..), cause_{\neg f}(a,s,..) > g(..,s)$, with $f$, $g$ arbitrary fluents.

[9] The main difference is that it lacks the action precondition predicate $poss(a,s)$. Action preconditions can be added to the inductive definition, by extending the definition with cases defining $poss$, by adding the atom $poss(a,s)$ as a conjunct to the second and third case defining $f$ and adding a fourth rule

$$f(\overline{x}, do(a,s)) \leftarrow f_o(\overline{x}, do(a,s)) \wedge \neg poss(a,s)$$

where $f_o$ is a new open predicate.

The inductive definition representation of situation calculus in ID-logic represents initiating and terminating effects in a case-by-case way. This results in a modular, elaboration tolerant representation of the domain in the sense that one can easily add new cases or drop or refine existing ones. This definition can be further extended with definitions for defined fluents, e.g. the definition of the transitive closure of physical connections in a computer network, in the context in which these physical connections may change:

$$\begin{array}{rcl} connected(c1,c2,s) & \leftarrow & physical\_connection(c1,c2,s) \\ connected(c1,c2,s) & \leftarrow & connected(c1,c3,s) \wedge \\ & & connected(c3,c2,s) \end{array}$$

Also, similarly as in (Ternovskaia 1998), ramification rules can be added to this theory.

**Inductive definitions as an approach to Causality.**

In (Denecker, Theseider Dupré, & Van Belleghem 1998) we argued that inductive definitions are a suitable formalization of causality. Causality information is an example of constructive information. Effects and forces propagate in a dynamic system through a constructive process in the following sense:

- there are no deus ex machina effects. Each effect has a cause; it is caused by a nonempty combination of actions and other effects.
- The causation order among effects is a well-ordering. I.e. there is no pair of effects each of which have caused the other; stronger, there is no infinite descending chain of effects each of which has been caused by the previous one in the chain.

The construction process of an inductive definition formally mimics this physical process of the propagation of the causes and effects. Based on this idea, (Denecker, Theseider Dupré, & Van Belleghem 1998) proposes a general solution to model ramifications. One point of (Denecker, Theseider Dupré, & Van Belleghem 1998) was that effects may easily depend on both presence and absence of other effects. For example, in the case that one latch of a suitcase is open, the effect of opening the second latch produces a derived effect of opening the suitcase, but only if the first latch is not closed simultaneously. As a consequence, if fluents mutually can influence each other, descriptions of ramifications may easily contain positive and negative loops. As shown in (Denecker, Theseider Dupré, & Van Belleghem 1998), the well-founded semantics deals well with these loops.

**Induction axioms; Domain Closure Axiom (DCA).**

As a conservative extension of classical logic, ID-logic assumes uncertainty on the domain of discourse (due to the non-Herbrand interpretations). The Domain Closure Axiom (DCA) expresses that the domain of discourse contains only named objects. In (McCarthy 1980), McCarthy showed how the DCA can be represented by a combination of circumscription on a set of rules and a FOL assertion. The mapping to ID-logic is straightforward. The set of rules is an inductive definition of a new predicate $U$; it consists of one case per constant $C$ and per functor $f$:

$$U :: \left\{ \begin{array}{l} U(C) \leftarrow \\ .. \\ U(f(\overline{x})) \leftarrow U(x_1), .., U(x_n) \end{array} \right\}$$

This defines $U$ as the set of all named objects. The FOL axiom expresses that all objects in the domain are named[10]:

$$\forall x.U(x)$$

The DCA is a generalized induction axiom. In the case of the language of the natural numbers (0 and $S/1$), the above formalization of the DCA is equivalent with Peano's second order induction axiom. The induction axiom for situations as needed in Reiter's situation calculus can be expressed in a similar way.

The semantics of many logics, e.g. logic programming and deductive databases, is based on Herbrand interpretations. This introduces the implicit ontological constraint that all terms in the domain of discourse are named. This constraint is absent in classical logic and in ID-logic but can be explicitly formalized by the pair of the DCA and the Unique Names Axioms (UNA) (Reiter 1980) or the Clark Equality Theory (CET) (Clark 1978). It is easy to show that each model of DCA+UNA is isomorphic with a Herbrand interpretation.

**Tables.**

The simplest way of defining a concept is by exhaustive enumeration of its elements. A table, as in the context of databases, can naturally be viewed as a definition by exhaustive enumeration. Tables are commonly used to define concepts, not only in databases but also in common sense knowledge representation, e.g. to define some scenario.

## Relationships to other logics.

ID-logic shows tight relationships with a class of different logics in different areas of AI, computer science and mathematical logic. Earlier in this paper, strong relations to circumscription and Clark completion came to light. Despite differences in the syntactic sugar, there are strong relationships between description logics and ID-logic. ID-logic fits into the schema of description logics, with a Tbox consisting of the definitions and an Abox consisting of classical logic axioms. The correspondence between description logics and a sub-logic of ID-logic were formally investigated in (Van Belleghem, Denecker, & De Schreye 1997).

[10]Note here the distinction between defining knowledge and assertional knowledge. If one would add the FOL assertion as a case to the inductive definition, then $U$ would be defined to be the complete domain of discourse.

Also fixpoint logic can be embedded in ID-logic, modulo syntactic sugar. The difference in syntactic sugar is that in ID-logic, the defined concepts are named by global predicate symbols, whereas in fixpoint logic, the defined concepts are represented by an operator form[11]. Inflationary fixpoint logic is an extension of fixpoint logic for fixpoint forms with negation in it. However, as mentioned in the introduction, this logic doesn't give the intended semantics to inductive definitions with negation. For example, the concept of even numbers is represented in ID-logic by:

$$even :: \{\ even(x) \leftarrow x = 0 \vee \exists y.x = S(y) \wedge \neg even(y)\ \}$$

The corresponding inflationary fixpoint form

$$\Gamma^{x}_{\Psi}(x = 0 \vee \exists y.x = S(y) \wedge \neg\Psi(y))$$

denotes the set of all natural numbers.

Logic Programming can be embedded in ID-logic in a straightforward way. Some of its extensions can be embedded as well. Abductive logic programming (Kakas, Kowalski, & Toni 1993) (or open logic programming, as it is called in (Denecker 1995)) can be embedded also in ID-logic. An abductive logic framework is a triple $< A, P, T >$ of a set $A$ of abducible predicates, a set $P$ of rules defining non-abducible predicates and a set $T$ of FOL axioms, called *constraints*. Its embedding in ID-logic is trivial: it is the theory $T \cup \{D_p\}$ where $D_P$ is a definition with $Rules(D) = P$ and with $Defined(D)$ the set of non-abducible predicates. In this embedding, the semantics of an abductive logic program is given by general non-Herbrand well-founded models.

The formalism of deductive databases (Abiteboul, Hull, & Vianu 1995) descends from logic programming. A deductive database consists of a triple (EDB,IDB,IC) where EDB (extensional database) consists of tables for extensional predicates, IDB (intensional database) consists of a logic program defining intensional predicates in terms of intensional and extensional predicates, and IC (integrity constraints) is a set of FOL axioms. The embedding of a deductive database in ID-logic consists of a set of definitions (by exhaustive enumeration) for the extensional predicates, a simultaneous definition for the intensional predicates, the set IC as FOL axioms and finally, the DCA+UNA.

## ID-logic compared to NMR

As argued, definitional knowledge is an important component of human expert knowledge. Such knowledge consists of definitions of terminology but is not restricted to that: e.g. induction axioms, knowledge of physical causation, etc.. ID-logic is specifically designed for representing such knowledge.

With respect to other applications such as the representation of defaults, the scope and applicability of inductive definitions is more restricted than other non-monotonic principles such as circumscription and default logic. The use of definitions for such forms of common sense knowledge requires a more rigorous methodology of a priori analysis and restructuring of the knowledge on the problem domain.

As mentioned earlier, Reiter (Reiter 1996) and Amati et al (Amati, Carlucci Aiello, & Pirri 1997) observe that *compilation* of non-monotonic theories into first order definitions (in Beth's style) can produce theories that clarify the meaning of the original theories and are computationally more attractive. Using the more general notion of inductive definitions in ID-logic, this compilation of non-monotonic theories to definitions becomes easier and can be performed for larger classes of NMR theories. Moreover, due to the non-monotonicity of the logic, the compiled representation will often maintain most of the elaboration tolerance of the original NMR representation.

With respect to computational efficiency, reasoning on ID-logic is undecidable in general, like in the case of other predicate non-monotonic logics. However, there is plenty of evidence that in many cases, definitions can be reasoned on more efficiently that other non-monotonic formalisms or even classical logic (e.g. constructing a well-founded model of a propositional definition is polynomial, while constructing a model of a propositional theory is NP-complete). A common aspect of virtually all logics related to ID-logic, is the strong focus on efficient implementation. Techniques from these areas can be used to implement efficient solvers.

[11] E.g. the transitive closure of a predicate $p$ is represented by the form:

$$\Gamma^{(x,y)}_{\Psi}(p(x,y) \vee (\exists z.\Psi(x,z) \wedge \Psi(z,y)))$$

It corresponds to the ID-logic definition of a binary predicate $tr$ defined by:

$$tr :: \{\ tr(x,y) \leftarrow p(x,y) \vee tr(x,z) \wedge tr(z,y)\ \}$$


## Acknowledgements

This work has benefitted from discussions with the following people: Maurice Bruynooghe, Danny Deschreye, Michael Gelfond, Vladimir Lifschitz, Victor Marek, Ray Reiter, Eugenia Ternovskaia, Mirek Truszczynski, Kristof Van Belleghem. Thanks!


## References


[Abiteboul, Hull, & Vianu 1995] Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addusin-Wesley Publishing Company.

[Aczel 1977] Aczel, P. 1977. An Introduction to Inductive Definitions. In Barwise, J., ed., *Handbook of Mathematical Logic*. North-Holland Publishing Company. 739–782.

[Amati, Carlucci Aiello, & Pirri 1997] Amati, G.; Carlucci Aiello, L.; and Pirri, F. 1997. Definability and commonsense reasoning. *Artificial Intelligence Journal* 93:1 – 30. Abstract of this paper appeared also in

Third Symposium on Logical Formalization of Commonsense Reasoning, Stanford, USA, 96.

[Apt & Bezem 1990] Apt, K., and Bezem, M. 1990. Acyclic programs. In *Proc. of the International Conference on Logic Programming*, 579–597. MIT press.

[Brachman & Levesque 1982] Brachman, R. J., and Levesque, H. 1982. Competence in Knowledge Representation. In *Proc. of the National Conference on Artificial Intelligence*, 189–192.

[Buchholz, Feferman, & Sieg 1981] Buchholz, W.; Feferman, S.; and Sieg, W. P. W. 1981. *Iterated Inductive Definitions and Subsystems of Analysis: Recent Proof-Theoretical Studies*. Springer-Verlag, Lecture Notes in Mathematics 897.

[Clark 1978] Clark, K. 1978. Negation as failure. In Gallaire, H., and Minker, J., eds., *Logic and Databases*. Plenum Press. 293–322.

[Denecker & De Schreye 1995] Denecker, M., and De Schreye, D. 1995. Representing Incomplete Knowledge in Abductive Logic Programming. *Journal of Logic and Computation* 5(5):553–578.

[Denecker, Theseider Dupré, & Van Belleghem 1998] 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/.

[Denecker 1995] Denecker, M. 1995. A Terminological Interpretation of (Abductive) Logic Programming. In Marek, V.; Nerode, A.; and Truszczynski, M., eds., *International Conference on Logic Programming and Nonmonotonic Reasoning*, Lecture notes in Artificial Intelligence 928, 15–29. Springer.

[Denecker 1998] Denecker, M. 1998. The well-founded semantics is the principle of inductive definition. In Dix, J.; nas del Cerro, L. F.; and Furbach, U., eds., *Logics in Artificial Intelligence*, 1–16. Schloss Daghstull: Springer-Verlag, Lecture notes in Artificial Intelligence 1489.

[Feferman 1970] Feferman, S. 1970. Formal theories for transfinite iterations of generalised inductive definitions and some subsystems of analysis. In Kino, A.; Myhill, J.; and Vesley, R., eds., *Intuitionism and Proof theory*. North Holland. 303–326.

[Fitting 1985] Fitting, M. 1985. A Kripke-Kleene Semantics for Logic Programs. *Journal of Logic Programming* 2(4):295–312.

[Gelder 1993] Gelder, A. V. 1993. The Alternating Fixpoint of Logic Programs with Negation. *Journal of computer and system sciences* 47:185–221.

[Gelfond & Lifschitz 1988] Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of the International Joint Conference and Symposium on Logic Programming*, 1070–1080. IEEE.

[Gurevich & Shelah 1986] Gurevich, Y., and Shelah, S. 1986. Fixed-point Extensions of First-Order Logic. *Annals of Pure and Applied Logic* 32:265–280.

[Kakas, Kowalski, & Toni 1993] Kakas, A. C.; Kowalski, R.; and Toni, F. 1993. Abductive Logic Programming. *Journal of Logic and Computation* 2(6):719–770.

[McCarthy & Hayes 1969] McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh University Press. 463–502.

[McCarthy 1980] McCarthy, J. 1980. Circumscription - a form of nonmonotonic reasoning. *Artifical Intelligence* 13:27–39.

[McCarthy 1998] McCarthy, J. 1998. Elaboration tolerance. In *COMMON SENSE 98, Symposium On Logical Formalizations Of Commonsense Reasoning*.

[Moschovakis 1974] Moschovakis, Y. N. 1974. *Elementary Induction on Abstract Structures*. North-Holland Publishing Company, Amsterdam- New York.

[Przymusinski 1988] Przymusinski, T. 1988. On the semantics of Stratified Databases. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufman.

[Przymusinski 1990a] Przymusinski, T. 1990a. Extended Stable Semantics for Normal and Disjunctive Programs. In Warren, D., and Szeredi, P., eds., *Proc. of the seventh international conference on logic programming*, 459–477. MIT press.

[Przymusinski 1990b] Przymusinski, T. 1990b. Well founded semantics coincides with three valued Stable Models. *Fundamenta Informaticae* 13:445–463.

[Reichgelt 1991] Reichgelt, H. 1991. *Knowledge Representation: an AI Perspecitive*. Ablex Publishing Corporation.

[Reiter 1980] Reiter, R. 1980. Equality and domain closure in first-order databases. *JACM* 27:235–249.

[Reiter 1991] Reiter, R. 1991. The Frame Problem in the Situation Calculus: A simple Solution (Sometimes) and a Completeness Result for Goal Regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honour of John McCarthy*, 359–380. Academic Press.

[Reiter 1996] Reiter, R. 1996. Nonmonotonic Reasoning: Compiled vs Interpreted Theories. Distributed at the conference of Nonmonotonic Reasoning NMR96 as considerations for the panel discussion.

[Ternovskaia 1998] Ternovskaia, E. 1998. Inductive Definability and the Situation Calculus. In Freitag, B.; Decker, H.; Kifer, M.; and Voronkov, A., eds., *Transactions and Change in Logic Databases*, volume 1472 of *LNCS*. Berlin: Springer-Verlag.

[Van Belleghem, Denecker, & De Schreye 1997] Van Belleghem, K.; Denecker, M.; and De Schreye, D. 1997. A strong correspondence between description logics and open logic programming. In Naish, L.,

ed., *Proc. of the International Conference on Logic Programming, 1997*, 346–360. MIT-press.

[Van Gelder, Ross, & Schlipf 1991] Van Gelder, A.; Ross, K.; and Schlipf, J. 1991. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM* 38(3):620–650.

[Verbaeten, Denecker, & De Schreye 2000] Verbaeten, S.; Denecker, M.; and De Schreye, D. 2000. Compositionality of normal open logic programs. *Journal of Logic Programming* 41(3):151–183.