TLP **10** (4–6): 581–596, 2010. © Cambridge University Press 2010
doi:10.1017/S1471068410000293

FO(FD): Extending classical logic with rule-based fixpoint definitions

PING HOU, BROES DE CAT and MARC DENECKER
Department of Computer Science, K.U. Leuven, Belgium
(*e-mail:* {ping.hou, broes.decat, marc.denecker}@cs.kuleuven.be)

submitted 7 February 2010; revised 10 April 2010; accepted 20 May 2010

Abstract

We introduce fixpoint definitions, a rule-based reformulation of fixpoint constructs. The logic FO(FD), an extension of classical logic with fixpoint definitions, is defined. We illustrate the relation between FO(FD) and FO(ID), which is developed as an integration of two knowledge representation paradigms. The satisfiability problem for FO(FD) is investigated by first reducing FO(FD) to difference logic and then using solvers for difference logic. These reductions are evaluated in the computation of models for FO(FD) theories representing fairness conditions and we provide potential applications of FO(FD).

KEYWORDS: fixpoint constructs, fixpoint definitions, classical logic extensions

1 Introduction

Two mainstream knowledge representation paradigms of the moment are on the one hand, classical logic-based approaches such as description logics (Baader *et al.* 2003), and on the other hand, rule-based approaches from logic programming and extensions such as Answer Set Programming and Abductive Logic Programming (Baral 2003; Kakas *et al.* 1992). The latter disciplines are rooted in the discipline of Non-Monotonic Reasoning (McCarthy 1986). FO(ID) (Denecker and Ternovska 2008) integrates both paradigms in a tight, conceptually clean manner. The key to integrate "rules" into classical logic (FO) is the observation that natural language, or more precisely, the informal language of mathematicians, has an informal rule-based construct: the construct of *inductive/recursive definitions* (IDs). FO(ID) extends FO not only with an inductive definition construct but also with an expressive and precise non-monotonic reasoning principle. It is an extension of FO with inductive definitions and an integration of FO and LP. It integrates monotonic and non-monotonic logics. The inductive definition construct of FO(ID) formally generalizes Datalog (Abiteboul *et al.* 1995). FO(ID) is also strongly related to fixpoint logics. Monotone definitions in FO(ID) are a different rule-based syntactic sugar of the fixpoint formulas of Least Fixpoint Logic (LFP) (Park 1969). Last but not least, FO(ID), being a clear, well-founded integration of rules into classical logic, might play a unifying role in the current attempts of extending FO-based description logics with rules (Vennekens and Denecker 2009). It thus appears that FO(ID)

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

occupies quite a central position in the spectrum of computational and knowledge representation logics.

The work in this paper is inspired by work on FO(ID) to integrate LP-style rules into fixpoint constructs. The resulting constructs are called *fixpoint definitions* (FDs). Fixpoint definitions use the rule-based format which will enable us to more easily link fixpoint constructs with the rule-based knowledge representation paradigm and the FO(ID) formalism. We define the logic FO(FD), which is an extension of classical logic with fixpoint definitions. In FO(FD), almost all kinds of inductions can be expressed in a natural way. The study of FO(FD) contributes to the understanding of rule-based systems and thus, to the study of the relation between non-monotonic inductive definitions and fixpoint definitions, to the study of the correspondence between well-founded and fixpoint semantics and to the integration of classical logic-based and rule-based approaches for knowledge representation.

We investigate the connection between FO(FD) and FO(ID) by presenting equivalence preserving transformations from FO(ID) to FO(FD). It turns out that all kinds of inductive definitions in FO(ID) can be expressed in FO(FD). Meanwhile, due to the allowance of the nesting of least and greatest fixpoint constructs in FO(FD), the nesting of induction and coinduction can be represented in FO(FD). Thus, some concepts, e.g., infinite structures and the nesting of recursion and corecursion (Barwise and Moss 1996), which can not be defined in FO(ID) in a well-founded way, can be handled naturally in FO(FD). We show that in general, FO(FD) is strictly more expressive than FO(ID).

On the computational level, the satisfiability problem for FO(FD), deciding the satisfiability of FO(FD) theories, is a major research topic. One research direction is towards developing solvers for extensions of propositional logic, e.g., SMT. Difference logic (Nieuwenhuis and Oliveras 2005) can be seen as an instance of an SMT framework where propositional logic is extended with simple linear constraints. Efficient implementation techniques for difference logic are emerging in the SMT domain (Nieuwenhuis and Oliveras 2005; Cotton and Maler 2006), which makes it a good choice as base technology. In this paper, we develop translations from FO(FD) to difference logic, based on similar reductions of logic programs presented in (Niemelä 2008; Janhunen *et al.* 2009). The translations reduce the satisfiability check of FO(FD) theories to finding satisfying interpretations of difference logic theories. This provides a novel approach to model expansion for FO(FD). We also present experimental results.

The paper is organized as follows. In Section 2, we introduce fixpoint definitions and the logic FO(FD). FO(ID) and the relationship between FO(FD) and FO(ID) are presented in Section 3. We investigate the satisfiability problem for FO(FD) by providing the reductions from FO(FD) to difference logic in Section 4. The reductions are evaluated experimentally in Section 5. In Section 6, we present some potential applications of FO(FD) and a conclusion follows in Section 7.

2 FO(FD): A logic of fixpoint definitions

In this section, we extend first-order logic (FO) with an alternative rule-based fixpoint construct: the construct of *fixpoint definitions* (FDs), to formalize a new

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

logic FO(FD), which can be viewed as an extension of first-order logic with mixed induction and coinduction.

2.1 Syntax

We assume familiarity with classical logic. A vocabulary Σ consists of a set of predicate and function symbols. Terms and FO formulae are defined as usual, and are built inductively from variables, constant and function symbols, logical connectives ($\neg$, $\wedge$, $\vee$) and quantifiers ($\forall$, $\exists$). Note that predicate symbols occurring in a fixpoint definition are viewed as predicate constants but not predicate variables.

A *rule* over a vocabulary Σ is an expression of the form $\forall \bar{x}(P(\bar{x}) \leftarrow \varphi[\bar{x}])$, where P is a predicate symbol of Σ and $\varphi[\bar{x}]$ is an arbitrary first-order formula over Σ. Atomic formula $P(\bar{x})$ is known as the *head* of the rule and $\varphi[\bar{x}]$ is known as the *body* of the rule. The *defined predicate* of the rule is P. The connective $\leftarrow$ is called *definitional implication* and is to be distinguished from material implication $\supset$, an abbreviation for $\neg body \vee head$. We say that a predicate symbol occurs positively (negatively) in a formula if it occurs in the scope of an even (odd) number of negations. A rule is *positive* in a set of predicate symbols if these symbols occur only positively in φ.

For a set $\mathscr{R}$ of rules, we denote $def(\mathscr{R})$ as the set of defined predicates of its rules, and we denote $open(\mathscr{R})$ as the set of all other symbols occurring in $\mathscr{R}$.

Without loss of generality, we assume from now on, that rule sets contain for each of its defined predicates exactly one rule of the form $\forall \bar{x}(P(\bar{x}) \leftarrow \varphi_P[\bar{x}])$. Indeed, any set of rules $\{\forall \bar{x}(P(\bar{x}) \leftarrow \varphi_1[\bar{x}]), \ldots, \forall \bar{x}(P(\bar{x}) \leftarrow \varphi_n[\bar{x}])\}$ can be transformed into a single rule $\forall \bar{x}(P(\bar{x}) \leftarrow \varphi_1[\bar{x}] \vee \ldots \vee \varphi_n[\bar{x}])$.

Definition 2.1
We define a *least fixpoint definition* (LFD), respectively *greatest fixpoint definition* (GFD) over vocabulary Σ by simultaneous induction, as a finite expression $\mathscr{D}$ of the form

$$\lfloor\ \mathscr{R}, \Delta_1, \ldots, \Delta_m, \nabla_1, \ldots, \nabla_n\ \rfloor, \text{ respectively } \lceil\ \mathscr{R}, \Delta_1, \ldots, \Delta_m, \nabla_1, \ldots, \nabla_n\ \rceil$$

with $0 \leqslant n, m$ such that:

1. $\mathscr{R}$ is a set of rules over Σ.
2. Each Δ_i is a least fixpoint definition and each ∇_j is a greatest fixpoint definition.

To express the remaining conditions, we need some auxiliary concepts and notations. For such an expression $\mathscr{D}$, we say that a rule r is *locally defined* in $\mathscr{D}$ if $r \in \mathscr{R}$, and that a predicate P is *locally defined* in $\mathscr{D}$ if $P \in def(\mathscr{R})$, and that P is *defined* in $\mathscr{D}$ if P is locally defined in $\mathscr{D}$ or defined in any of its subdefinitions $\Delta_1, \ldots, \nabla_n$. The set of defined predicates of $\mathscr{D}$ is denoted $def(\mathscr{D})$. A symbol is *open* in $\mathscr{D}$ if it occurs in $\mathscr{D}$ and is not defined in it. The set of open symbols of $\mathscr{D}$ is denoted $open(\mathscr{D})$.

3. Every defined symbol of $\mathscr{D}$ has only positive occurrences in the bodies of rules in $\mathscr{D}$.
4. Each symbol $P \in def(\mathscr{D})$ has exactly one local definition in $\mathscr{D}$. Formally, $\{def(\mathscr{R}), def(\Delta_1), \ldots, def(\nabla_n)\}$ is a partition of $def(\mathscr{D})$.

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

5. For every subdefinition $\mathscr{D}'$ of $\mathscr{D}$, $open(\mathscr{D}') \subseteq open(\mathscr{D}) \cup def(\mathscr{R})$. In particular, a symbol defined in another subdefinition $\mathscr{D}'' \neq \mathscr{D}'$, does not occur in $\mathscr{D}'$.

A *fixpoint definition* is either a least fixpoint definition or a greatest fixpoint definition. We allow arbitrary nesting of least and greatest fixpoint definitions.

An *FO(FD) formula* is either an FO formula or a fixpoint definition. An *FO(FD) theory* is a set of fixpoint definitions and FO sentences.

Example 2.2
Assume a binary predicate T denoting a transition graph on a set of vertices, representing the states. Assume a property on states R, i.e., a unary predicate on vertices. The set of states P that have an (infinite) path passing an infinite number of times through a state satisfying R, is defined by:

$$\left[\begin{array}{l} \forall x\ (P(x) \leftarrow Q(x)) \\ \left[\begin{array}{l} \forall x\ (Q(x) \leftarrow R(x) \wedge \exists y(T(x,y) \wedge P(y))) \\ \forall x\ (Q(x) \leftarrow \exists y(T(x,y) \wedge Q(y))) \end{array}\right] \end{array}\right]$$

2.2 Semantics

The semantics of FO(FD) is an integration of standard FO semantics with fixpoint semantics of definitions. We start by defining the fixpoint semantics.

Given two disjoint first-order vocabularies Σ and Σ', a Σ-interpretation I and a Σ'-interpretation I', the $\Sigma \cup \Sigma'$-interpretation mapping each element e of Σ to e^I and each $e \in \Sigma'$ to $e^{I'}$ is denoted by $I + I'$. When $\Sigma' \subseteq \Sigma$, we denote the restriction of a Σ-interpretation I to the symbols of Σ' by $I|_{\Sigma'}$. For a Σ-interpretation I and a tuple of domain elements $\overline{d}$, we denote by $I[\overline{x}/\overline{d}]$ the interpretation that has the same domain as I, interprets $\overline{x} = (x_1, \ldots, x_n)$ by $\overline{d} = (d_1, \ldots, d_n)$, and coincides with I on all other symbols.

With a set $\mathscr{R}$ of rules over Σ and a (partial) two-valued Σ-interpretation I interpreting at least all open symbols and no defined symbols, i.e., $\Sigma \cap def(\mathscr{R}) = \emptyset$ and $open(\mathscr{R}) \subseteq \Sigma$, there is a standard way of associating an operator $\Gamma_I^{\mathscr{R}}$ on the set of $def(\mathscr{R})$-interpretations with the domain of I. For two such interpretations J, K, we define $\Gamma_I^{\mathscr{R}}(J) = K$ if for every $\forall \overline{x}(P(\overline{x}) \leftarrow \varphi_P[\overline{x}]) \in \mathscr{R}$, $P^K = \{\overline{d} | (I+J)[\overline{x}/\overline{d}] \models \varphi_P[\overline{x}]\}$.

If each defined symbol in $def(\mathscr{R})$ has only positive occurrences in the body of a rule in $\mathscr{R}$, the operator $\Gamma_I^{\mathscr{R}}$ is monotone with respect to the standard truth order on interpretations and hence, it has least and greatest fixpoints in this set denoted $lfp(\Gamma_I^{\mathscr{R}})$, respectively $gfp(\Gamma_I^{\mathscr{R}})$. Importantly, if $P^I \leqslant P^{I'}$ for every symbol $P \in open(\mathscr{R})$ with only positive occurrences in rule bodies of $\mathscr{R}$, then $lfp(\Gamma_I^{\mathscr{R}}) \leqslant lfp(\Gamma_{I'}^{\mathscr{R}})$ and $gfp(\Gamma_I^{\mathscr{R}}) \leqslant gfp(\Gamma_{I'}^{\mathscr{R}})$.

Given an expression $\mathscr{D}$ which might be an LFD or a GFD, and an $open(\mathscr{D})$-interpretation I interpreting at least all open symbols of $\mathscr{D}$ and no defined ones. We define an operator $\Gamma_I^{\mathscr{D}}$ on the set of $def(\mathscr{D})$-interpretations with domain $dom(I)$. This operator is monotone with respect to the standard truth order on interpretations and hence, it has least and greatest fixpoints in this set. We define $\Gamma_I^{\mathscr{D}}(J)$ inductively as the interpretation $K + K'$ where

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

- K is the $(def(\mathscr{D}) \setminus def(\mathscr{R}))$-interpretation such that, for $J' = I + J|_{def(\mathscr{R})}$:
  - $K|_{def(\Delta_i)} = lfp(\Gamma^{\Delta_i}_{J'})$ for all $i = 1, \ldots, m$.
  - $K|_{def(\nabla_j)} = gfp(\Gamma^{\nabla_j}_{J'})$ for all $j = 1, \ldots, n$.

  Observe that J' interprets all open symbols in every subdefinition of $\mathscr{D}$.
- K' is the $def(\mathscr{R})$-interpretation $\Gamma^{\mathscr{R}}_{I+K}(J|_{def(\mathscr{R})})$.

Definition 2.3 (*Model of* $\mathscr{D}$)
Let $\mathscr{D}$ be a fixpoint definition and I a two-valued Σ-interpretation such that Σ contains all symbols in $\mathscr{D}$. If $\mathscr{D}$ is an LFD, then I *satisfies* $\mathscr{D}$, or I is a *model* of $\mathscr{D}$, iff $I|_{def(\mathscr{D})} = lfp(\Gamma^{\mathscr{D}}_{I|_{open(\mathscr{D})}})$. If $\mathscr{D}$ is a GFD, then I *satisfies* $\mathscr{D}$, or I is a *model* of $\mathscr{D}$, iff $I|_{def(\mathscr{D})} = gfp(\Gamma^{\mathscr{D}}_{I|_{open(\mathscr{D})}})$. As usual, this is denoted $I \models \mathscr{D}$.

Example 2.4 (*Continued 2.2*)
Semantically, the fixpoint definition in Example 2.2 has the following meaning: the relationship P is the result of iteratively computing a least (for P) and a greatest fixpoint (for Q). In the n-th iteration of the outer fixpoint, P will contain a vertex iff it has a (finite) path that goes through at least n times through vertices with property R. At fixpoint, P (and Q) will contain a vertex iff it has a path that infinitely often reaches a vertex with property R.

Definition 2.5 (*Model of an FO(FD) theory*)
Let T be an FO(FD) theory over Σ and I a two-valued Σ-interpretation. Then I is a *model* of T, denoted by $I \models T$, iff $I \models \varphi$ for every $\varphi \in T$.

Definition 2.6 (*Equivalence*)
A theory T_1 with vocabulary Σ_1 is equivalent to a theory T_2 with vocabulary Σ_2 iff each model M_1 of T_1 restricted to Σ_2 can be extended to a model M_2 of T_2 and vice versa.

2.3 PC(FD)

In this section, we introduce PC(FD), the propositional fragment of FO(FD). We assume familiarity with propositional logic.

A propositional vocabulary Σ is a set of propositional atoms. A literal is an atom p or its negation $\neg p$. An atom p is called a *positive* literal, $\neg p$ a negative one. For a literal l, we identify $\neg\neg l$ with l.

A propositional fixpoint definition is a fixpoint definition such that all symbols occurring in it are propositional symbols.

Example 2.7
Consider the propositional fixpoint definition

$$\mathscr{D} = \left[\begin{array}{l} p \leftarrow q \vee r \\ q \leftarrow p \\ \left[\begin{array}{l} r \leftarrow p \\ s \leftarrow t \vee a \\ t \leftarrow s \end{array} \right] \end{array} \right]$$

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

It is obvious that a is the only open atom in this fixpoint definition. There are only two interpretations satisfying $\mathscr{D}$, namely, $I_1 = \{a \mapsto \mathbf{f}, p \mapsto \mathbf{f}, q \mapsto \mathbf{f}, r \mapsto \mathbf{f}, s \mapsto \mathbf{t}, t \mapsto \mathbf{t}\}$ and $I_2 = \{a \mapsto \mathbf{t}, p \mapsto \mathbf{f}, q \mapsto \mathbf{f}, r \mapsto \mathbf{f}, s \mapsto \mathbf{t}, t \mapsto \mathbf{t}\}$. The construction of I_1 is illustrated as follows: $I_1^1 = \{a \mapsto \mathbf{f}, p \mapsto \mathbf{f}, q \mapsto \mathbf{f}, r \mapsto \mathbf{t}, s \mapsto \mathbf{t}, t \mapsto \mathbf{t}\}$ and, because the body of the only rule for r is false, $I_1^2 = \{a \mapsto \mathbf{f}, p \mapsto \mathbf{f}, q \mapsto \mathbf{f}, r \mapsto \mathbf{f}, s \mapsto \mathbf{t}, t \mapsto \mathbf{t}\}$, which is the limit of the iterations and thus, $I_1 = I_1^2$.

A propositional fixpoint definition $\mathscr{D}$ is in *definitional normal form* (DefNF) if for any $p \in \Sigma$, the fixpoint definition contains at most one rule $p \leftarrow \varphi_p$, and either $\varphi_p = \bigvee B_p$ or $\varphi_p = \bigwedge B_p$, where B_p is a set of literals called the *body literals*. Any propositional fixpoint definition can be transformed into DefNF in polynomial time using Tseitin transformation (Tseitin 1968). Hence without loss of generality, we can from now on assume that propositional fixpoint definitions are in DefNF.

A *PC(FD) theory* is a set of propositional formulas and propositional fixpoint definitions. An interpretation I satisfies a PC(FD) theory if it satisfies every formula and every definition of the theory.

3 A comparison of FO(FD) and FO(ID)

FO(ID) is an extension of first-order logic with a new construct, namely *generalized inductive definitions*, for representing definitions that occur often in mathematics, but in general cannot be expressed in first-order logic. It was originally introduced in Denecker (2000), and further developed in Denecker and Ternovska (2008). In this section, we compare FO(FD) to FO(ID) by providing transformations from generalized inductive definitions to alternating fixpoint definitions and showing that in general, the FO(FD) formalism is strictly more expressive than the FO(ID).

Definition 3.1
Let Σ be a vocabulary. A *(generalized) inductive definition (GID) D* over Σ is a finite set of rules over Σ. Its sets of defined symbols $def(D)$, respectively open symbols $open(D)$ are defined as usual.

We do not insist on defined predicates to occur positively in rule bodies in a generalized inductive definition, but allow non-monotone inductive definitions.

An *FO(ID) formula* is a Boolean combination of FO formulas and generalized inductive definitions. An *FO(ID) theory* is a set of generalized inductive definitions and FO sentences. A model of a generalized inductive definition is a two-valued well-founded model (Denecker and Ternovska 2008). The semantics of FO(ID) is an integration of standard two-valued FO semantics with the well-founded semantics of generalized inductive definitions.

Example 3.2
Consider the following non-monotone inductive definition of even and odd numbers over the structure of the natural numbers with zero and the successor function:

$$\left\{ \begin{array}{l} \forall x(Even(x) \leftarrow x = 0 \vee \exists y(x = s(y) \wedge \neg Even(y))) \\ \forall x(Odd(x) \leftarrow \exists y(x = s(y) \wedge Even(y))) \end{array} \right\}$$

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

We begin our comparison of FO(FD) and FO(ID) by presenting equivalence preserving transformations from generalized inductive definitions to alternating fixpoint definitions. New symbols may be introduced to the original vocabulary Σ.

Definition 3.3
Let D be a generalized inductive definition. For each defined predicate P of D, we introduce a new predicate symbol $P^{\neg}$ of the same arity of P. For each formula φ, let $\overline{\varphi}$ denote the formula obtained by substituting each negative occurrence $P(\bar{t})$ of a defined predicate P in φ by $\neg P^{\neg}(\bar{t})$. We define two sets of rules: $\mathscr{R}_D = \{\forall \bar{x}(P(\bar{x}) \leftarrow \overline{\varphi_P[\bar{x}]}) \mid P \in def(D)\}$ and $\mathscr{R}_D^{\neg} = \{\forall \bar{x}(P^{\neg}(\bar{x}) \leftarrow \overline{\neg\varphi_P[\bar{x}]}) \mid P \in def(D)\}$. Now define Δ_D as $\lfloor\ \mathscr{R}_D, \lceil\ \mathscr{R}_D^{\neg}\ \rceil\ \rfloor$.

Let D be a generalized inductive definition over Σ. Then Δ_D is a least fixpoint definition over $\Sigma' = \Sigma \cup \{P^{\neg} \mid P \in def(D)\}$. Note that $open(D) = open(\Delta_D)$.

Example 3.4 (*Continued 3.2*)
Translating the previous FO(ID) formula into FO(FD) leads to

$$\left\lfloor \begin{array}{l} \forall x(Even(x) \leftarrow x = 0 \vee \exists y(x = s(y) \wedge Even^{\neg}(y))) \\ \forall x(Odd(x) \leftarrow \exists y(x = s(y) \wedge Even(y))) \\ \left\lceil \begin{array}{l} \forall x(Even^{\neg}(x) \leftarrow x \neq 0 \wedge \forall y(x = s(y) \supset Even(y))) \\ \forall x(Odd^{\neg}(x) \leftarrow \forall y(x = s(y) \supset Even^{\neg}(y))) \end{array} \right\rceil \end{array} \right\rfloor$$

Theorem 3.5
Let D be a generalized inductive definition over Σ. Then there exists a one-to-one mapping between the Σ-models I of D and the Σ'-models I' of Δ_D such that the domain of I is the same as that of I', $I'|_{\Sigma} = I$ and $(P^{\neg})^{I'}$ is the (relative) complement of P^I for each $P \in def(D)$.

In the following we show that in general, FO(FD) and FO(ID) do not have the same expressive power.

Theorem 4.4 in Schlipf (1995), for the well-founded semantics, states that a relation is definable in the well-founded semantics iff it is inductively (Π_1^1) definable over the natural numbers. However, on the other hand, Theorem 10 in Bradfield (1996) presents that the FO(FD) alternation hierarchy, the hierarchy of alternating LFD and GFD expressions (ordered along the number of alternations) in any fixpoint definitions, is strict. A consequence is the following result.

Corollary 3.6
FO(ID) is strictly less expressive than FO(FD) on infinite structures.

4 Satisfiability of FO(FD)

The second part of this paper presents an approach to *finite model expansion* for FO(FD), the inference task consisting of, given a theory T, generating a model for the theory. As a declarative problem solving technique, model generation for FO(FD) allow to represent e.g. temporal properties in an application, increasing its general applicability to among others program verification.

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

Finite model expansion is equivalent to checking the satisfiability of a Boolean formula, the *satisfiability* problem, solved by *SAT solvers*. One approach to check the satisfiability of FO theories, taken by many state-of-the-art solvers, is by reducing the theory to propositional logic (a transformation called *grounding*) and using a SAT solver afterwards. Grounding generally consists of replacing all variables in a formula by all possible substitutions, but intelligent techniques exist that greatly reduce the size of such a grounding, see e.g. Wittocx *et al.* (2008).

Satisfiability checking of FO(FD) theories can be done in a similar way. First the FO(FD) theory is grounded to a PC(FD) theory. Afterwards, the PC(FD) theory is reduced to *difference logic* (Nieuwenhuis and Oliveras 2005), propositional logic extended with linear constraints, and a difference logic solver is used to check the satisfiability of the resulting theory. In the domain of SMT, efficient difference logic solvers have been developed, see e.g. Cotton and Maler (2006).

Difference logic, denoted PC(DL), is the extension of propositional logic with linear difference constraints of the form $x + c < y$, where x, y and c are integer variables, of which c is known. Syntactically, a linear constraint can occur in the same positions as an atom. An interpretation of a difference logic theory assigns truth values to atoms and integer values to variables.

We first introduce the grounding of FO(FD) to a variable free form. Then, we address the reductions of PC(FD) to difference logic.

Without loss of generality, we only consider theories in function free FO(FD) for the rest of the paper (any FO(FD) theory can be transformed into a function free theory in polynomial time).

4.1 Grounding FO(FD)

The reduction of an FO(FD) theory T to a PC(FD) theory is defined by:

Definition 4.1
Given an FO(FD) theory T and a finite domain D. To allow grounding of quantified formulas, we introduce a new constant c_d for each domain element $d \in \mathrm{D}$, which maps to d in every interpretation I. The grounding of T according to domain D, denoted $G(T)$, consists of all $G(\varphi)$ where $\varphi \in T$ and φ is either an FO sentence or a fixpoint definition, and $G(\varphi)$ is defined as:

$$G(\varphi) = \begin{cases} \bigwedge_{d \in \mathrm{D}} G(\psi[x/c_d]) & \text{if } \varphi := \forall x\ \psi[x] \\ \bigvee_{d \in \mathrm{D}} G(\psi[x/c_d]) & \text{if } \varphi := \exists x\ \psi[x] \\ G(\psi_1) \wedge G(\psi_2) & \text{if } \varphi := \psi_1 \wedge \psi_2 \\ G(\psi_1) \vee G(\psi_2) & \text{if } \varphi := \psi_1 \vee \psi_2 \\ \left\lfloor\ G(\psi)\ \right\rfloor & \text{if } \varphi := \left\lfloor\ \psi\ \right\rfloor \\ \neg G(\psi) & \text{if } \varphi := \neg\psi \\ \left\lceil\ G(\psi)\ \right\rceil & \text{if } \varphi := \left\lceil\ \psi\ \right\rceil \\ p \leftarrow G(\psi) & \text{if } \varphi := p \leftarrow \psi \text{ and } p \text{ is an atom} \\ \psi & \text{if } \psi \text{ is an atom} \end{cases}$$

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

Proposition 4.2
An interpretation I is a model of an FO(FD) theory T iff it is a model of $G(T)$.

4.2 Reduction to difference logic

The aim is to reduce a PC(FD) theory $G(T)$ to an *equivalent* theory $DL(T)$ in difference logic. The reduction of FO sentences to a PC(DL) theory coincides with their grounding, so for each FO sentence $\varphi \in T$, $DL(T)$ contains a sentence $G(\varphi)$. The reduction of fixpoint definitions consists of the *completion* and *level mapping* constraints.

4.2.1 Completion

The *completion*, introduced by Clark (1978) for logical rules, expresses in FO the consistency between the truth value of the head and the body of a rule.

The *completion* of a propositional rule $r = p \leftarrow \varphi_p$, denoted $Comp(r)$, is given by the formula $p \equiv \varphi_p$. The *completion* of a propositional fixpoint definition $\mathscr{D}$, denoted by $Comp(\mathscr{D})$, is $\bigcup_{r \in \mathscr{D}} Comp(r)$.

An important property is that $I \models \mathscr{D}$ implies $I \models Comp(\mathscr{D})$. The converse is not true, $\mathscr{D}$ generally has fewer models than $Comp(\mathscr{D})$.

Example 4.3
Consider the propositional fixpoint definition

$$\mathscr{D} = \left\{ \begin{array}{l} p \leftarrow p \vee a \\ \left\lceil \; q \leftarrow q \wedge p \; \right\rceil \end{array} \right\}$$

Then $Comp(\mathscr{D}) = (p \equiv p \vee a) \wedge (q \equiv q \wedge p)$. $\mathscr{D}$ has two models: $\{a \mapsto \mathbf{f}, p \mapsto \mathbf{f}, q \mapsto \mathbf{f}\}$ and $\{a \mapsto \mathbf{t}, p \mapsto \mathbf{t}, q \mapsto \mathbf{t}\}$; $Comp(\mathscr{D})$ has the same two models, and the additional three models: $\{a \mapsto \mathbf{f}, p \mapsto \mathbf{t}, q \mapsto \mathbf{t}\}$, $\{a \mapsto \mathbf{f}, p \mapsto \mathbf{t}, q \mapsto \mathbf{f}\}$ and $\{a \mapsto \mathbf{t}, p \mapsto \mathbf{t}, q \mapsto \mathbf{f}\}$.

4.2.2 Level mappings

To obtain equivalence of T and $DL(T)$, it is necessary to ensure that only interpretations consistent with the operator $\Gamma_I^{\mathscr{D}}$ are models of $DL(T)$. We take a *level mapping* approach to characterize the models of the fixpoint operator. This is an extension of the technique presented in (Niemelä 2008; Janhunen *et al.* 2009), where stable model generation of logic programs is obtained by reduction to difference logic.

Definition 4.4 (*level mapping*)
Given a fixpoint definition $\mathscr{D}$, define a function $l_{\mathscr{D}} : def(\mathscr{D}) \rightarrow \mathbb{N}$, with $def(\mathscr{D})$ the set of all defined atoms in $\mathscr{D}$. Function l is then the *level mapping* function and $l_{\mathscr{D}}(p)$ is the *level* of defined atom p for fixpoint definition $\mathscr{D}$.

A level mapping function $l_{\mathscr{D}}$ is introduced for each (nested) fixpoint definition $\mathscr{D}$ in $G(T)$. In ground form, for each fixpoint definition $\mathscr{D}$ and for each defined atom p in $\mathscr{D}$, we introduce an integer variable, denoted $l_{\mathscr{D}}^p$.

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

The level mapping should ensure that the truth of a least fixpoint relation or the falsity of a greatest fixpoint relation can always be *finitely justified* in terms of locally defined atoms or open ones.

4.2.3 Level mapping constraints

We introduce PC(DL) formulas which, as part of $DL(T)$, act as constraints on the relation between the levels of different defined atoms within one fixpoint definition. Theory $DL(T)$ will be satisfiable iff such a finite justification exists.

As mentioned earlier, all rules are considered to be in DefNF. For a given rule r in fixpoint definition $\mathscr{D}$, h denotes the head and $body(r)$ is the set of all literals occurring in the body of r. The sets $BL_{def}(\mathscr{D},r)$ and $BL_{open}(\mathscr{D},r)$ denote the set of defined, respectively open body literals(BL)

$$BL_{def}(\mathscr{D},r) = \{d | d \in def(\mathscr{D}) \cup \neg def(\mathscr{D}) \text{ and } d \in body(r)\} \tag{1}$$

$$BL_{open}(\mathscr{D},r) = \{o | o \in open(\mathscr{D}) \cup \neg open(\mathscr{D}) \text{ and } o \in body(r)\} \tag{2}$$

We now introduce the constraints.

No justification is necessary for an atom defined in a *GFD* if it is true, nor for an atom defined in an *LFD* which is false. Formally represented by the constraints:

$$\text{if } \mathscr{D} \text{ is a GFD:} \quad a \supset l^h_{\mathscr{D}} = 0 \tag{3}$$

$$\text{if } \mathscr{D} \text{ is an LFD:} \quad \neg a \supset l^h_{\mathscr{D}} = 0 \tag{4}$$

When an atom defined in a *GFD* is not true or an atom defined in an *LFD* is not false, a *justification* is necessary. A justification is a set of body literals of a rule sufficient to derive the head in a given interpretation. Although looping is allowed over literals defined in lower fixpoints, it has to be possible to construct a justification which does not loop over literals in the same level.

Deriving that the head of a rule with a disjunctive body in an LFD is true requires only one body atom to be true. If it were a rule with a conjunctive body, all body literals would be necessary as justification. This also holds for the relation between their levels: in the disjunctive rule, the minimal level of all true body literals can act as the level of the justification. In the conjunctive case, the level is the maximum level of all body literals.

These ideas can be generalized and formalized as constraints. For clarity, the constraints are not in PC(DL), but we introduce $min\{\}$ and $max\{\}$ notation to represent respectively the minimum and maximum of a set of levels. Assume an interpretation I to further simplify the aggregate notation. All aggregates can be translated out easily, independent of I (see further). Also assume a fixpoint definition $\mathscr{D}$ with a locally defined atom h in a rule r.

1. If $\mathscr{D}$ is an LFD and r has a conjunctive body, the translation of r is:

$$h \supset l^h_{\mathscr{D}} > max\{l^d_{\mathscr{D}} | d \in BL_{def}(\mathscr{D},r) \text{ and } I(d) = \mathbf{t}\} \tag{5}$$

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

2. If $\mathscr{D}$ is an LFD and r has a disjunctive body, the translation of r is:

$$h \supset (l^h_{\mathscr{D}} > min\{l^d_{\mathscr{D}} | d \in BL_{def}(\mathscr{D}, r) \text{ and } I(d) = \mathbf{t}\} \vee \bigvee_{d \in BL_{def}(\mathscr{D},r)} d \vee \bigvee_{o \in BL_{open}(\mathscr{D},r)} o) \tag{6}$$

3. If $\mathscr{D}$ is a GFD and r has a disjunctive body, the translation of r is:

$$\neg h \supset l^h_{\mathscr{D}} > max\{l^d_{\mathscr{D}} | d \in BL_{def}(\mathscr{D}, r) \text{ and } I(d) = \mathbf{f}\} \tag{7}$$

4. If $\mathscr{D}$ is a GFD and r has a conjunctive body, the translation of r is:

$$\neg h \supset (l^h_{\mathscr{D}} > min\{l^d_{\mathscr{D}} | d \in BL_{def}(\mathscr{D}, r) \text{ and } I(d) = \mathbf{f}\} \vee \bigvee_{d \in BL_{def}(\mathscr{D},r)} \neg d \vee \bigvee_{o \in BL_{open}(\mathscr{D},r)} \neg o) \tag{8}$$

Similar constraints apply for the level of the head h of rules defined in a subdefinition of $\mathscr{D}$, but the inequality $l^h_{\mathscr{D}} > \ldots$ is relaxed to $l^h_{\mathscr{D}} \geqslant \ldots$

Proposition 4.5
The truth value of a higher defined atom can only be justified by finite looping over literals in the same definition or infinite looping over literals in lower definitions. This is expressed by using similar constraints for locally defined rules and for rules defined in subdefinitions, but dropping the strict order requirement on the second, effectively allowing infinite looping over literals defined in subdefinitions.

Example 4.6
In the following fixpoint definition, using only strict ordering would lead to a contradiction, although a model exists.

$$\left\lfloor \begin{array}{l} a \leftarrow c \\ \left\lceil \begin{array}{l} c \leftarrow d \\ d \leftarrow c \end{array} \right\rceil \end{array} \right\rfloor$$

Theorem 4.7
If an FO(FD) theory is transformed using the presented reduction to PC(DL) via PC(FD), the resulting PC(DL) theory will be satisfiable iff the FO(FD) theory is satisfiable. Any model of the PC(DL) theory can be transformed into a model of the FO(FD) theory.

4.2.4 Aggregate reduction

To obtain PC(DL) constraints, the aggregates *min* and *max* have to be transformed into difference constraints, which can be done in the following fashion:

Replace	$l^h_{\mathscr{D}} > max(\{l^d_{\mathscr{D}} \mid d \in BL_{def}(\mathscr{D}, r) \text{ and } I(d) = \mathbf{t}\})$
by	$\bigwedge_{d \in BL_{def}(\mathscr{D},r)} (l^h_{\mathscr{D}} > l^d_{\mathscr{D}} \vee \neg d)$

Replace	$l^h_{\mathscr{D}} > min(\{l^d_{\mathscr{D}} \mid d \in BL_{def}(\mathscr{D}, r) \text{ and } I(d) = \mathbf{t}\})$
by	$\bigvee_{d \in BL_{def}(\mathscr{D},r)} (l^h_{\mathscr{D}} > l^d_{\mathscr{D}} \wedge d)$

For a condition $I(d) = \mathbf{f}$ instead of $I(d) = \mathbf{t}$, the literal d is replaced with $\neg d$.

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

4.2.5 Optimization: partial level mapping

Level mappings constraints are used to enforece dependencies between defined atoms. Often, a preprocessing step (before PC(DL) reduction) allows to deduce that certain atoms will never depend on each other. In that case, less mapping constraints are necessary. A simple example are non-cyclic dependencies, for which no level mapping constraints are necessary ($Comp(\mathscr{D}) \models \mathscr{D}$). These dependencies can be obtained by calculating the *strongly connected components* (Tarjan 1972) on the *dependency graph* of the fixpoint definition, a general technique used among others in stable model generation (Syrjänen and Niemelä 2001).

The *dependency graph* consists of all edges $h \rightsquigarrow b$, for each rule r in $\mathscr{D}$ with head h and for each body literal b of r that is defined in $\mathscr{D}$ or in a parent of $\mathscr{D}$. A *strongly connected component* of a directed graph is a maximal subset in which a path exists between any two nodes in the set.

Proposition 4.8
Only defined atoms that are in a strongly connected component with $\|nodes\| \geqslant 2$ or have recursion over themselves (e.g. $h \rightsquigarrow h$) need a level mapping. Body atoms that are not in the same strongly connected component as the head can be treated as open instead of defined atoms.

To implement this idea, the set of open body literals $BL_{open}(\mathscr{D}, r)$ is redefined: for a rule r, a body literal of r is considered open if it is not defined, defined in an ancestor of the definition of r or if it is not in the same strongly connected component as the head of r. The set $BL_{def}(\mathscr{D}, r)$ contains all remaining body literals.

4.2.6 Optimization: stronger constraints

The presented constraints are *weak*: infinitely many models of the PC(DL) reduction exist that are equivalent (modulo shared vocabulary) to one model of the FO(FD) theory. Exact one-to-one mapping is not possible because expressions of the form $x = c$, where c is a known integer constant, cannot be expressed in difference logic. By expressing all constraints in terms of one integer variable, which acts as a floating ground, we can greatly reduce the number of redundant models.

The presented constraints can be adapted to obtain such *stronger* constraints by enforcing that the level of the head of a rule is the minimum allowed by its associated constraint, adapted from in (Niemelä 2008; Janhunen *et al.* 2009). For example for a rule with a conjunctive body in a least fixpoint, which is subject to the constraint expressed by equation 5, a second constraint is added of the form:

$$h \supset \bigvee_{d \in BL_{def}(\mathscr{D}, r)} (l^h_{\mathscr{D}} = l^d_{\mathscr{D}} + 1 \wedge d) \tag{9}$$

5 Implementation and experiments

In this section, we report our first experiments, on model checking of fairness conditions, with a prototype implementation of the reductions from FO(FD) to

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

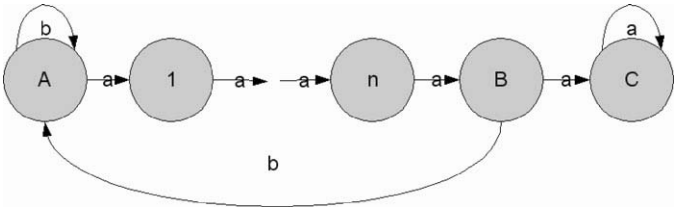


Fig. 1. A transition graph

Table 1. *Model checking results.*

‖*nodes*‖	weak(sec)	strong(sec)
503	0.011	0.004
1503	0.21	0.09
2503	20.51	14.19

difference logic. We used the μ-calculus fairness expression presented in (Liu *et al.* 1998):

$$\nu X.\mu Y.[-](\langle a\rangle X) \vee Y \tag{10}$$

It expresses that a state in the transition system is fair if on all possible paths, an a-labeled edge is infinitely often taken. Translated into an FO(FD) theory:

$$\left[\begin{array}{l} \forall x\ (P(x) \leftarrow Q(x)) \\ \lfloor\ \forall x\ (Q(x) \leftarrow \forall y\ (Edge(x,y) \supset (L(y,a) \wedge P(y)) \vee Q(y)))\ \rfloor \end{array}\right]$$

where the relations P and Q contain states from which infinitely often a state labelled a will be reached. The predicate L is the labelling relation, expressing that a state has a certain label. The predicate $Edge$ is the transition relation.

The task consists of doing model expansion, where the transitions and labellings are known, to decide which nodes are fair. Both weak and strong constraints were tested. The experiments were done on the graph depicted in Figure 1. The results of these experiments are as shown in Table 1, grounding times are included. The machine used is a dual-core 2.4 GHz with 4 Gb RAM, with Ubuntu 8.04 OS. Yices2 was used as difference logic solver.

From these preliminary results, we conclude that fairness conditions can be evaluated efficiently using our reduction to difference logic. Strong constraints are significantly faster due to their fewer degrees of freedom, which presumably allow more propagation and pruning of the search space. In Keinänen and Niemelä (2004), similar results were obtained with the same experiment.

6 Applications

Many applications can be found on the use of fixpoint expressions. Most of them focus on inductive and coinductive definitions (which have nesting depth 1), used e.g. for expressing transitive closure (reachability), bisimulation and situation calculus. One important application domain for nested fixpoint definitions is the verification

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

of automata. Temporal logics like CTL* allow to express time-variant properties of automata, e.g. fairness. The μ-calculus, a superset logic of those temporal logics bound on fixpoint expressions, can be transformed into fixpoint definitions. So any application of model checking or model generation of temporal logics can be expressed in FO(FD). Another application domain are so-called *parity games*, which are infinite games played on a graph with priority-annotated nodes. For more information we refer to Friedmann and Lange (2009). Parity games can be expressed in fixpoint logic, the nesting increasing polynomially with the number of priorities.

7 Conclusions and related work

In this paper, we have introduced fixpoint definitions, an alternative rule-based expression of fixpoint constructs, and the logic FO(FD), which is an extension of classical logic with fixpoint definitions. We have compared FO(FD) and FO(ID) by providing equivalence preserving transformations of non-monotone inductive definitions to alternating fixpoint definitions and showed that FO(FD) is strictly more expressive than FO(ID) on infinite structures. We have investigated the satisfiability problem for FO(FD) by developing reductions from FO(FD) to difference logic. Hence, SMT solvers supporting difference logic can be used for computing fixpoint models of FO(FD) theories *without any modifications*. We have implemented these reductions and evaluated the resulting solver in the computation of models of FO(FD) theories. In general, our transformation to difference logic is exponential in the nesting depth of a fixpoint definition, but for most practical applications they prove compact and efficient.

μMALL$^{=}$, which is the logic obtained by extending MALL (multiplicative, additive linear logic) with equality, quantification (via $\forall$ and $\exists$) and mixed least and greatest fixpoint constructors, was introduced in Baelde and Miller (2007). It seems that μMALL$^{=}$ has the same expressive power as FO(FD). However, μMALL$^{=}$ is developed from a proof theory standpoint whereas FO(FD) is developed from a model theory point of view.

Gupta *et al.* (2007) introduced coinduction, corresponding to the greatest fixpoint constructor, into logic programming to obtain coinductive logic programming. Discussed applications are verification, model checking, non-monotonic reasoning, etc. However, in coinductive logic programming, naively mixing coinduction and induction leads to contradictions while arbitrary cyclical nesting of least and greatest fixpoint constructs is allowed in FO(FD). Another difference is on the computational level. The main computational task for FO(FD) is model generation in the context of a finite domain. However, model generation in coinductive logic programming is applied to constructs of an infinite Herbrand model based on an infinite Herbrand universe.

Niemelä (2008) and Janhunen *et al.* (2009) introduced stable model generation of general logic programs via reductions to difference logic. They also used stable model generation to find solutions to Boolean equation systems (Keinänen and Niemelä 2004). This is a related fixpoint formalism, in which among others μ-calculus can be expressed.

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

There are several other solvers for solving the satisfiability and validity problems for fixpoint logics, e.g., Friedmann and Lange (2009). Our reduction is based on SMT solver technology, whereas referenced works are based on characterizations of satisfiability through infinite (cyclic) tableaux. Well-foundedness for unfoldings of least fixpoints is then checked using deterministic parity automata.

References

Abiteboul, S., Hull, R., and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., Eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Baelde, D. and Miller, D. 2007. Least and greatest fixed points in linear logic. In *LPAR*, N. Derschowitz and A. Voronkov, Eds. LNCS, vol. 4790. Springer, 92–106.

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

Barwise, J. and Moss, L. S., Eds. 1996. *Vicious Circles: On the mathematics of Non-Wellfounded Phenomena*. University of Chicago Press.

Bradfield, J. C. 1996. The modal mu-calculus alternation hierarchy is strict. In *CONCUR*, U. Montanari and V. Sassone, Eds. LNCS, vol. 1119. Springer, 233–246.

Clark, K. L. 1978. Negation as failure. In *Logic and Data Bases*. Plenum Press, 293–322.

Cotton, S. and Maler, O. 2006. Fast and flexible difference constraint propagation for DPLL(*T*). In *SAT*, A. Biere and C. P. Gomes, Eds. LNCS, vol. 4121. Springer, 170–183.

Denecker, M. 2000. Extending classical logic with inductive definitions. In *CL*, J. W. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, and P. J. Stuckey, Eds. LNCS, vol. 1861. Springer, 703–717.

Denecker, M. and Ternovska, E. 2008. A logic of nonmonotone inductive definitions. *ACM Transactions on Computational Logic (TOCL) 9,* 2, Article 14.

Friedmann, O. and Lange, M. 2009. Solving parity games in practice. In *ATVA*, Z. Liu and A. P. Ravn, Eds. LNCS, vol. 5799. Springer, 182–196.

Gupta, G., Bansal, A., Min, R., Simon, L., and Mallya, A. 2007. Coinductive logic programming and its applications. In *ICLP*, V. Dahl and I. Niemelä, Eds. LNCS, vol. 4670. Springer, 27–44.

Janhunen, T., Niemelä, I., and Sevalnev, M. 2009. Computing stable models via reductions to difference logic. In *LPNMR*, E. Erdem, F. Lin, and T. Schaub, Eds. LNCS, vol. 5753. Springer, 142–154.

Kakas, A. C., Kowalski, R. A., and Toni, F. 1992. Abductive logic programming. *Journal of Logic and Computation 2,* 6, 719–770.

Keinänen, M. and Niemelä, I. 2004. Solving alternating boolean equation systems in answer set programming. In *INAP/WLP*, D. Seipel, M. Hanus, U. Geske, and O. Bartenstein, Eds. LNCS, vol. 3392. Springer, 134–148.

Liu, X., Ramakrishnan, C. R., and Smolka, S. A. 1998. Fully local and efficient evaluation of alternating fixed points (extended abstract). In *TACAS*, B. Steffen, Ed. LNCS, vol. 1384. Springer, 5–19.

McCarthy, J. 1986. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence 28,* 1, 89–116.

Niemelä, I. 2008. Stable models and difference logic. *Annals of Mathematics and Artificial Intelligence 53,* 1–4, 313–329.

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press

Nieuwenhuis, R. and Oliveras, A. 2005. DPLL(T) with exhaustive theory propagation and its application to difference logic. In *CAV*, K. Etessami and S. K. Rajamani, Eds. LNCS, vol. 3576. Springer, 321–334.

Park, D. 1969. Fixpoint induction and proofs of program properties. *Machine Intelligence 5*, 59–78.

Schlipf, J. S. 1995. The expressive powers of the logic programming semantics. *Journal of Computer and System Sciences 51,* 1, 64–86.

Syrjänen, T. and Niemelä, I. 2001. The smodels system. In *LPNMR*, T. Eiter, W. Faber, and M. Truszczyński, Eds. LNCS, vol. 2173. Springer, 434–438.

Tarjan, R. E. 1972. Depth-first search and linear graph algorithms. *SIAM Journal on Computing 1,* 2, 146–160.

Tseitin, G. S. 1968. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic II*, A. O. Slisenko, Ed. Consultants Bureau, N.Y., 115–125.

Vennekens, J. and Denecker, M. 2009. FO(ID) as an extension of DL with rules. In *ESWC*, L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. P. B. Simperl, Eds. LNCS, vol. 5554. Springer, 384–398.

Wittocx, J., Mariën, M., and Denecker, M. 2008. Grounding with bounds. In *AAAI*, D. Fox and C. P. Gomes, Eds. AAAI Press, 572–577.

https://doi.org/10.1017/S1471068410000293 Published online by Cambridge University Press