Journal of Artificial Intelligence Research 45 (2012) 79-124 Submitted 03/12; published 09/12

An Approximative Inference Method for Solving ∃∀SO Satisfiability Problems

Hanne Vlaeminck HANNE.VLAEMINCK@CS.KULEUVEN.BE
Joost Vennekens JOOST.VENNEKENS@CS.KULEUVEN.BE
Marc Denecker MARC.DENECKER@CS.KULEUVEN.BE
Maurice Bruynooghe MAURICE.BRUYNOOGHE@CS.KULEUVEN.BE
Department of Computer Science,
Celestijnenlaan 200a
3001 Heverlee, Belgium

Abstract

This paper considers the fragment $\exists\forall SO$ of second-order logic. Many interesting problems, such as conformant planning, can be naturally expressed as *finite domain* satisfiability problems of this logic. Such satisfiability problems are computationally hard (Σ_2^P) and many of these problems are often solved approximately. In this paper, we develop a general approximative method, i.e., a sound but incomplete method, for solving $\exists\forall SO$ satisfiability problems. We use a syntactic representation of a constraint propagation method for first-order logic to transform such an $\exists\forall SO$ satisfiability problem to an $\exists SO(ID)$ satisfiability problem (second-order logic, extended with inductive definitions). The finite domain satisfiability problem for the latter language is in NP and can be handled by several existing solvers. Inductive definitions are a powerful knowledge representation tool, and this motivates us to also approximate $\exists\forall SO(ID)$ problems. In order to do this, we first show how to perform propagation on such inductive definitions. Next, we use this to approximate $\exists\forall SO(ID)$ satisfiability problems. All this provides a general theoretical framework for a number of approximative methods in the literature. Moreover, we also show how we can use this framework for solving practical useful problems, such as conformant planning, in an effective way.

1. Introduction

Finite model generation is a logical paradigm for solving constraint problems. A successful instance is the field of SAT, where efficient solvers for the low level CNF language are developed. Other instances, but for more expressive languages, are Answer Set Programming (ASP) (Baral, 2003) and model expansion (MX) for (extensions of) first order logic. In ASP, e.g., finite Herbrand models of an answer set program are computed (Baral, 2003). Model expansion (MX) (Mitchell & Ternovska, 2005) generalizes Herbrand model generation and aims at computing one or more models of a theory T that expand a finite interpretation I_0 for a (possibly empty) subset of symbols of T. MX for first order logic (MX(FO)) is formally equivalent to the finite domain satisfiability checking problem for existential second-order logic ($SAT(\exists SO)$)[1] which is known from Fagin's celebrated theorem to *capture NP* (Fagin, 1974). That is, the problems in NP are exactly those that are in a precise sense equivalent to an ∃SO satisfiability problem, and hence an MX(FO) problem. A range of solvers exists

1. Or more specifically, to the search problem for a *witness* of such a problem.

for finite model generation, e.g., an overview of state-of-the-art ASP and MX(FO($\cdot$)) solvers (here, FO($\cdot$) refers to the family of extensions of FO) can be found in the reports of the ASP competition (e.g., Denecker, Vennekens, Bond, Gebser, & Truszczyński, 2009).

Example 1.1. Here is a bounded planning problem modeled as a Finite Model Generation problem. The problem is deliberately kept simple as it will serve as the running example in this paper: *a glass may be clean or not, and can be cleaned by the action of wiping.* We can represent this dynamic domain by the following FO theory T_{act}:

$$\begin{aligned} \forall t : (Clean(t+1) &\Leftrightarrow Clean(t) \vee Wipe(t)). && (1)\\ Clean(0) &\Leftrightarrow InitiallyClean. && (2) \end{aligned}$$

The bounded planning problem we are considering is then to turn a dirty glass into a clean one in n steps, where $n \in \mathbb{N}$ is a given constant. This can indeed be formulated as a Model Expansion problem: find a model that satisfies T_{act} while $InitiallyClean$ is false, and $Clean(n)$ true. We can formulate this problem equivalently as an $\exists SO$ finite domain satisfiability problem, namely as the satisfiability problem in the range $[0 \ldots n]$ of time points of the following $\exists SO$ formula:

$$\exists Wipe, Clean, InitiallyClean : (\psi_{act} \wedge \neg InitiallyClean \wedge Clean(n)), \tag{3}$$

where with ψ_{act} we denote the conjunction of sentences in T_{act}. For $n > 0$, this formula is indeed satisfiable in the suitable interpretation of constants 0 and n and the binary function $+$, and, moreover, each *witness* W for its satisfiability provides a plan. For instance, wiping at time point 0 does the job, as is verified by the witness W for which $Wipe^W = \{0\}$ and $Clean^W = \{1, \ldots, n\}$.

While a large number of search problems can indeed be seen as Finite Model Generation problems, there are also a number of problems that are of a higher complexity than NP, and consequently cannot be formulated as an MX(FO) problem. Indeed, in this paper we are not interested in NP, but in the next level Σ_2^P of the polynomial hierarchy. Perhaps the prototypical such problem is finite domain satisfiability for $\exists\forall SO$: satisfaction in finite interpretations is in Σ_2^P for every $\exists\forall SO$ sentence and is Σ_2^P-hard for some such sentences (Immerman, 1998). An interesting Σ_2^P problem is that of *conformant planning*, which we discuss in detail in Section 7, but already introduce in the next example.

Example 1.2. Extending Example 1.1, suppose that we do not know whether the object is initially clean or dirty, but still want a plan that is guaranteed to make it clean, *no matter what* the initial situation was. This is no longer a standard planning problem, but is called a conformant planning problem. We can formulate this as the following $\exists\forall SO$ satisfiability problem:

$$\exists Wipe \, \forall InitiallyClean, Clean : \quad (\psi_{act} \Rightarrow Clean(n)). \tag{4}$$

In words, we need an assignment to the action $Wipe$ such that the goal $Clean(n)$ is satisfied for every initial situation $InitiallyClean$ and fluent $Clean$ that satisfy the action theory.

Solving problems like this would require us to make a choice for the existentially quantified predicates and then check that the implication is satisfied for every interpretation of the universally quantified predicates. While this can be done in principle, in practice it is often too expensive. In this paper, we explore an alternative based on a propagation method for first order logic developed by Wittocx, Denecker, and Bruynooghe (2010). This method computes in polynomial time, for a given theory T and a partial interpretation $\mathcal{I}$, an approximation of what has to be *certainly true* (= true in all models of T that expand $\mathcal{I}$) and *certainly false* (= false in all such models). Now, an interesting property of this propagation method is that it can be syntactically represented as a *monotone inductive definition* (Denecker & Ternovska, 2008) that defines (in an approximative way) these underestimates of the predicates in T and of their complements. Such a monotone inductive definition is essentially just a set of propagation rules, similar to a definite logic program, that is interpreted by the least fixpoint of its immediate consequence operator. For any given theory T we can obtain this approximating definition by a linear transformation of the original FO formula.

Returning to the above example, we need to find an interpretation for the action predicates, such that *for every* interpretation of the other predicates, the implication $\psi_{act} \Rightarrow Clean(n)$ is satisfied, i.e., such that without knowing anything about these other predicates, we should already be *certain* that the implication is satisfied. The basic idea behind our method is to approximate an $\exists\forall SO$ problem of the form $\exists\bar{P}\forall\bar{Q}\,\psi$, using the approximate definition from Wittocx et al. to check whether an interpretation for the existentially quantified predicates $\bar{P}$ has the property of making ψ true, regardless of the predicates $\bar{Q}$. Essentially, this reduces an $\exists\forall SO$ problem to an $\exists SO(ID)$ problem (where with SO(ID) we refer to SO extended with inductive definitions).

In Section 5, we extend our method to $\exists\forall SO(ID)$ problems. As argued by Denecker and Ternovska, inductive definitions are a useful tool for knowledge representation. For example, many dynamic domains can be formulated naturally and in a modular way using inductive definitions, while this can be quite tedious in FO. We already mentioned *conformant planning* as a typical $\exists\forall SO$ satisfiability problem. Typically, these conformant planning problems require the modeling of a dynamic domain. We will come back to the syntax and semantics of inductive definitions in the next section, but the dynamic domain of Example 1.1 can, as an alternative to the action theory T_{act}, be formulated as the following inductive definition Δ_{act}:

$$\left\{ \begin{array}{lcl} Clean(t+1) & \leftarrow & Clean(t). \\ Clean(t+1) & \leftarrow & Wipe(t). \\ Clean(0) & \leftarrow & InitiallyClean. \end{array} \right\} \tag{5}$$

The conformant planning problem can then be formulated alternatively as the satisfiability problem of the formula $\exists\, Wipe\, \forall InitiallyClean, Clean : (\Delta_{act} \Rightarrow Clean(n))$. However, this is no longer a $\exists\forall SO$ satisfiability problem, but a $\exists\forall SO(ID)$ satisfiability problem. This motivates us to see how we can extend our approximation method to such $\exists\forall SO(ID)$ satisfiability problems. For this purpose, we first show how to symbolically represent propagation on inductive definitions. Next, we show how we can use this together with the representation of propagation for FO to approximate finite domain $\exists\forall SO(ID)$ satisfiability problems.

Our approximation method has a number of benefits. First of all, it is a general method, that can be applied automatically to approximately solve any $\exists\forall SO$ problem. Second, the required computation can be carried out by any off-the-shelf MX(FO(ID)) solver, allowing our method to benefit effortlessly from improvements in solver technology, such as the IDP system by Mariën, Wittocx, and Denecker (2006). Finally, as we show in Section 7, our method elegantly generalizes a number of approximate reasoning methods from the literature (e.g., Baral, Gelfond, & Kosheleva, 1998; Son, Tu, Gelfond, & Morales, 2005; Denecker, Cortés Calabuig, Bruynooghe, & Arieli, 2010; Doherty, Magnusson, & Szalas, 2006; Son et al., 2005).

Parts of this work have already been presented at the JELIA 2011 conference (Vlaeminck, Wittocx, Vennekens, Denecker, & Bruynooghe, 2010).

2. Preliminaries

We assume familiarity with classical first-order logic (FO) and second-order logic (SO). In this section we introduce some of the notations and conventions used throughout this paper.

2.1 First-order Logic

A *vocabulary* Σ is a finite set of predicate symbols Σ^P and function symbols Σ^F, each with an associated arity. Constants are function symbols with arity 0. We often denote a symbol S with arity n by S/n. A $\Sigma-$*interpretation* I consists of a domain D and an assignment of a relation $P^I \subseteq D^n$ to each predicate symbol $P/n \in \Sigma$ and an assignment of a function $F^I : D^n \rightarrow D$ to each function symbol $F/n \in \Sigma$. We assume that Σ^P contains the equality predicate '=' interpreted as the identity relation. A *pre-interpretation* of Σ consists of a domain and an interpretation of the function symbols. If I is a Σ-interpretation and $\Sigma' \subseteq \Sigma$, we denote by $I|_{\Sigma'}$ the restriction of I to the symbols of Σ'. If Σ_1 and Σ_2 are two disjoint vocabularies, I a Σ_1-interpretation with domain D and J a Σ_2-interpretation with the same domain, then $I + J$ denotes the unique $(\Sigma_1 \cup \Sigma_2)$-interpretation with domain D such that $(I + J)|_{\Sigma_1} = I$ and $(I + J)|_{\Sigma_2} = J$.

Terms and formulas over a vocabulary Σ are defined as usual. An expression of the form $P(\bar{d})$ where P is an n-ary predicate and $\bar{d} \in D^n$ is called a *domain atom*. A *domain literal* is a domain atom $P(\bar{d})$ or the negation $\neg P(\bar{d})$ thereof. As usual, we denote a formula φ by $\varphi[\bar{x}]$ to indicate that the set of free variables of φ is a subset of $\bar{x}$. A formula without free variables is called a *sentence*. The satisfaction relation $\models$ is defined as usual. For an interpretation I, a formula φ with n free variables and a tuple of n domain elements $\bar{d}$, we use $I \models \varphi[\bar{d}]$ as a shorthand for $I, \theta[\bar{x} : \bar{d}] \models \varphi[\bar{x}]$, where θ is a variable assignment, and $\theta[\bar{x} : \bar{d}]$ is the variable assignment that is the same as θ except that it maps the variables $\bar{x}$ to the domain elements $\bar{d}$. We define the truth evaluation function $(\varphi[\bar{d}])^I$ as follows: $(\varphi[\bar{d}])^I = \mathbf{t}$ iff $I \models \varphi[\bar{d}]$ and $(\varphi[\bar{d}])^I = \mathbf{f}$ otherwise. We say that a formula φ is in *negation normal form* if φ contains no implications or equivalences, and all negations occur directly in front of atoms. We define the inverse on truth values as follows: $(\mathbf{f})^{-1} = \mathbf{t}$ and $(\mathbf{t})^{-1} = \mathbf{f}$. We also define the following strict order on the truth values: $\mathbf{f} <_t \mathbf{t}$. The truth order point-wise extends to interpretations: if I and J are two Σ-interpretations, then we say that $I \leq_t J$ if for every predicate symbol P and tuple of domain elements $\bar{d}$ it holds that $P^I(\bar{d}) \leq_t P^J(\bar{d})$.

Similar to how a real number r can be approximated by an interval $[l, u]$ such that $l \leq r \leq u$, in this paper we approximate Σ-interpretations K by a pair (I, J) of Σ-interpretations, such that $I \leq_t K \leq_t J$. We denote by $[I, J]$ the interval of all such interpretations K. This interval is empty if and only if $I \not\leq_t J$. It follows easily from well-known monotonicity results, that if we evaluate all positive occurrences (i.e., in the scope of an even number of negations) of atoms in some formula φ by I, and all negative occurrences (i.e., in the scope of an odd number of negations) by J, we are underestimating the truth of φ in the interval $[I, J]$. Conversely, if we evaluate positive occurrences in J and negative occurrences in I, we are overestimating the truth of φ in $[I, J]$. To state this property more formally, we introduce the following notation.

Definition 2.1 (Pos-neg evaluation relation φ^{+I-J})**.** Let φ be a Σ-formula and let I and J be Σ-interpretations. We define the *pos-neg evaluation* of φ in I and J, denoted by φ^{+I-J}, by induction over the size of φ:

- for an atom $\varphi = P(\bar{t})$, $\varphi^{+I-J} = \varphi^I$;
- for $\varphi = \neg\psi$, $\varphi^{+I-J} = (\psi^{+J-I})^{-1}$;
- for $\varphi = \psi_1 \wedge \psi_2$, $\varphi^{+I-J} = \mathbf{t}$ iff $\psi_i^{+I-J} = \mathbf{t}$ for both $i = 1, 2$;
- for $\varphi = \exists x\ \psi$, $\varphi^{+I-J} = \mathbf{t}$ iff there is a $d \in D$ such that $\psi^{+I[x/d]-J[x/d]} = \mathbf{t}$.

We now indeed have that, for each $K \in [I, J]$, $\varphi^{+I-J} \leq \varphi^K \leq \varphi^{+J-I}$. Also, we have that $\varphi^K = \varphi^{+K-K}$.

There is an intimate connection between the approximation of an interpretation by a pair of interpretations and Belnap's four-valued logic (1977). We denote the truth values *true, false, unknown* and *inconsistent* of four-valued logic by respectively $\mathbf{t}, \mathbf{f}, \mathbf{u}$ and $\mathbf{i}$. On these truth values, the *truth order* $\leq_t$ and *precision order* $\leq_p$ are defined as shown in Figure 1.

A four-valued relation of arity n on some domain D is a function from D^n to $\{\mathbf{t}, \mathbf{f}, \mathbf{u}, \mathbf{i}\}$. A four-valued interpretation $\mathcal{I}$ of vocabulary Σ consists of a pre-interpretation and of $P^{\mathcal{I}}$, a four-valued relation of arity n on D for each predicate symbol $P/n \in \Sigma$. Again, the precision order pointwise extends to interpretations: if $\mathcal{I}$ and $\mathcal{J}$ are two Σ-interpretations, then we say that $\mathcal{I} \leq_p \mathcal{J}$ if for every predicate symbol P and tuple of domain elements $\bar{d}$ it holds that $P^{\mathcal{I}}(\bar{d}) \leq_p P^{\mathcal{J}}(\bar{d})$. Similarly, also the truth order is extended to interpretations.

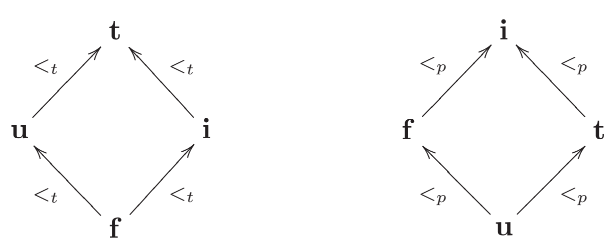

Figure 1: The truth and precision order

There is a natural isomorphism between Belnap's four truth values and pairs of the two standard truth values:

$$\begin{aligned}\tau(\mathbf{t},\mathbf{t}) &= \mathbf{t};\\ \tau(\mathbf{f},\mathbf{t}) &= \mathbf{u};\\ \tau(\mathbf{t},\mathbf{f}) &= \mathbf{i};\\ \tau(\mathbf{f},\mathbf{f}) &= \mathbf{f}.\end{aligned}$$

Intuitively, this mapping τ interprets its first argument as an underestimate to the "real" truth value, and its second argument as an overestimate: if the underestimate is $\mathbf{f}$ and the overestimate is $\mathbf{t}$, then the real truth value is indeed unknown; whereas, if the underestimate is $\mathbf{t}$ and the overestimate if $\mathbf{f}$, then there cannot exist a real truth value, since $\mathbf{t} \not\leq \mathbf{f}$, so we end up with inconsistency. This isomorphism τ extends in an obvious way to an isomorphism between pairs (I, J) of two-valued interpretations and four-valued interpretations $\mathcal{I}$ which all share the same pre-interpretations: (I, J) and $\mathcal{I}$ are isomorphic iff, for each predicate P/n and tuple $\bar{d} \in D^n$, $P^{\mathcal{I}}(\bar{d}) = \tau(P^I(\bar{d}), P^J(\bar{d}))$. We also denote this isomorphism by τ.

There is a tight link between the pos-neg evaluation function φ^{+I-J} and the Belnap evaluation $\varphi^{\mathcal{I}}$:

$$\varphi^{\mathcal{I}} = \tau(\varphi^{+I-J}, \varphi^{+J-I}), \text{ where } \tau(I, J) = \mathcal{I}.$$

When $\mathcal{I}$ is a three-valued structure (i.e., it never assigns $\mathbf{i}$) this corresponds to the standard Kleene evaluation (1952). In the rest of this paper, we will often omit the isomorphism τ, and, e.g., simply denote the four-valued truth value of a formula φ in a pair of interpretations (I, J) as $\varphi^{(I,J)}$. An important property, that we already stated above in different notation, is that $\varphi^{(I,J)} \leq_p \varphi^K$ for all $K \in [I, J]$.

There is a natural and well-known alternative way of using an interval $[I, J]$ for which $I \leq_t J$ to assign a truth value to a formula φ: the *supervaluation* (van Fraassen, 1966).

Definition 2.2 (Supervaluation $sv_{(I,J)}(.)$)**.** The supervaluation $sv_{(I,J)}(\varphi)$ of a sentence φ in a pair of interpretations (I, J) (or equivalently, a three-valued interpretation $\tau(I, J)$) is defined as

$$sv_{(I,J)}(\varphi) = glb_{\leq_p}(\{\varphi^K | K \in [I, J]\}).$$

It is easy to see that always $sv_{(I,J)}(\varphi) \geq_p \varphi^{(I,J)}$. This inequality may be strict. For instance, if we take $\varphi = Q \vee \neg Q$ and interpretations I and J such that $Q^{(I,J)} = \mathbf{u}$, then $sv_{(I,J)}(\varphi) = \mathbf{t}$, but $\varphi^{(I,J)} = \mathbf{u}$. The supervaluation has the following interesting property. Let I be an interpretation for the free vocabulary of a $\forall$SO formula $\varphi = \forall \bar{Q}\, \psi$, and let (J_1, J_2) be the least precise pair of interpretations for $\bar{Q}$ in the domain D of I (i.e., $Q^{(J_1,J_2)}(\bar{d}) = \mathbf{u}$ for all $Q/n \in \bar{Q}$ and $\bar{d} \in D^n$). We then have that $sv_{(I+J_1,I+J_2)}(\psi) = \mathbf{t}$ if and only if $\varphi^I = \mathbf{t}$.

Key to our approach is that we can simulate the four-valued truth evaluation in pairs of interpretations by encoding what is certainly true and certainly false, using a single two-valued structure $\mathcal{I}^{tf}$ over a new vocabulary Σ^{tf}. As we show in the next section, this gives us a convenient vocabulary to syntactically represent the construction of such an approximation. The new vocabulary Σ^{tf} contains the function symbols Σ^F of Σ and, for each predicate $P \in \Sigma^P$, two symbols P^{ct} and P^{cf}. The interpretations of P^{ct} and P^{cf} in $\mathcal{I}^{tf}$ contain, respectively, those tuples $\bar{d}$ for which $P^{\mathcal{I}}(\bar{d})$ is certainly true and those for which it is certainly false. Formally, for a vocabulary Σ and a four-valued Σ-interpretation

$\mathcal{I} = \tau(I, J)$, the Σ^{tf}-interpretation $\mathcal{I}^{tf}$ has the same pre-interpretation as $\mathcal{I}$, and is defined by:

$$(P^{ct})^{\mathcal{I}^{tf}} = \{\bar{d} | P^{\mathcal{I}}(\bar{d}) \geq_p \mathbf{t}\} = P^I,$$
$$(P^{cf})^{\mathcal{I}^{tf}} = \{\bar{d} | P^{\mathcal{I}}(\bar{d}) \geq_p \mathbf{f}\} = D^n \setminus P^J.$$

An interpretation $\mathcal{I}$ is three-valued iff $(P^{ct})^{\mathcal{I}^{tf}}$ and $(P^{cf})^{\mathcal{I}^{tf}}$ are disjoint for any $P \in \Sigma$. $\mathcal{I}$ is two-valued iff $(P^{ct})^{\mathcal{I}^{tf}}$ and $(P^{cf})^{\mathcal{I}^{tf}}$ are each others complement in D^n. Also, if $\mathcal{I} \leq_p \mathcal{J}$, then, for each P, $(P^{ct})^{\mathcal{I}^{tf}} \subseteq (P^{ct})^{\mathcal{J}^{tf}}$ and $(P^{cf})^{\mathcal{I}^{tf}} \subseteq (P^{cf})^{\mathcal{J}^{tf}}$.

Definition 2.3 (φ^{ct} and φ^{cf})**.** For any given Σ-formula $\varphi[\bar{x}]$, let $\varphi^{ct}[\bar{x}]$ be the Σ^{tf}-formula obtained by first reducing to negation normal form and then replacing all occurrences of positive literals $P(\bar{t})$ by $P^{ct}(\bar{t})$ and all negative literals $\neg P(\bar{t})$ by $P^{cf}(\bar{t})$, and let $\varphi^{cf}[\bar{x}]$ be the formula $(\neg\varphi[\bar{x}])^{ct}$.

An interesting property of the formulas φ^{ct} and φ^{cf} is that they do not contain negations. Also, the following proposition is well-known.

Proposition 2.1 (Feferman, 1984)**.** *For every Σ-formula φ and interpretation $\mathcal{I}$, it holds that $\varphi[\bar{d}]^{\mathcal{I}} \geq_p \mathbf{t}$ if and only if $\varphi[\bar{d}]^{+I-J} = \mathbf{t}$ if and only if $(\varphi^{ct}[\bar{d}])^{\mathcal{I}^{tf}} = \mathbf{t}$. Also, $\varphi[\bar{d}]^{\mathcal{I}} \geq_p \mathbf{f}$ if and only if $\varphi[\bar{d}]^{+J-I} = \mathbf{f}$ if and only if $(\varphi^{cf}[\bar{d}])^{\mathcal{I}^{tf}} = \mathbf{t}$.*

2.2 FO(ID)

In this subsection we recall FO(ID) (Denecker & Ternovska, 2008), the extension of FO with a construct to respresent some of the most common forms of inductive definitions, such as monotone induction, induction over a well-founded order or iterated induction. As illustrated by Denecker and Ternovska, FO(ID) can be used to represent different sorts of common knowledge, such as temporal and dynamic domains, the closed world assumption, defaults, causality, etc. In this paper, we use definitions to symbolically represent propagation, not only for FO formulas, as already mentioned in the introduction, but also propagation for inductive definitions themselves.

A definitional rule over a vocabulary Σ is an expression of the form $\forall\bar{x}\ P(\bar{t}) \leftarrow \varphi$ where $P(\bar{t})$ is an atomic formula and φ an FO formula. The symbol $\leftarrow$ is a new connective, called the *definitional implication*, to be distinguished from the FO material implication symbol $\Leftarrow$ (or its converse $\Rightarrow$). A definition Δ is a finite set of definitional rules. A predicate symbol P in the head of a rule of Δ is called a *defined predicate*; all other predicate and function symbols in Δ are called the *open symbols* or the *parameters* of the definition; the set of defined predicates is denoted $Def(\Delta)$, the remaining symbols $Open(\Delta)$ (note that $Open(\Delta)$ therefore also includes Σ^F).

Given an interpretation for its open predicates, each definition will have a unique model, that can be constructed by "firing" its rules in an appropriate order. Before defining this formally, we first consider an example.

Example 2.1. Reachability in a graph is not expressible in FO. That is, there is no FO formula φ over the vocabulary consisting of two predicates $Edge/2$ and $Reach/2$ such that in any model M of $\varphi, (d_1, d_2) \in Reach^M$ iff there is a non-empty path from d_1 to d_2 in the graph represented by $Edge^M$. We can represent reachability with an inductive definition however. The following definition defines the predicate *Reach* in terms of the open predicate *Edge*.

$$\left\{\begin{array}{l}\forall x \forall y\, Reach(x,y) \leftarrow Edge(x,y). \\ \forall x \forall y\, Reach(x,y) \leftarrow \exists z (Reach(x,z) \wedge Reach(z,y)).\end{array}\right\}$$

Definition 2.4 (Operator T^O_Δ)**.** With a definition Δ and a given $Open(\Delta)$-interpretation O, we define the immediate consequence operator T^O_Δ on two-valued $Def(\Delta)$-interpretations such that $T^O_\Delta(I) = J$ iff for each defined predicate P/n and tuple $\bar{d} \in D^n$, it holds that $P^J(\bar{d}) = \mathbf{t}$ iff there exists a rule $\forall x\, P(\bar{t}) \leftarrow \varphi[\bar{x}]$, such that $\bar{t}^{(O+I)} = \bar{d}$ and $\varphi^{(O+I)}[\bar{d}] = \mathbf{t}$.

The model of a *positive definition* (i.e., no defined predicates occur negatively in the body of rules) can be defined as the *least fixpoint* of this immediate consequence operator. We use $Mod_{I|_{Open(\Delta)}}(\Delta)$ to denote the model of the definition Δ extending the restriction of I to the open predicates and function symbols of Δ. When Δ has no open predicates, we omit the subscript and simply use $Mod(\Delta)$. We postpone going into more detail about how to construct the model of a general (non-monotonic) inductive definition until Section 5. In the next two sections, we only use positive definitions.

FO(ID) formulas are inductively defined by the same rules as standard FO formulas, augmented with one extra case:

- A definition Δ over Σ is an FO(ID) formula (over Σ)).

Note that rule bodies do not contain definitions, and that rules only occur inside definitions and are not FO(ID) formulas themselves whereas definitions can be used in FO(ID) formulas anywhere atoms can be used.

We can now define the satisfaction relation $I \models \varphi$ of FO(ID) using the standard inductive rules of FO, augmented with one extra rule:

- $I \models \Delta$ if $I = Mod_{I|_{Open(\Delta)}}(\Delta)$,

From now on, we assume without loss of generality that for any definition Δ, it holds that every defined predicate $P \in Def(\Delta)$ is defined by exactly one rule, denoted by $\forall \bar{x}(P(\bar{x}) \leftarrow \varphi_P[\bar{x}])$. Indeed, any definition Δ can be brought into this form by a process similar to predicate completion. The transformation consists of first transforming rules of the form $\forall \bar{x}(P(\bar{t}) \leftarrow \varphi)$ into equivalent rules $\forall \bar{y}(P(\bar{y}) \leftarrow \exists \bar{x}(\bar{y} = \bar{t} \wedge \varphi))$. Next, one merges all rules of the form $\forall \bar{x}(P(\bar{x}) \leftarrow \varphi_i[\bar{x}])$ into $\forall \bar{x}(P(\bar{x}) \leftarrow \varphi_1[\bar{x}] \vee \ldots \vee \varphi_n[\bar{x}])$.

3. Propagation for FO

In this section we give a general, symbolic representation of propagation for first-order logic. For this, we base ourselves on the work by Wittocx et al. (2010). We come back to the precise relation between the material presented in this section and their work at the end of this section.

Suppose we have an FO theory T in vocabulary Σ, a pre-interpretation of Σ, and a finite three-valued interpretation $\mathcal{I}$ that represents some (incomplete) knowledge about the

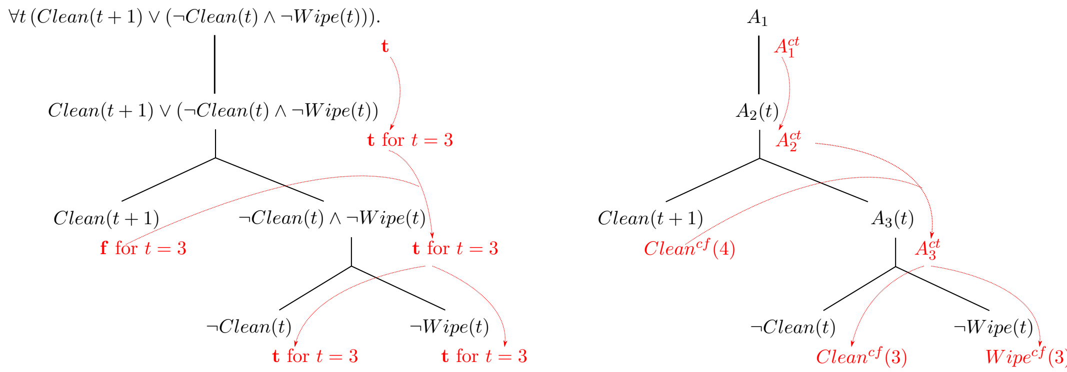

Figure 2: Propagation for FO.

predicates of Σ. We would now like to know the implications of this knowledge, assuming the theory T is satisfied in the context of $\mathcal{I}$. To find this out, we can look at the set $\mathcal{M}$ of all models of T that complete this three-valued interpretation, i.e., $\mathcal{M} = \{M \mid M \models T \text{ and } \mathcal{I} \leq_p M\}$. Given the partial information $\mathcal{I}$, everything that is true in all $M \in \mathcal{M}$ must certainly be true according to T, while everything that is false in all such M must certainly be false according to T. In other words, all the information that T allows us to derive from $\mathcal{I}$ can be captured by the greatest lower bound $\mathcal{G} = \text{glb}_{\leq_p}\mathcal{M}$.

In general, computing this greatest lower bound may be too expensive (the data complexity is in Δ_2^P) to be of practical use. However, we may still achieve useful results by computing some approximation $\tilde{\mathcal{M}}$ such that $\mathcal{I} \leq_p \tilde{\mathcal{M}} \leq_p \mathcal{G}$. We can compute such an approximation by *propagating* the three-valued interpretation $\mathcal{I}$ through the parse tree of T. We illustrate this with the following example.

Example 3.1. Consider the sentence ψ: $\forall t\, Clean(t+1) \Leftarrow Clean(t) \vee Wipe(t)$. Rewriting this into negation normal form, it becomes:

$$\forall t\, Clean(t+1) \vee (\neg Clean(t) \wedge \neg Wipe(t)).$$

Now, assume that ψ is satisfied, and that we know that *Clean* is false at timepoint 4. From the knowledge that ψ is satisfied, it immediately follows that, for all timepoints t, the disjunctive formula $Clean(t+1) \vee (\neg Clean(t) \wedge \neg Wipe(t))$ is satisfied. Using the fact that *Clean* is false at timepoint 4, we can now deduce that the conjunction $(\neg Clean(t) \wedge \neg Wipe(t))$ is true for timepoint 3. Therefore, in all models of ψ where *Clean* is false at timepoint 4, *Wipe* and *Clean* have to be false at timepoint 3. This reasoning process is illustrated on the left part of Figure 2.

We now construct a symbolic representation of this propagation process. First, we introduce some additional vocabulary Σ_{Aux} to refer to the different nodes of the parse tree on which this process operates. We then use this additional vocabulary to transform an FO formula into an *equivalence normal form* formula. This is similar to the Tseitin transformation (1968) for propositional logic.

Definition 3.1 ($ENF(\psi)$)**.** For an FO formula ψ in negation normal form, we introduce a new predicate symbol A_φ of arity n for each non-literal subformula $\varphi[\bar{x}]$ of ψ with n

free variables. We denote the set of these new predicates by $Aux(\psi)$. Each of these new predicate symbols is defined by a formula $Eq(A_\varphi)$ as follows. To make notation simpler we assume that each $\varphi_1, \ldots, \varphi_n$ is a non-literal subformula. The definitions are analogous whenever a φ_i is a literal, but instead of A_{φ_i} the literal φ_i itself is used in the body of the definition.

- If $\varphi[\bar{x}]$ is a subformula of the form $\varphi_1[\bar{x_1}] \wedge \varphi_2[\bar{x_2}] \wedge \ldots \wedge \varphi_n[\bar{x_n}]$, then $Eq(A_\varphi)$ is $\forall \bar{x}\,(A_\varphi(\bar{x}) \Leftrightarrow A_{\varphi_1}(\bar{x_1}) \wedge A_{\varphi_2}(\bar{x_2}) \wedge \ldots \wedge A_{\varphi_n}(\bar{x_n}))$.
- If $\varphi[\bar{x}]$ is a subformula of the form $\varphi_1[\bar{x_1}] \vee \varphi_2[\bar{x_2}] \vee \ldots \vee \varphi_n[\bar{x_n}]$, then $Eq(A_\varphi)$ is $\forall \bar{x}\,(A_\varphi(\bar{x}) \Leftrightarrow A_{\varphi_1}(\bar{x_1}) \vee A_{\varphi_2}(\bar{x_2}) \vee \ldots \vee A_{\varphi_n}(\bar{x_n}))$.
- If $\varphi[\bar{x}]$ is a subformula of the form $\forall \bar{y}\, \varphi_1[\bar{x}, \bar{y}]$, then $Eq(A_\varphi)$ equals $\forall \bar{x}\,(A_\varphi(\bar{x}) \Leftrightarrow \forall \bar{y}\, A_{\varphi_1}(\bar{x}, \bar{y}))$.
- If $\varphi[\bar{x}]$ is a subformula of the form $\exists \bar{y}\, \varphi_1[\bar{x}, \bar{y}]$, then $Eq(A_\varphi)$ equals $\forall \bar{x}\,(A_\varphi(\bar{x}) \Leftrightarrow \exists \bar{y}\, A_{\varphi_1}(\bar{x}, \bar{y}))$.

We define the *equivalence normal form* of ψ as the set of all such $Eq(A_\varphi)$, and denote it as $ENF(\psi)$.

Example 3.2. According to this definition, the $ENF(\psi)$ theory from Example 3.1 is:

$$\begin{array}{rcl} A_1 & \Leftrightarrow & \forall t\, A_2(t). \\ \forall t\, A_2(t) & \Leftrightarrow & Clean(t+1) \vee A_3(t). \\ \forall t\, A_3(t) & \Leftrightarrow & \neg Clean(t) \wedge \neg Wipe(t). \end{array}$$

This is illustrated in the right side of Figure 2.

Using the auxiliary vocabulary, we can now write down the propagations shown in Figure 2 as the following implications.

$$\begin{array}{lcl} A_1 & \Rightarrow & A_2(3). \\ A_2(3) \wedge \neg Clean(4) & \Rightarrow & A_3(3). \\ A_3(3) & \Rightarrow & \neg Clean(3). \\ A_3(3) & \Rightarrow & \neg Wipe(3). \end{array}$$

Note that these rules are all *top-down* rules, that is, implications that propagate information about a subformula down the parse tree, to a component of that subformula (possibly using also information about other components of the subformula, as in the implication $A_2(3) \wedge \neg Clean(4) \Rightarrow A_3(3)$). In general, also *bottom-up* propagations are of course possible. For instance, from $Clean(4)$ we could derive $A_2(3)$. For every predicate A_ψ, we can derive from $Eq(A_\psi)$ a set of implications $\varphi_1 \Rightarrow \varphi_2$, such that each such propagation corresponds to deriving the consequent φ_2 from the antecedent φ_1 (so, different implications can be logically equivalent). This is defined in Table 1. The last column of this table indicates whether the rule is top-down (TD) or bottom-up (BU).

Definition 3.2 ($INF(\psi)$)**.** Given an equivalence $\varphi \in ENF(\psi)$ for a certain formula ψ, we denote with $Imp(\varphi)$ the set of all implications obtained through Table 1. We define the *implication normal form* of ψ, denoted by $INF(\psi)$, as follows: $INF(\psi) = \bigcup_{\varphi \in ENF(\psi)} Imp(\varphi)$.

$\varphi = Eq(A_\psi)$	$Imp(\varphi)$		
$\forall \bar{x}\ (L \Leftrightarrow L_1 \wedge \ldots \wedge L_n)$.	$\forall \bar{x}\ (L_1 \wedge \ldots \wedge L_n \Rightarrow L)$.		BU
	$\forall \bar{x}\ (\neg L_i \Rightarrow \neg L)$.	$1 \le i \le n$	BU
	$\forall \bar{x}\ (L \Rightarrow L_i)$.	$1 \le i \le n$	TD
	$\forall \bar{x}\ (\neg L \wedge L_1 \wedge \ldots \wedge L_{i-1} \wedge L_{i+1} \wedge \ldots \wedge L_n \Rightarrow \neg L_i)$.	$1 \le i \le n$	TD
$\forall \bar{x}\ (L \Leftrightarrow L_1 \vee \ldots \vee L_n)$.	$\forall \bar{x}\ (\neg L_1 \wedge \ldots \wedge \neg L_n \Rightarrow \neg L)$.		BU
	$\forall \bar{x}\ (L_i \Rightarrow L)$.	$1 \le i \le n$	BU
	$\forall \bar{x}\ (\neg L \Rightarrow \neg L_i)$.	$1 \le i \le n$	TD
	$\forall \bar{x}\ (L \wedge \neg L_1 \wedge \ldots \wedge \neg L_{i-1} \wedge \neg L_{i+1} \wedge \ldots \wedge \neg L_n \Rightarrow L_i)$.	$1 \le i \le n$	TD
$\forall \bar{x}\ (L[\bar{x}] \Leftrightarrow \forall \bar{y}\ L'[\bar{x}, \bar{y}])$.	$\forall \bar{x}\ ((\forall \bar{y}\ L'[\bar{x}, \bar{y}]) \Rightarrow L[\bar{x}])$.		BU
	$\forall \bar{x} (\exists \bar{y}\ \neg L'[\bar{x}, \bar{y}]) \Rightarrow \neg L[\bar{x}])$.		BU
	$\forall \bar{x} \forall \bar{y}\ (L[\bar{x}] \Rightarrow L'[\bar{x}, \bar{y}])$.		TD
	$\forall \bar{x} \forall \bar{y}\ ((\neg L[\bar{x}] \wedge \forall \bar{z}\ (\bar{y} \neq \bar{z} \Rightarrow L'[\bar{x}, \bar{y}][\bar{y}/\bar{z}])) \Rightarrow \neg L'[\bar{x}, \bar{y}])$.		TD
$\forall \bar{x}\ (L[\bar{x}] \Leftrightarrow \exists \bar{y}\ L'[\bar{x}, \bar{y}])$.	$\forall \bar{x}\ ((\forall \bar{y}\ \neg L'[\bar{x}, \bar{y}]) \Rightarrow \neg L[\bar{x}])$.		BU
	$\forall \bar{x} (\exists \bar{y}\ L'[\bar{x}, \bar{y}]) \Rightarrow L[\bar{x}])$.		BU
	$\forall \bar{x} \forall \bar{y}\ (\neg L[\bar{x}] \Rightarrow \neg L'[\bar{x}, \bar{y}])$.		TD
	$\forall \bar{x} \forall \bar{y}\ ((L[\bar{x}] \wedge \forall \bar{z}\ (\bar{y} \neq \bar{z} \Rightarrow \neg L'[\bar{x}, \bar{y}][\bar{y}/\bar{z}])) \Rightarrow L'[\bar{x}, \bar{y}])$.		TD

Table 1: From ENF to INF

In the work of Wittocx et al. (2010) it is proven that models of ψ and models of $INF(\psi)$ where A_ψ is true correspond, in the sense that the restriction of a model of $INF(\varphi) \wedge A_\psi$ to Σ is also a model of ψ, and vice versa, every model of ψ can be extended to a model of $INF(\psi) \wedge A_\psi$. These implications will form the core of our approximation method. While our approximation could be made more complete by adding more implications to $INF(\psi)$, the above definition tries to strike a balance between completeness and the ease of automatically deriving the implications.

Example 3.3. For each of the three formulas φ in $ENF(\psi)$ in Example 3.2, the following table shows the corresponding set of implications $Imp(\varphi)$. The complete theory $INF(\psi)$ consists of the union of these three sets.

$A_1 \Leftrightarrow \forall t\, A_2(t)$.	$\forall t\, (A_2(t) \Leftrightarrow Clean(t+1) \vee A_3(t))$.	$\forall t\, (A_3(t) \Leftrightarrow (\neg Clean(t) \wedge \neg Wipe(t)))$.
$(\forall t\, A_2(t)) \Rightarrow A_1$.	$\forall t\, (\neg Clean(t+1) \wedge \neg A_3(t) \Rightarrow \neg A_2(t))$.	$\forall t\, (\neg Clean(t) \wedge \neg Wipe(t) \Rightarrow A_3(t))$.
$(\exists t\, \neg A_2(t)) \Rightarrow \neg A_1$.	$\forall t\, (Clean(t+1) \Rightarrow A_2(t))$.	$\forall t\, (Clean(t) \Rightarrow \neg A_3(t))$.
$\forall t\, (A_1 \Rightarrow A_2(t))$.	$\forall t\, (A_3(t) \Rightarrow A_2(t))$.	$\forall t\, (Wipe(t) \Rightarrow \neg A_3(t))$.
$\forall t\, ((\neg A_1 \wedge \forall t'\, (t \neq t'$	$\forall t\, (\neg A_2(t) \Rightarrow \neg Clean(t+1))$.	$\forall t\, (A_3(t) \Rightarrow \neg Clean(t))$.
$\Rightarrow A_2(t'))) \Rightarrow \neg A_2(t))$.	$\forall t\, (\neg A_2(t) \Rightarrow \neg A_3(t))$.	$\forall t\, (A_3(t) \Rightarrow \neg Wipe(t))$.
	$\forall t\, (A_2(t) \wedge \neg A_3(t) \Rightarrow Clean(t+1))$.	$\forall t\, (\neg A_3(t) \wedge \neg Wipe(t) \Rightarrow Clean(t))$.
	$\forall t\, (A_2(t) \wedge \neg Clean(t+1) \Rightarrow A_3(t))$.	$\forall t\, (\neg A_3(t) \wedge \neg Clean(t) \Rightarrow Wipe(t))$.

The reader can verify that the four implications representing the propagation in Example 3.1 all indeed belong to $INF(\psi)$.

The propagation process in Example 3.1 can now be described as a least fixpoint-computation, where we 'apply' the implications (i.e., infer the head when the body is already inferred), until we no longer can infer any new information. We will represent this fixpoint computation as an inductive definition in the syntax of FO(ID). However, there are two complications.

First, in this paper, we do not always need all of the implications in $INF(\psi)$. Indeed, there will typically be some subset $\sigma \subseteq \Sigma$ of symbols about which we already know all there is to know. In the conformant planning example, for instance, this will be the case for the existentially quantified predicate $Wipe/2$, simply because we will use a model expansion system to guess a complete interpretation for this predicate. The job of the propagation process is then to figure out the consequences of each particular guess. For this, the implications with a predicate from σ (i.e., $Wipe/2$) in their head are obviously not needed.

Second, the fixpoint computation not only needs to infer that atoms are true but also that they are false. However, the syntax of FO(ID) does not allow negative literals in the heads of rules. Therefore, our definition will not contain rules with the predicates P of the original vocabulary Σ in their head, but will instead use predicates P^{ct} and P^{cf} from the Σ^{tf}-vocabulary. Since we do not need rules with the fully known predicates σ in the head, we will only introduce these P^{ct} and P^{cf} predicates for those P that are in $\Sigma \setminus \sigma$. For a given formula φ, we therefore define φ^{ct}_{σ} as the formula φ^{ct} (see Definition 2.3) but with P^{ct} replaced by P and P^{cf} by $\neg P$ for every predicate $P \in \sigma$.

Definition 3.3 ($Approx_{\sigma}(\psi)$)**.** For a formula ψ and $\sigma \subseteq \Sigma$, we define $Approx_{\sigma}(\psi)$ as the inductive definition that contains, for every sentence $\forall \bar{x}\ (\varphi \Rightarrow L[\bar{x}])$ of $INF(\psi)$ in which L is a literal of a predicate not in σ, the definitional rule $\forall \bar{x}(L[\bar{x}]^{\mathrm{ct}}_{\sigma} \leftarrow \varphi^{\mathrm{ct}}_{\sigma})$. We also define $Approx^{BU}_{\sigma}(\psi)$ (and $Approx^{TD}_{\sigma}(\psi)$) in the same way as $Approx_{\sigma}(\psi)$, but only containing definitional rules coming from the *bottom-up* (respectively, *top-down*) rules of $INF(\psi)$.

We can often assume without loss of generality that $\sigma = \emptyset$. Whenever this is the case we drop the σ and use $Approx(\psi)$ rather than $Approx_{\sigma}(\psi)$, to denote the approximative definition.

Example 3.4. Using the implications $INF(\psi)$ of Example 3.3, we obtain the definition shown in Figure 3 for $Approx(\psi)$. If we take $\sigma = \{Wipe\}$, we get the same definition for $Approx_{\sigma}(\psi)$ as in Figure 3, apart from the last seven definitional rules that are replaced by the following five definitional rules.

$$\left\{ \begin{array}{lcl} & \vdots & \\ A_3^{ct}(t) & \leftarrow & Clean^{cf}(t) \land \neg Wipe(t). \\ A_3^{cf}(t) & \leftarrow & Clean^{ct}(t). \\ A_3^{cf}(t) & \leftarrow & Wipe(t). \\ Clean^{cf}(t) & \leftarrow & A_3^{ct}(t). \\ Clean^{ct}(t) & \leftarrow & A_3^{cf}(t) \land \neg Wipe(t). \end{array} \right\}$$

In contrast with $Approx(\psi)$ this definition no longer approximates the predicate $Wipe$. The definition $Approx_{\sigma}(\psi)$ can be used to find out what certainly holds or not holds given a *two valued* interpretation for the predicates in σ.

Example 3.5. For a larger example, we look again at the Example 1.1. Let us again take $\psi = \psi_{act}$ and $\sigma = \{Wipe\}$. Then the definition $Approx_{\sigma}(\psi)$ can be found in the Appendix A.

This approximative definition has some useful properties, which we formulate in the next two theorems. The first property is that, when using the approximative definition

$$\left\{\begin{array}{ll}
A_1^{ct} & \leftarrow \forall t\, A_2^{ct}(t). \\
A_1^{cf} & \leftarrow \exists t\, A_2^{cf}(t). \\
A_2^{ct}(t) & \leftarrow A_1^{ct}. \\
A_2^{cf}(t) & \leftarrow A_1^{cf} \wedge \forall t' \,(t \neq t' \Rightarrow A_2^{ct}(t')). \\
 & \\
A_2^{cf}(t) & \leftarrow Clean^{cf}(t+1) \wedge A_3^{cf}(t). \\
A_2^{ct}(t) & \leftarrow Clean^{ct}(t+1). \\
A_2^{ct}(t) & \leftarrow A_3^{ct}(t). \\
Clean^{cf}(t+1) & \leftarrow A_2^{cf}(t). \\
A_3^{cf}(t) & \leftarrow A_2^{cf}(t). \\
Clean^{ct}(t+1) & \leftarrow A_2^{ct}(t) \wedge A_3^{cf}(t). \\
A_3^{ct}(t) & \leftarrow A_2^{ct}(t) \wedge Clean^{cf}(t+1). \\
 & \\
A_3^{ct}(t) & \leftarrow Clean^{cf}(t) \wedge Wipe^{cf}(t). \\
A_3^{cf}(t) & \leftarrow Clean^{ct}(t). \\
A_3^{cf}(t) & \leftarrow Wipe^{ct}(t). \\
Clean^{cf}(t) & \leftarrow A_3^{ct}(t). \\
Wipe^{cf}(t) & \leftarrow A_3^{ct}(t). \\
Clean^{ct}(t) & \leftarrow A_3^{cf}(t) \wedge Wipe^{cf}(t). \\
Wipe^{ct}(t) & \leftarrow A_3^{cf}(t) \wedge Clean^{cf}(t).
\end{array}\right\}$$

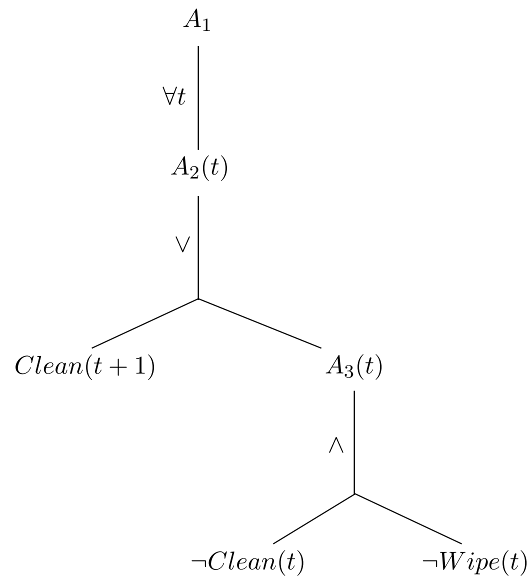

Figure 3: Example of an approximative definition

together with an encoding of a three-valued interpretation $\mathcal{I}$ of the original vocabulary, we can give an exact characterization of what the approximative definition computes. Indeed, in this setting, $Approx^{BU}(\psi)$ actually encodes the three-valued Kleene evaluation of ψ in $\mathcal{I}$. Moreover, adding the top-down rules does not change this, since they will not compute actually anything, as long as only information about the original vocabulary is provided as input. Before we can formally state this property, we need to define how we encode a four-valued interpretation as a definition. From here on, we assume that for any vocabulary Σ and Σ-pre-interpretation I, Σ contains a constant symbol C_d for every domain element d in the domain D of I, and that for the pre-interpretation I it holds that $(C_d)^I = d$. This allows us to identify C_d and d and therefore, abusing notation, we will use d to denote C_d in what follows.

Definition 3.4 ($\Delta_{\mathcal{I}}$)**.** Given a four-valued Σ-interpretation $\mathcal{I}$, the *definition associated to* $\mathcal{I}$ is denoted by $\Delta_{\mathcal{I}}$ and is defined by

$$\begin{array}{rcl}\Delta_{\mathcal{I}} & = & \{P^{ct}(\bar{d}) \leftarrow \mathbf{t} \mid P^{\mathcal{I}}(\bar{d}) \geq_p \mathbf{t}\} \\ & & \cup \{P^{cf}(\bar{d}) \leftarrow \mathbf{t} \mid P^{\mathcal{I}}(\bar{d}) \geq_p \mathbf{f}\}\end{array}$$

Theorem 3.1. *Given a Σ-formula ψ and a four-valued Σ-interpretation $\mathcal{I}$, the following holds:*

a) In the case that $\mathcal{I}$ is three-valued it holds that $Approx(\psi) \cup \Delta_{\mathcal{I}}$ is logically equivalent to $Approx^{BU}(\psi) \cup \Delta_{\mathcal{I}}$, that is, $Mod(Approx(\psi) \cup \Delta_{\mathcal{I}}) = Mod(Approx^{BU}(\psi) \cup \Delta_{\mathcal{I}})$

b) Let M be $Mod(Approx(\psi) \cup \Delta_{\mathcal{I}})$, v_1 the truth value of A^{ct}_{ψ} in M and v_2 be the truth value of $\neg A^{cf}_{\psi}$ in M, then (v_1, v_2) corresponds to the four-valued truth value $\psi^{\mathcal{I}}$, i.e., $\psi^{\mathcal{I}} = \tau(v_1, v_2)$.

Proof. See Appendix B. □

In summary, what this theorem says is that, first of all, the approximation always computes the four-valued Belnap evaluation of ψ in the four-valued structure $\mathcal{I}$. Moreover, this computation is done by the bottom-up rules of the approximation alone. If $\mathcal{I}$ is three-valued, then the top-down rules actually have no effect at all. If $\mathcal{I}$ is four-valued, then they may still serve a purpose, however: once the bottom-up rules have derived that some subformula is inconsistent, they can then propagate this information to derive that smaller formulas are also inconsistent. To see this, consider the following formula $P \wedge Q$, and take for $\mathcal{I}$ the four-valued interpretation such that $P = \mathbf{i}$ and $Q = \mathbf{t}$. Then one can verify that the bottom-up rules in $Approx(\psi) \cup \Delta_{\mathcal{I}}$ will infer that both $A^{ct}_{P\wedge Q}$ and $A^{cf}_{P\wedge Q}$ are true. However, now the top-down rules can also infer that Q^{cf} has to be true.

In the theorem above the only information we add to the approximative definition is in the form of the definition $\Delta_{\mathcal{I}}$, i.e., we only assert the truth, resp. falsity of *domain atoms*. The following definition now allows us to assert the truth or the falsity of any grounded subformula $\varphi[\bar{d}]$.

Definition 3.5 (Δ_Φ)**.** Given a Σ-formula ψ, a Σ-pre-interpretation I, and a set Φ of formulas $(\neg)\varphi[\bar{d}]$, such that $\varphi[\bar{x}]$ is a subformula of ψ of arity n and $\bar{d} \in D^n$ where D is the domain of I, we then define Δ_Φ as follows:

$$\Delta_\Phi = \{A^{ct}_{\varphi}(\bar{d}) \leftarrow \mathbf{t} \mid \varphi[\bar{d}] \in \Phi\} \cup \{A^{cf}_{\varphi}(\bar{d}) \leftarrow \mathbf{t} \mid \neg\varphi[\bar{d}] \in \Phi\}.$$

If we assert in this way the truth (or the falsity) of a set of grounded subformulas Φ, then we will obtain an approximation of everything that holds (respectively, does not hold) in all models of Φ. However, as opposed to the theorem above, the next theorem does not give an exact characterization of the approximation we get.

Theorem 3.2. *Given a Σ-formula ψ, a set Φ as defined above and a subformula $\varphi'[\bar{x}']$ of ψ. Let M be $Mod(Approx(\psi) \cup \Delta_\Phi)$. If $M \models A^{ct}_{\varphi'}(\bar{d}')$ (resp. $A^{cf}_{\varphi'}(\bar{d}')$), then $\Phi \models \varphi'[\bar{d}']$ (resp. $\Phi \models \neg\varphi'[\bar{d}']$).*

Note that an interesting special case of this theorem is where we take Φ equal to $\{\psi\}$ and thus add $A^{ct}_{\psi} \leftarrow \mathbf{t}$ to $Approx(\psi)$. Then this definition gives an approximation of everything that is *certainly true* resp. *certainly false* in *all models* of ψ.

Returning now to the exact relationship between the work of Wittocx et al. (2010) and the content of this section, we see that Wittocx et al. are only interested in this special case, i.e., in approximating all models of a theory. For this reason their transformation from a formula ψ to $ENF(\psi)$ already includes a formula $A_\psi \Leftrightarrow \mathbf{t}$, which will cause the rule $A^{ct}_{\psi} \leftarrow \mathbf{t}$ to always be included in the approximating definition. All their soundness results have also been formulated and proven in this setting. However, it is not difficult to see that the proofs can be trivially adapted to a proof of Theorem 3.2 in the more general setting used in this section.

4. Approximating $\exists\forall SO$-Satisfiability Problems

We now use the approximate definition from the previous section to approximate the following problem. Take an $\exists\forall SO$ formula $F = \exists\bar{P}\forall\bar{Q} : \psi$. For ease of presentation, we assume that the second-order formulas in this paper contain no free predicate symbols, but all results generalize to the setting where there are free predicate symbols. We also assume that $\bar{Q}$ contains only predicate symbols. In what follows, we denote the vocabulary of ψ by Σ. The question we want to answer is whether the formula F is satisfied in a given finite-domain pre-interpretation I of the constant and function symbols of the formula. This satisfiability problem boils down to deciding whether we can find a witness for the satisfiability of this formula, in the following sense.

Definition 4.1 (Witness)**.** We call J a *witness* for the satisfiability of a formula $\exists\bar{P}\forall\bar{Q} : \psi$ given a finite Σ pre-interpretation I, if J is an interpretation of $\Sigma \setminus \bar{Q}$ extending I (i.e., J is an interpretation of the whole vocabulary without the universally quantified predicates) such that $\forall\bar{Q} : \psi$ is satisfied in J. Equivalently, J is a witness if in the three-valued Σ-interpretation $\mathcal{J}$ that expands J by assigning $\mathbf{u}$ to each domain atom $Q(\bar{d})$, it holds that $sv_{\mathcal{J}}(\psi) = \mathbf{t}$.

Our goal in this section is now to approximate an $\exists\forall SO$ satisfiability problems by an $\exists SO$ satisfiability problem in the following sense.

Definition 4.2 (Sound approximation)**.** Consider the $\exists\forall SO$ satisfiability problem for a formula $\exists\bar{P}\forall\bar{Q} : \psi$, where ψ is an FO formula in alphabet Σ. An $\exists SO(ID)$ formula of the form $\exists\bar{P}\bar{R} : \psi'$, where ψ' is an FO(ID) formula in the alphabet $\Sigma \setminus \bar{Q} \cup \bar{R}$, is a *sound approximation* of this satisfiability problem if, whenever J is a witness for the satisfiability of $\exists\bar{P}\bar{R} : \psi'$, then $J|_{\Sigma\setminus\bar{Q}}$ is a witness for the satisfiability of $\exists\bar{P}\forall\bar{Q} : \psi$.

In other words, a sound approximation G of the satisfiability problem for an $\exists\forall SO$ formula F is a stronger $\exists SO(ID)$ formula, i.e., one that has fewer witnesses for $\bar{P}$.

4.1 A Naive Method

We can now use the results of Theorem 3.1 to construct a sound approximation for a given $\exists\forall SO$ formula.

Definition 4.3 ($\mathcal{APP}(F)$)**.** Given a formula $F = \exists\bar{P}\forall\bar{Q} : \psi$. Take σ to be the alphabet of all function symbols in ψ and the predicates $\bar{P}$. We define $\mathcal{APP}(F)$ as the $\exists SO$ formula $\exists\bar{P}\exists\bar{R} : \mathit{Approx}_\sigma(\psi) \land A^{ct}_\psi$ in the vocabulary $\sigma \cup \bar{R}$, where $\bar{R} = (\bar{Q} \cup \mathit{Aux}(\psi))^{tf}$.

The intuition here is that for any σ-interpretation I, $\mathit{Approx}_\sigma(\psi)$ will give the result of the four-valued evaluation $\psi^{\mathcal{I}}$ in the Σ-interpretation $\mathcal{I}$ that expands I by assigning unknown to all universally quantified predicates $\bar{Q}$. If the entire FO formula ψ evaluates to true in this four-valued interpretation, we know that ψ will be satisfied in *any* Σ interpretation that expands I (in other words, for every interpretation of the $\bar{Q}$ predicates), and thus that I is a witness for the satisfiability of the entire formula F. The auxiliary predicates $\mathit{Aux}(\psi)$ — introduced by the transformation to ENF — are needed because of the way in which the propagation works, but their value is completely determined by that of $\bar{P}$.

Proposition 4.1. *For each $\exists\forall SO$ formula F of the form $\exists\bar{P}\forall\bar{Q} : \psi$, it holds that $\mathcal{APP}(F)$ is a sound approximation of F.*

Proof. This follows immediately from Theorem 3.1, where we take as three-valued interpretation, the interpretation $\mathcal{I}$ such that $(Q(\bar{d}))^{\mathcal{I}} = \mathbf{u}$ for all $Q \in \bar{Q}$ and $\bar{d} \in D^n$, and $(P(\bar{d}))^{\mathcal{I}} = (P(\bar{d}))^I$ for all $P \in \bar{P}$ and $\bar{d} \in D^n$, with D the domain of I. $\square$

For example, if we take F to be the formula $\exists P \forall Q : \psi$ where $\psi = P \vee Q$, then $\mathcal{APP}(F)$ becomes:

$$\exists P, Q^{ct}, Q^{cf}, A_\psi^{ct}, A_\psi^{cf} : \left\{ \begin{array}{l} A_\psi^{ct} \leftarrow P \vee Q^{ct} \\ A_\psi^{cf} \leftarrow \neg P \wedge Q^{cf} \\ Q^{ct} \leftarrow A_\psi^{ct} \wedge \neg P \\ Q^{cf} \leftarrow A_\psi^{cf} \end{array} \right\} \wedge A_\psi^{ct}.$$

We start from an interpretation O of the open predicate P of the definition $Approx_{\{P\}}(\psi)$. Let us take the interpretation O that makes P true. The unique model I of the definition that extends O is then the interpretation that assigns true to A_ψ^{ct} and false to Q^{ct}, Q^{cf} and A_ψ^{cf}. Therefore, this I satisfies both $Approx_{\{P\}}(\psi)$ and A_ψ^{ct}. Hence, it is a witness for the satisfiability of $\mathcal{APP}(F)$, and, indeed, it is also a witness for the satisfiability of the original formula $\exists P \forall Q : P \vee Q$.

This approximation method is sound, but for many applications still too incomplete. Indeed, let us look at the following formula: $F = \forall Q : Q \vee \neg Q$. Then $\mathcal{APP}(F)$ becomes:

$$\exists Q^{ct}, Q^{cf}, A_\psi^{ct}, A_\psi^{cf} : \left\{ \begin{array}{l} A_\psi^{ct} \leftarrow Q^{ct} \vee Q^{cf} \\ A_\psi^{cf} \leftarrow Q^{cf} \wedge Q^{ct} \\ Q^{ct} \leftarrow A_\psi^{ct} \wedge Q^{ct} \\ Q^{cf} \leftarrow A_\psi^{cf} \\ Q^{cf} \leftarrow A_\psi^{ct} \wedge Q^{cf} \\ Q^{ct} \leftarrow A_\psi^{cf} \end{array} \right\} \wedge A_\psi^{ct}.$$

The definition does not entail that A_ψ^{ct}, so $\mathcal{APP}(F)$ is unsatisfiable, even though the original formula F is clearly always satisfied. The problem here is that, as we showed in the previous section, the definition encodes the three-valued Kleene evaluation, which is not strong enough to find out that the formula F is satisfied. To do this, we need the stronger supervaluation.

Recall that in the preliminaries we saw that supervaluation and Kleene evaluation are in general not equal. However, for some formulas ψ they *are* equal. In the literature, several *classes of formulas* for which they agree have been proposed, e.g., in the context of locally closed databases (Denecker et al., 2010), or in the context of reasoning with incomplete first-order knowledge (Liu & Levesque, 1998). The latter introduces a normal form $\mathcal{NF}$ for first-order formulas, for which the supervaluation coincides with the Kleene evaluation, and proves for certain classes of formulas that they are in the normal form $\mathcal{NF}$. One such class is that of all CNF formulas in which every two literals are *conflict-free*: a pair of literals is conflict-free if they either have the same polarity, or they use different predicates, or they use different constants at some argument position. It immediately follows that our approximation is complete for $\exists\forall SO$ formulas in which the first-order formula satisfies this condition.

Proposition 4.2. *Each $\exists\forall SO$ formula F of the form $\exists\bar{P}\forall\bar{Q} : \psi$, where ψ is in the $\mathcal{NF}$ normal form (according to Liu & Levesque, 1998) is satisfiable with respect to a given finite pre-interpretation I* if and only if *the $\exists SO$-formula $\mathcal{APP}(F)$ is satisfiable w.r.t. I.*

Proof. This follows immediately from the results by Liu and Levesque and Theorem 3.1. $\square$

4.2 A More Complete Method

Unfortunately, many applications give rise to formulas in which the first-order part falls outside the class $\mathcal{NF}$, which means that completeness of our method is not guaranteed. Particularly troublesome in practice are formulas of the common form $\exists\bar{P}\forall\bar{Q} : \psi_1 \Rightarrow \psi_2$. For such formulas, the naive approximation method of the previous section tries to find interpretations for $\bar{P}$ such that the implication $\varphi = (\psi_1 \Rightarrow \psi_2)$ holds for all $\bar{Q}$. However, if we look at the details of the approximative definitions, we find that A^{ct}_{φ} is defined by a rule with a body $\psi_1^{cf} \vee \psi_2^{ct}$. In other words, the approximation will only derive that φ holds for all $\bar{Q}$ if it is either the case that ψ_1 is false for all $\bar{Q}$ or that ψ_2 is true for all $\bar{Q}$. However, this will rarely be the case. In most practical applications, the witnesses of interest will typically satisfy the implication $\psi_1 \Rightarrow \psi_2$ not because they always falsify $\psi 1$ or always satisfy ψ_2, but rather because each interpretation for $\bar{Q}$ that satisfies ψ_1 also satisfies $\psi 2$. For instance, in the conformant planning example, there will always be interpretations for the fluents that do not satisfy the action theory ψ_{act}, because they arbitrarily assign some fluent a value that is wrong for its initial value and the actions that are performed. Even if a set of actions is a completely correct conformant plan, it therefore cannot make the goal certainly true, because it will still be unsatisfied in some of these wrong interpretations of the fluents. Of course, this should not bother a good method for finding conformant plans. The only thing that should matter is that the goal is satisfied in those interpretations of the fluents that do satisfy the action theory.

Luckily, our approximation method can also be used to discover this kind of witnesses. The only thing that is required is to add to the approximative definition $\Delta = Approx_{\sigma}(\varphi)$ a rule $A^{ct}_{\psi_1} \leftarrow \mathbf{t}$. By doing do, we seed our approximation with the assumption that ψ_1 holds. Starting from this assumption, the top-down rules will then derive properties of the predicates $\bar{Q}$ that are shared by all interpretations for $\bar{Q}$ that actually satisfy ψ_1. The bottom-up rules will then propagate this information upwards and discover whether these properties suffice to ensure that ψ_2 also holds. If they do, then we know that ψ_2 indeed must hold in every interpretation for $\bar{Q}$ that satisfies ψ_1 and that we therefore have found a witness for our formula.

If we want to find both witnesses of this kind and degenerate witnesses that either make ψ_1 false for all $\bar{Q}$ or ψ_2 true for all $\bar{Q}$, we could simply combine our new method with the old one and check either whether $A^{ct}_{\psi_2}$ holds according to $\Delta \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}$ or whether A^{ct}_{φ} holds according to just Δ itself. However, it turns out that this is not necessary: we can achieve the same effect by just checking whether $\Delta \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}$ implies A^{ct}_{φ}. This is because, first, the definition $\Delta \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}$ will be able to derive A^{ct}_{φ} whenever Δ itself can: if Δ can derive that $A^{ct}_{\psi_2}$ then $\Delta \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}$ will obviously still be able to do so; if Δ would be able to derive that $A^{cf}_{\psi_1}$, then $\Delta \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}$ will also be able to do so, simply because our approximation has no flow of information between the ct and cf variants of the same formula, so the additional assumption that $A^{ct}_{\psi_1}$ holds will not change the original derivation of $A^{cf}_{\psi_1}$. Second, if $\Delta \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}$ can derive $A^{ct}_{\psi_2}$, then it also derives A^{ct}_{φ}, simply because it contains the rule $A^{ct}_{\varphi} \leftarrow A^{cf}_{\psi_1} \vee A^{ct}_{\psi_2}$. Therefore, we can find both kinds of witnesses by checking whether A^{ct}_{φ} is implied by the single definition $\Delta \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}$.

Definition 4.4 ($\mathcal{APP}^{\Rightarrow}(F)$)**.** For an $\exists\forall SO$ formula $F = \exists\bar{P}\forall\bar{Q} : \psi$, where ψ is of the form $\psi_1 \Rightarrow \psi_2$, we define $\mathcal{APP}^{\Rightarrow}(F)$ as $\exists\bar{P}\exists\bar{R} : \Delta^{\Rightarrow} \wedge A^{ct}_{\psi}$, where $\Delta^{\Rightarrow}$ is the definition $Approx_{\sigma}(\psi_1 \Rightarrow \psi_2) \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}$.

Note that we obtain Definition 4.3 as a special case when taking the trivial formula $\mathbf{t}$ for ψ_1. This approximation method is still sound, as the following proposition states.

Proposition 4.3. *Given a formula F of the form $\exists\bar{P}\forall\bar{Q} : \psi$, where $\psi = \psi_1 \Rightarrow \psi_2$, the $\exists SO(ID)$ formula $\mathcal{APP}^{\Rightarrow}(F)$ is a sound approximation of F.*

Proof. See Appendix C. □

Since the approximative definition $\Delta^{\Rightarrow}$ in $\mathcal{APP}^{\Rightarrow}(F)$ contains all the rules of $Approx(F)$, it is not hard to see that this new approximation method is at least as complete as the one using $\mathcal{APP}(F)$ (Definition 4.3). Moreover, as can be seen from the following example, it is also strictly more complete.

Example 4.1. Let us consider the following formula $F = \exists P\forall Q : (Q \Leftarrow P) \Rightarrow Q$. We have that $P = \mathbf{t}$ is clearly a witness for this satisfiability problem. If we denote $(Q \Leftarrow P) \Rightarrow Q$ by φ_1 and $(Q \Leftarrow P)$ by φ_2, then $\mathcal{APP}(F)$ is the following $\exists SO$ formula.

$$\exists P\bar{R} : \left\{ \begin{array}{lcl} A^{ct}_{\varphi_1} & \leftarrow & A^{cf}_{\varphi_2} \vee Q^{ct} \\ A^{cf}_{\varphi_1} & \leftarrow & A^{ct}_{\varphi_2} \wedge Q^{cf} \\ A^{ct}_{\varphi_2} & \leftarrow & \neg P \vee Q^{ct} \\ A^{cf}_{\varphi_2} & \leftarrow & P \wedge Q^{cf} \\ Q^{ct} & \leftarrow & A^{ct}_{\varphi_2} \wedge P \\ Q^{cf} & \leftarrow & A^{cf}_{\varphi_2} \end{array} \right\} \wedge A^{ct}_{\varphi_1}.$$

Now, even for $P = \mathbf{t}$, the definition in the body of this formula will not entail $A^{ct}_{\varphi_1} = \mathbf{t}$. Therefore, $\mathcal{APP}(F)$ is not satisfiable. On the other hand, $\mathcal{APP}^{\Rightarrow}(F)$ is the same formula as above, apart from that the definition contains one more rule, that is, the rule $A^{ct}_{\varphi_2} \leftarrow \mathbf{t}$. It is easy to verify that $\mathcal{APP}^{\Rightarrow}(F)$ *is* satisfiable, and indeed has $P = \mathbf{t}$ as a witness.

Obviously, the new method is still complete on formulas $\exists\forall\psi_1 \Rightarrow \psi_2$, where $\psi_1 \Rightarrow \psi_2$ satisfies the normal form $\mathcal{NF}$. However, our method also works for many formulas outside this class. Unfortunately, it is difficult to characterize precisely how much more complete the new method is. For instance, one source of loss in completeness comes from the fact that our current translation to ENF cannot recognize multiple occurrences of the same subformula, and will introduce a different Tseitin predicate for each occurrence. Even though we cannot guarantee completeness for our method in general, we always found all solutions in the conformant planning benchmarks we considered in Section 6.

A final remark about this method is that the approximative definition $Approx_\sigma(\psi_1 \Rightarrow \psi_2)$ contains a number of rules that are superfluous in our context. Indeed, with our method, this definition takes as its input an interpretation for $\bar{P}$ together with the assumption that ψ_1 is certainly true. It then uses the bottom-up and top-down rules derived from ψ_1 to compute the effect of these inputs on the predicates $\bar{Q}$. Finally, the rules derived from ψ_2 then compute whether the derived information about $\bar{Q}$ suffices to make ψ_2 certainly true. However, as we know from Theorem 3.1, only the bottom-up rules for ψ_2 are needed for this. Therefore, the top-down rules for ψ_2 actually contribute nothing and could just as well be removed. Adapting Definition 4.4 to use $\Delta^{\Rightarrow}_{BU} = \Delta^{\Rightarrow} \setminus Approx^{TD}_\sigma(\psi_2)$ instead of $\Delta^{\Rightarrow}$ leads to the following definition.

Definition 4.5 ($\mathcal{APP}^{\Rightarrow}_{BU}(F)$)**.** For an $\exists\forall SO$ formula $F = \exists\bar{P}\forall\bar{Q} : \psi$, where ψ is of the form $\psi_1 \Rightarrow \psi_2$, we define $\mathcal{APP}^{\Rightarrow}_{BU}(F)$ as $\exists\bar{P}\exists\bar{R} : \Delta^{\Rightarrow}_{BU} \wedge A^{ct}_\psi$, where

$$\Delta^{\Rightarrow}_{BU} = Approx_\sigma(\psi_1 \Rightarrow \psi_2) \setminus Approx^{TD}_\sigma(\psi_2) \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}.$$

It follows directly from Theorem 3.1 and Proposition 4.3 that this too is a sound approximation. Having removed the top-down rules from the approximation of ψ_2, the remaining rules just serve, as we already know, to compute the Kleene evaluation of ψ_2. They do this by computing the Kleene evaluation of each subformula γ of ψ_2, for which they use the Tseitin predicates A^{ct}_γ and A^{pt}_γ. An alternative is to avoid these Tseitin predicates by defining $A^{ct}_{\psi_2}$ directly by the single rule:

$$A^{ct}_{\psi_2} \leftarrow (\psi_2)^{ct}$$

This variant is summarized in the following definition.

Definition 4.6 ($\mathcal{APP}^{\Rightarrow}_{BU,Unf}(F)$)**.** For an $\exists\forall SO$ formula $F = \exists\bar{P}\forall\bar{Q} : \psi$, where ψ is $\psi_1 \Rightarrow \psi_2$, we define $\mathcal{APP}^{\Rightarrow}_{BU,Unf}(F)$ as $\exists\bar{P}\exists\bar{R} : \Delta^{\Rightarrow}_{BU,Unf} \wedge A^{ct}_\psi$, where

$$\Delta^{\Rightarrow}_{BU,Unf} = Approx_\sigma(\psi) \setminus Approx_\sigma(\psi_2) \cup \{A^{ct}_{\psi_2} \leftarrow (\psi_2)^{ct}\} \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}.$$

This approximation is actually equivalent to that of Def. 4.5. This follows from the fact that all bottom-up rules for ψ_2 are positive and non-recursive, which allows us to eliminate the Tseitin predicates introduced for the parse tree of ψ_2 by applying the *unfolding* procedure from Tamaki and Sato (1984). By iteratively applying this equivalence preserving procedure, we can reduce all the rules that were generated to approximate ψ_2 to just the single rule $A^{ct}_{\psi_2} \leftarrow (\psi_2)^{ct}$.

5. Approximating $\exists\forall SO(ID)$-Satisfiability Problems

Inductive definitions are important for knowledge representation. As argued by Denecker and Ternovska (2008), inductive definitions are not only used to represent mathematical concepts, but also the sort of common sense knowledge that is often represented by logic programs, such as dynamic domains, the closed world assumption, defaults, causality, etc.

Therefore, inductive definitions can make the task of representing a problem in logic considerably easier. An example of this is the use of inductively defined Situation Calculus for reasoning about actions (Denecker & Ternovska, 2007). Recall that in the introduction we showed how to represent T_{act} from Example 1.1 as an inductive definition Δ_{act}:

$$\left\{\begin{array}{lcl} Clean(t+1) & \leftarrow & Clean(t). \\ Clean(t+1) & \leftarrow & Wipe(t). \\ Clean(0) & \leftarrow & InitiallyClean. \end{array}\right\}.$$

The associated conformant planning problem can then be expressed as an $\exists\forall SO(ID)$ satisfiability problem:

$$\exists Wipe \forall Clean, InitiallyClean : \Delta_{act} \Rightarrow Clean(n).$$

As we will show in more detail in Section 7, a general conformant planning problem can be seen as a satisfiability problem of the form

$$\exists \bar{A} \forall \bar{I} \forall \bar{F} : (\Delta_{act} \wedge \psi_{init}) \Rightarrow (\psi_{prec} \wedge \psi_{goal}),$$

where the predicates $\bar{A}$ represent the actions, $\bar{I}$ the initial fluents and $\bar{F}$ the other fluents. The definition Δ_{act} defines how the fluents change in terms of the action, ψ_{init} is a first order formula about the initial situation, ψ_{prec} describes the preconditions of the actions and ψ_{goal} the goal. This motivates the extension of our approximation method to formulas including definitions. However, we do not analyze the general case where definitions may appear at arbitrary locations in a formula, but instead restrict attention to formulas of the form

$$\exists \bar{P} \forall \bar{Q} : (\Delta \wedge \psi_1) \Rightarrow \psi_2,$$

where Δ is a definition such that $Def(\Delta) \subseteq \bar{Q}$ and ψ_1 and ψ_2 are FO formulas. Even though these restrictions are not strictly necessary, they allow us to keep the technical details relatively simple (in particular, we avoid the need for approximation rules that infer that a definition as a whole is certainly true/false), while still covering the way in which definitions are typically used: under the assumption that all predicates indeed are what the definition Δ and the formula ψ_1 say they should be, ψ_2 then states what properties they should satisfy.

To extend our approximative method to $\exists\forall(\Delta \wedge \psi_1) \Rightarrow \psi_2$ satisfiability problems, we need a syntactic representation (i.e., an approximative definition) that describes sound inferences that can be made from the definition in a three-valued context. In this section we propose two ways to obtain such an approximative definition, and accordingly, two ways to approximate $\exists\forall(\Delta \wedge \psi_1) \Rightarrow \psi_2$ satisfiability problems. Before we can continue, we first need to recall some more preliminaries.

5.1 Preliminaries on the Well-founded Semantics for Inductive Definitions

Earlier, we defined the model of a positive inductive definition given a *two-valued* interpretation for the open predicates. From here on, the inductive definitions are no longer only *positive* definitions, so the model of a definition can no longer always be computed as the least fixpoint of the immediate consequence operator introduced in Section 2. Moreover, in what follows we want to use inductive definitions together with four-valued information about the open predicates (for example, information obtained through propagation on a first order theory). Therefore, we now recall (see, e.g., Denecker & Ternovska, 2008) how to define the well-founded model of a *non-monotone* inductive definition Δ (that is, negation in the body of rules is allowed), given some *four-valued information* $\mathcal{O}$ about its open predicates, which we denote by $WFM_{\mathcal{O}}(\Delta)$. In order to do this, we first need to define some additional concepts. Recall that φ_P denotes the body of the unique rule with predicate P in the head.

Definition 5.1 (Operator $\mathcal{T}_\Delta^{\mathcal{O}}$)**.** For a definition Δ and a given (potentially 4-valued) $Open(\Delta)$-interpretation $\mathcal{O}$, we define the operator $\mathcal{T}_\Delta^{\mathcal{O}}$ on 4-valued $Def(\Delta)$-interpretations with the same domain as $Open(\Delta)$ such that $\mathcal{T}_\Delta^{\mathcal{O}}(\mathcal{I}) = \mathcal{J}$ iff for each defined predicate P/n and n-tuple $\bar{d} \in D^n$, it holds that

$$P^{\mathcal{J}}(\bar{d}) = \varphi_P^{\mathcal{O}+\mathcal{I}}[\bar{d}]$$

Recall that in the preliminaries we defined the isomorphism τ that maps a pair of interpretations (I, J) to the corresponding four-valued interpretation $\mathcal{I}$.

Definition 5.2 ($WFM_{\mathcal{O}}(\Delta)$)**.** We define the well-founded model of Δ in $\mathcal{O}$ as the 4-valued interpretation $\tau(I, J)$ such that (I, J) is the maximal oscillation pair of the operator $ST_\Delta^{\mathcal{O}}$, where $ST_\Delta^{\mathcal{O}}$ is the operator $\lambda J(lfp(\lambda K(\mathcal{T}_\Delta^{\mathcal{O}}(K, J)_1))$. I.e., (I, J) is the least precise pair of 2-valued interpretations such that

$$I = ST_\Delta^{\mathcal{O}}(J) \text{ and } J = ST_\Delta^{\mathcal{O}}(I).$$

Some explanation is in order. First look at the operator $\lambda K(\mathcal{T}_\Delta^{\mathcal{O}}(K, J)_1)$. This operator takes a $Def(\Delta)$-structure K, turns it into a 4-valued one by combining it with J, applies the operator $\mathcal{T}_\Delta^{\mathcal{O}}$, and projects the result on its first argument. We can see that $\lambda K(\mathcal{T}_\Delta^{\mathcal{O}}(K, J)_1) = L$ iff for each defined predicate P/n and n-tuple $\bar{d} \in D^n$, it holds that

$$P^L(\bar{d}) = \mathbf{t} \text{ iff } (\varphi_P[\bar{d}])^{+(O_1+L)-(O_2+J)} = \mathbf{t}$$

In other words, positive occurrences of atoms are evaluated in $O_1 + L$, negative occurrences in $O_2 + J$. For each J, this operator $\lambda K(\mathcal{T}_\Delta^{\mathcal{O}}(K, J)_1)$ is monotone, and therefore has a least fixpoint. The operator $ST_\Delta^{\mathcal{O}}$ now maps J to this least fixpoint. It can be proven that this operator is antimonotone, and therefore has a maximal oscillation pair. The definition of the well-founded model as the maximal oscillation pair of the operator $ST_\Delta^{\mathcal{O}}$ is a non-constructive definition. This maximal oscillation pair can be *constructed* by iterating the following operator, starting from the least precise interpretation $\mathcal{I}$ that extends $\mathcal{O}$, until it reaches its least fixpoint.

Definition 5.3 (Stable operator $\mathcal{ST}_\Delta^{\mathcal{O}}$)**.** We define the operator $\mathcal{ST}_\Delta^{\mathcal{O}}$ on pairs of interpretations as:

$$\mathcal{ST}_\Delta^{\mathcal{O}}(I, J) = (ST_\Delta^{\mathcal{O}}(J), ST_\Delta^{\mathcal{O}}(I)).$$

The stable operator is monotone w.r.t. the precision order $\leq_p$ on pairs of interpretations and its fixpoints therefore form a complete lattice. The fixpoints of this operator are called the *four-valued stable fixpoints* of Δ, and the least precise of these fixpoints is precisely the well-founded model of Δ given $\mathcal{O}$.

We can now define the semantics of inductive definitions in the general case. The reader can easily verify that this indeed generalizes the definition of $Mod_O(\Delta)$ for positive definitions we gave in Section 2.2.

Definition 5.4 (Satisfaction relation for definitions)**.** $I \models \Delta$ iff $(I|_{Def(\Delta)}, I|_{Def(\Delta)})$ is the well-founded model of Δ in $I|_{Open(\Delta)}$.

Note that when the definition Δ has a three-valued well-founded model for every possible interpretation of the open predicates $Open(\Delta)$, the definition has no model (i.e. there does not exists an interpretation I such that $I \models \Delta$). We call a definition *total* if it has a two-valued well-founded model for every possible two-valued interpretation of its open predicates.

The above definitions generalize in a rather obvious way the standard well-founded semantics of propositional logic programs and are strongly linked to the stable semantics (Gelfond & Lifschitz, 1988). In the case of a propositional logic program Δ, where $Open(\Delta) = \{\}$, the operator $\lambda K(\mathcal{T}_\Delta(K, J)_1)$ is nothing else than the immediate consequence operator T_{Δ_J} of the Gelfond Lifschitz reduct Δ_J of Δ, and the operator that maps J to $lfp(\lambda K(\mathcal{T}_\Delta(K, J)_1)$ is the stable operator of Δ. As shown by Van Gelder (1993), its maximal oscillation pair is indeed the well-founded model of Δ.

5.2 Approximating $\exists\forall SO(ID)$-Satisfiability Problems

Assume we have a formula $\exists\forall\psi$, where ψ is an FO(ID) formula instead of an FO. The concepts of witness (Definition 4.1) and sound approximation (Definition 4.2) can straightforwardly be generalised for ψ being a FO(ID) formula. This allows us to develop two approaches. The first one, in Section 5.2.1, replaces the definition by its completion and then applies the method of Section 4. However, the completion can be weaker than the definition. Therefore, in Section 5.2.2, we develop another approach that constructs an approximation for the conjunction of a definition with a FO formula.

5.2.1 Using the Completion of a Definition

Our first approach is based on the use of the completion (Clark, 1978). The completion of a definition Δ is the conjunction of the equivalences $\forall \bar{x}\, P(\bar{x}) \Leftrightarrow \varphi_P(\bar{x})$ for all predicates $P \in Def(\Delta)$, where $\varphi_P(\bar{x})$ is the body of the unique rule with P in its head. A useful property is that each definition Δ implies its completion $compl(\Delta)$. Moreover, if Δ is non-recursive, then the two are actually equivalent. Replacing the definition by its completion in $(\Delta \wedge \psi_1) \Rightarrow \psi_2$ we obtain the formula $(comp(\Delta) \wedge \psi_1) \Rightarrow \psi_2$. As every model of Δ is a model of $comp(\Delta)$, every model of $(comp(\Delta) \wedge \psi_1) \Rightarrow \psi_2$ is a model of $(\Delta \wedge \psi_1) \Rightarrow \psi_2$ and every witness of the $\exists\forall\,(compl(\Delta) \wedge \psi_1) \Rightarrow \psi_2$ satisfiability problem is a witness of the $\exists\forall\,(\Delta \wedge \psi_1) \Rightarrow \psi_2$ satisfiability problem. Hence we can use the results of Section 4 and formulate the following proposition.

Proposition 5.1. *The formula* $\mathcal{APP}^{\Rightarrow}_{BU}((compl(\Delta) \wedge \psi_1) \Rightarrow \psi_2)$ *is a sound approximation of* $\exists\bar{P}\forall\bar{Q}(\Delta \wedge \psi_1) \Rightarrow \psi_2$.

The disadvantage of using the completion is that no matter how complete the approximation method defined in Definition 4.4 is, it will never be able to infer something that follows from Δ but not from $compl(\Delta)$. For instance, the inductive definition $\{P \leftarrow P\}$ entails $\neg P$, but its completion $P \Leftrightarrow P$ does not.

Denecker and Ternovska (2008) have proven that, in addition to non-recursive definitions, a class of recursive definitions *are* equivalent to their completion. In particular, this is the case for definitions over a strict *well-founded order* $\prec$[2]. We can therefore replace those definitions by their completion without losing precision. The theory T_{act} in Example 1.1 is actually the completion of the definition Δ_{act}. Since Δ_{act} is a recursive definition over a strict well-founded order (we can make use of the time argument in the predicates to construct such a well-founded order), Δ_{act} and T_{act} are equivalent.

The Gaspipe conformant planning problem (Son et al., 2005), on the other hand, uses a dynamic domain for which the completion does not suffice. Summarized, the objective of this conformant planning problem is to start a flame in a burner which is connected to a gas tank through a pipe line. The pipe line consists of sections connected to each other with valves. When a valve is opened with gas on one side and not on the other, the gas will spread as far as possible. This can be formalized by an inductive definition of the reachability relation on the pipe line:

$$\left\{\begin{array}{l}\forall x,t\ Gas(x,t) \leftarrow \exists y\ Gas(y,t) \wedge \exists v\ Connected(x,y,v) \wedge Open(v,t).\\ \forall x,t\ Gas(x,t) \leftarrow Tank(x).\end{array}\right\}$$

Such reachability definitions are not equivalent to their completion. Therefore, the approximative method presented in this subsection will not work. The problem with the completion in this case is that it does not correctly minimize the defined predicates in the presence of recursion, which would allow models in which a loop in the pipe line is filled with gas even when it is not connected to a tank. What is missing, therefore, is the *unfounded set* reasoning that allows the well-founded semantics to correctly minimize the defined predicates.

5.2.2 Using a Certainly True/Possibly True Approximation

The approximative definition $Approx_\sigma(\psi)$ used in Section 4 has the nice property that it defines, for each subformula of ψ (including ψ itself), whether it is certainly true or certainly false. It was this property that allowed us to find witnesses by simply asserting that A^{ct}_ψ had to hold according to this definition. If we want to apply the same method to formulas ψ that contain a definition Δ, we have to construct an approximative definition that defines whether each of the subformulas of Δ (including Δ itself) is certainly true or certainly false. In Section 5.2.1, our naive method managed to do this by simply replacing Δ by its completion. We now want to improve on this method by constructing an approximation that also takes into account the unfounded set reasoning that is performed by the well-founded semantics.

Once we also take this aspect of the well-founded semantics into account, however, it becomes difficult to define when a definition as a whole is certainly true or certainly false. Luckily, this is not needed if we stick to our assumption that definitions appear only in the antecedent of the implication ψ. Indeed, because we approximate implications by

2. An order $<$ is well-founded if there are no infinite descending chains $\ldots < x_n < x_{n-1} < \ldots < x_1$

assuming that their antecedent is certainly true (Definition 4.4), all that we really need is an approximation of the consequences of a definition. To this end, we will transform the original definition Δ into an approximative definition Δ' such that the well-founded model of Δ', given an approximation $\mathcal{O}$ of the open predicates of Δ, approximates each of the well-founded models of Δ given an interpretation O for the open predicates that is approximated by $\mathcal{O}$. In other words, we construct an approximative definition Δ' whose two-valued well-founded model encodes the potentially four-valued well-founded model of the original definition Δ, given a potentially four-valued interpretation for the predicates of Δ. We therefore again represent a four-valued interpretation of the orginal vocabulary Σ of Δ by a two-valued interpretation of a larger vocabulary Σ'. However, instead of introducing, for each predicate P of Σ, a predicate P^{ct} (P is certainly true) and P^{cf} (P is certainly false), as we did before, we now introduce predicates P^{ct} and P^{pt} (P is possibly true, i.e., P is *not* certainly false). Let $\Sigma^{ct/pt}$ denote this vocabulary $\Sigma^F \cup \{P^{ct} \mid P \in \Sigma\} \cup \{P^{pt} \mid P \in \Sigma\}$. For a four-valued Σ-interpretation $\mathcal{I}$, we define the corresponding $\Sigma^{ct/pt}$-interpretation $\mathcal{I}^{ct/pt}$ as the interpretation with the same pre-interpretation as $\mathcal{I}$ such that $(P^{ct})^{\mathcal{I}^{ct/pt}} = \{\bar{d} \mid P^{\mathcal{I}}(\bar{d}) \geq_p \mathbf{t}\}$ and $(P^{pt})^{\mathcal{I}^{ct/pt}} = \{\bar{d} \mid (P^{\mathcal{I}}(\bar{d}) \leq_p \mathbf{t})\}$.

Also, for a Σ-formula φ, we define the formula $\varphi^{ct/pt}$ as the formula that we obtain after replacing all positive occurrences of a predicate P in φ by P^{ct}, and all negative occurrences by P^{pt}, and finally reducing to negational normal form. It is easy to see that $\varphi^{ct/pt}$ can also be obtained from φ^{ct} by replacing, for every predicate P, all occurrences of P^{cf} by $\neg P^{pt}$. Unlike φ^{ct}, $\varphi^{ct/pt}$ is not always a positive formula as it can contain negations. In particular, P^{ct} occurs only positively while P^{pt} occurs only negatively. For a subvocabulary $\sigma \subseteq \Sigma$, $\varphi^{ct/pt}_\sigma$ again denotes $\varphi^{ct/pt}$ but with both P^{ct} and P^{pt} replaced by P for every predicate $P \in \sigma$. Again, in what follows we will use σ to denote predicates that do not need to be approximated because we have two-valued information about them.

Definition 5.5 ($App^{ct/pt}_\sigma(\Delta)$)**.** For a definition Δ, we define $App^{ct/pt}_\sigma(\Delta)$ as the definition $\{\mathcal{R}^{ct} \cup \mathcal{R}^{pt}\}$ where $\mathcal{R}^{ct}$ consists of the rules

$$\forall \bar{x}(P^{ct}(\bar{x}) \leftarrow \varphi^{ct/pt}_\sigma)$$

and $\mathcal{R}^{pt}$ consists of the rules

$$\forall \bar{x}(P^{pt}(\bar{x}) \leftarrow \neg(\neg\varphi)^{ct/pt}_\sigma)$$

for every definitional rule $\forall \bar{x}\ (P(\bar{x}) \leftarrow \varphi)$ in Δ.

We again assume for the rest of this paragraph without loss of generality that σ is empty, and we drop it in the notation of $App^{ct/pt}_\sigma$.

Example 5.1. Consider the following inductive definition.

$$\left\{ \begin{array}{lcl} B & \leftarrow & B \vee \neg A \\ A & \leftarrow & \neg D \end{array} \right\}$$

Assume $\sigma = \{\}$. Then

$$App^{ct/pt} = \left\{ \begin{array}{lcl} B^{ct} & \leftarrow & B^{ct} \vee \neg A^{pt} \\ A^{ct} & \leftarrow & \neg D^{pt} \\ B^{pt} & \leftarrow & B^{pt} \vee \neg A^{ct} \\ A^{pt} & \leftarrow & \neg D^{ct} \end{array} \right\}.$$

We see that for a three-valued interpretation $\{D = \mathbf{u}\}$, which translates to $\{D^{ct} = \mathbf{f}, D^{pt} = \mathbf{t}\}$, the approximative definition will correctly infer that B^{ct} is false and B^{pt} is true. If we take $\{D = \mathbf{f}\}$ as an interpretation for the open predicate D, we see that the approximative definition correctly infers that both B^{ct} and B^{pt} are false. This is an example of *unfounded set* reasoning: once A is known to be true, the approximation detects that B could only be derived from B itself and therefore must be false. This kind of reasoning could not be done by our previous, completion-based approximation method, since it is only sound w.r.t. the semantics of the definition itself, and not w.r.t. its weaker completion.

This example also demonstrates why we have to use the vocabulary $\Sigma^{ct/pt}$ instead of $\Sigma^{ct/cf}$, since the latter would have yielded a definition:

$$\left\{ \begin{array}{rcl} B^{ct} & \leftarrow & B^{ct} \vee A^{cf} \\ A^{ct} & \leftarrow & D^{cf} \\ B^{cf} & \leftarrow & B^{cf} \wedge A^{ct} \\ A^{cf} & \leftarrow & D^{ct} \end{array} \right\}.$$

For $\{D = \mathbf{f}\}$, this definition would fail to infer B^{cf}. Intuitively, the reason for this is that the unfounded set reasoning of the well-founded semantics tries to minimize the extension of the defined predicates by making as many atoms false as possible. Using the $\Sigma^{ct/cf}$ vocabulary, the well-founded semantics of the approximating definition therefore attempts to falsify as many atoms $P^{cf}(\bar{d})$ as possible, which actually corresponds to *maximizing* the possible extension of the original predicates P, instead of minimizing it as the well-founded semantics of the original definition does.

Each two-valued interpretation of the "double" vocabulary $\Sigma^{ct/pt}$ corresponds to a four-valued interpretation of the original vocabulary Σ. We again want to establish a link between the well-founded model of the original definition Δ and that of $App_\sigma^{ct/pt}$. A complicating factor here is that, as the above example shows, the definition $App_\sigma^{ct/pt}$ is no longer monotone and therefore it is no longer guaranteed to have a two-valued well-founded model. Because a three-valued interpretation of $\Sigma^{ct/pt}$ no longer corresponds to even a four-valued interpretation of the original vocabulary Σ, we can only prove such a correspondence if the well-founded model of $App_\sigma^{ct/pt}$ is two-valued.

Theorem 5.2. *Let $\mathcal{O}$ be a four-valued interpretation for the open predicates of a definition Δ. $App^{ct/pt}(\Delta)$ has a two-valued well-founded model given $\mathcal{O}^{ct/pt}$ if and only if Δ has a unique four-valued stable fixpoint given $\mathcal{O}$. Moreover, if $App^{ct/pt}(\Delta)$ has a two-valued well-founded model I, then the unique four-valued stable fixpoint of Δ is the unique interpretation $\mathcal{I}$ for which $\mathcal{I}^{ct/pt} = I$.*

Proof. See Appendix D. □

This theorem requires that Δ has a unique four-valued stable model for each four-valued input interpretation $\mathcal{O}$. This is a stronger requirement than the more common condition of *totality*, which only requires a definition to have a two-valued well-founded model given a *two-valued* input interpretation O. As the following example shows, this stronger condition is indeed necessary.

Example 5.2. Consider the following definition:

$$\left\{\begin{array}{l} A \leftarrow \neg B. \\ B \leftarrow \neg A \wedge C. \\ C \leftarrow O \wedge \neg O. \end{array}\right\}$$

This definition is total, because, for each two-valued interpretation for the open predicate O, it has $(\{A\},\{A\})$ as its two-valued well-founded model. However, for the three-valued interpretation $(\{\},\{O\})$ (i.e., "O is unknown") for the open predicate O, the three-valued well-founded model $(\{\},\{O,A,B,C\})$ is *not* the unique three-valued stable fixpoint, since $(\{A\},\{O,A,C\})$ is also such a fixpoint. And indeed, we find that

$$App^{ct/pt}(\Delta) = \left\{\begin{array}{l} A^{ct} \leftarrow \neg B^{pt}. \\ B^{ct} \leftarrow \neg A^{pt} \wedge C^{ct}. \\ C^{ct} \leftarrow O^{ct} \wedge \neg O^{pt}. \\ A^{pt} \leftarrow \neg B^{ct}. \\ B^{pt} \leftarrow \neg A^{ct} \wedge C^{pt}. \\ C^{pt} \leftarrow O^{pt} \wedge \neg O^{ct}. \end{array}\right\}$$

does not have a two-valued well-founded model given $\{O^{pt}\}$. The easiest way to see this is to fill in the fact that we know that O^{pt} is $\mathbf{t}$ and O^{ct} is $\mathbf{f}$ and propagate this information:

$$\left\{\begin{array}{l} A^{ct} \leftarrow \neg B^{pt}. \\ B^{ct} \leftarrow \neg A^{pt} \wedge C^{ct}. \\ C^{ct} \leftarrow \mathbf{f} \wedge \neg \mathbf{t}. \\ A^{pt} \leftarrow \neg B^{ct}. \\ B^{pt} \leftarrow \neg A^{ct} \wedge C^{pt}. \\ C^{pt} \leftarrow \mathbf{t} \wedge \neg \mathbf{f}. \end{array}\right\} \rightsquigarrow \left\{\begin{array}{l} A^{ct} \leftarrow \neg B^{pt}. \\ B^{ct} \leftarrow \neg A^{pt} \wedge \mathbf{f}. \\ \\ A^{pt} \leftarrow \neg B^{ct}. \\ B^{pt} \leftarrow \neg A^{ct} \wedge \mathbf{t}. \\ C^{pt} \leftarrow \end{array}\right\} \rightsquigarrow \left\{\begin{array}{l} A^{ct} \leftarrow \neg B^{pt}. \\ \\ \\ A^{pt} \leftarrow \neg \mathbf{f}. \\ B^{pt} \leftarrow \neg A^{ct}. \\ C^{pt} \leftarrow \end{array}\right\} \rightsquigarrow \left\{\begin{array}{l} A^{ct} \leftarrow \neg B^{pt}. \\ \\ \\ A^{pt} \leftarrow \\ B^{pt} \leftarrow \neg A^{ct}. \\ C^{pt} \leftarrow \end{array}\right\}$$

So, we are left with a loop over negation, which means that A^{ct} and B^{pt} will remain unknown in the three-valued well-founded model $(\{A^{pt}, C^{pt}\}, \{A^{ct}, A^{pt}, C^{pt}, B^{pt}\})$ of the definition.

By computing the three-valued well-founded model of Δ given $\mathcal{O}$, the approximative definition $App^{ct/pt}(\Delta)$ can produce more precise results than the approximation of $compl(\Delta)$; in particular it can detect that atoms in an unfounded set must be false, as illustrated in Example 5.2 above. To use $App^{ct/pt}(\Delta)$ in an approximation of an $\exists\forall(\Delta\wedge\varphi) \Rightarrow \psi_2$-problem, we still need to show how it can be combined with the approximation $Approx_\sigma(\varphi)$ of φ to produce a sound approximation of $\Delta\wedge\varphi$. To do this, we need to combine one definition of ct/cf-predicates with another definition of ct/pt-predicates. We achieve this by first merging the two definitions and then adding rules that copy information from the one vocabulary to the other.

Definition 5.6 ($\mathcal{D}_\sigma^{\Delta\wedge\varphi}$)**.** Given a vocabulary Σ, a subvocabulary $\sigma \subset \Sigma$, an inductive definition Δ and first-order formula φ. We define $\mathcal{D}_\sigma^{\Delta\wedge\varphi}$ as the following inductive definition.

$$\begin{gathered} App_\sigma^{ct/pt}(\Delta) \\ \cup \\ Approx_\sigma(\varphi) \cup \{A_\varphi^{ct} \leftarrow \mathbf{t}\} \\ \cup \\ \{O^{pt} \leftarrow \neg O^{cf} \mid \text{for every predicate } O \in Open(\Delta) \setminus \sigma\} \\ \cup \\ \{P^{cf} \leftarrow \neg P^{pt} \mid \text{for every predicate } P \in Def(\Delta) \setminus \sigma\} \\ \cup \\ \{O^{cf} \leftarrow \mathbf{f}, O^{ct} \leftarrow \mathbf{f} \mid \text{for every predicate } O \in Open(\Delta) \setminus \sigma \text{ that does not occur in } \varphi\} \end{gathered}$$

This definition indeed contains both the rules from the approximation of Δ and the rules from the approximation of φ, that is, $App^{ct/pt}(\Delta)$ and $Approx(\varphi) \cup \{A_\varphi^{ct} \leftarrow \mathbf{t}\}$, respectively, but also a number of extra rules that make a connection between these two approximations. To approximate a defined predicate Q the approximation of Δ uses the pair of predicates Q^{ct} and Q^{pt} while the approximation of φ uses Q^{ct} and Q^{cf}. Hence, a number of extra rules are needed to transfer information between the predicates Q^{pt} and Q^{cf}. The rules $\{O^{pt} \leftarrow \neg O^{cf}\}$ transfer information that the approximation of φ has derived about the truth of an open predicate O (by means of O^{cf}) to the corresponding predicate O^{pt} of the approximation of the definition. The rules $\{P^{cf} \leftarrow \neg P^{pt}\}$ in turn propagate information derived about the truth of a defined predicate in the approximation of the definition to the corresponding predicate P^{cf} of the approximation of φ. Finally, the rules $\{O^{cf} \leftarrow \mathbf{f}, O^{ct} \leftarrow \mathbf{f}\}$ make sure that O^{cf} and O^{ct} are defined atoms (instead of open ones) and that their default value is $\mathbf{u}$. The following proposition relates the well founded model of $\mathcal{D}_\sigma^{\Delta\wedge\varphi}$ with the models of $\Delta \wedge \varphi$.

Proposition 5.3. *Given a vocabulary Σ, a subvocabulary σ and an FO(ID) formula $\Delta \wedge \varphi$. Then, for every σ-interpretation I, if $WFM_I(\mathcal{D}_\sigma^{\Delta\wedge\varphi}) \models P^{ct}(\bar{d})$ (resp. $P^{cf}(\bar{d})$), then it holds for every model M of $\Delta \wedge \varphi$ extending I that $M \models P(\bar{d})$ (resp. $M \models \neg P(\bar{d})$).*

Proof. See Appendix E. $\square$

This proposition is the analogue for inductive definitions of the result that Theorem 3.2 states for FO formulas. One difference between the two results is that the above proposition always assumes that the definition Δ itself holds, while Theorem 3.2 makes no such assumption about the FO formula ψ that is approximated. As discussed in the beginning of this section, this restriction is not a problem, because of the way in which we approximate implications and because we only allow definitions to appear in the antecedent. A second difference is that Theorem 3.2 applies to arbitrary subformulas φ of ψ, while the above proposition only considers atoms $P(\bar{d})$. It is, however, an easy corrolary of this proposition that the result in fact holds for each formula ψ that contains only predicates defined by $\mathcal{D}^{\Delta\wedge\varphi}$, i.e., whenever $WFM_I(\mathcal{D}_\sigma^{\Delta\wedge\varphi}) \models \psi^{ct}(\bar{d})$ (or $\psi^{cf}(\bar{d})$), then for every model M of $\Delta \wedge \varphi$ extending I, it holds that $M \models \psi(\bar{d})$ (resp. $M \models \neg\psi(\bar{d})$).

We introduced the approximation $App^{ct/pt}(\Delta)$ with the aim of being more complete than the completion-based approximation. As long as only a single definition Δ was considered in isolation, we succeeded in this goal. However, now that we have also incorporated the additional formula φ, this is no longer the case. For instance, consider the following FO(ID) theory:

$$\left\{\ Q \leftarrow P\ \right\} \wedge Q.$$

Here, the approximation $\mathcal{D}^{\Delta\wedge\varphi}$ cannot derive that P is certainly true, simply because the ct/pt-approximation $App^{ct/pt}(\Delta)$ does not contain rules for *head-to-body* propagation, i.e., there are no rules to infer something about the body of a definitional rule, given information about its head. By contrast, the approximation of the completion does contain such rules and therefore has no problems reaching this conclusion. This motivates us to not just use $\mathcal{D}^{\Delta\wedge\varphi}$ but to use $\mathcal{D}^{\Delta\wedge(compl(\Delta)\wedge\varphi)}$ instead. Because each definition implies its completion, this is sound.

To obtain a sound approximation for an $\exists\forall SO(ID)$ formula $\exists\bar{P}\forall\bar{Q}\,\psi$ with $\psi = (\Delta \wedge \varphi) \Rightarrow \psi_2$, we now just need to plug in our approximation $\mathcal{D}^{\Delta\wedge(compl(\Delta)\wedge\varphi)}$ for Δ into a suitable $\exists SO(ID)$ formula, similar to the one we defined in Definition 4.4 for an $\exists\forall SO$ formula $\exists\bar{P}\forall\bar{Q}\,\psi_1 \Rightarrow \psi_2$. A small complication, however, is that, as discussed previously, our approximation of a definition does not define predicates A^{ct}_{Δ} and A^{cf}_{Δ} that tell us when the definition Δ as a whole is certainly true or certainly false. Therefore, we can no longer use our normal approximation of when the entire implication ψ is certainly true. Before we present the approximation for $\exists\forall SO(ID)$, let us first introduce a reformulation of our original approximation for $\exists\forall SO$, that avoids the use of this A^{ct}_{ψ}.

Proposition 5.4. *For an $\exists\forall SO$ formula $F = \exists\bar{P}\forall\bar{Q} : \psi$, where ψ is of the form $\psi_1 \Rightarrow \psi_2$, the approximation defined in Definition 4.4, i.e., the formula*

$$\exists\bar{P}\exists\bar{R} : (Approx_\sigma(\psi_1 \Rightarrow \psi_2) \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}) \wedge A^{ct}_{\psi}$$

is equivalent to

$$\exists\bar{P}\exists\bar{R} : (Approx_\sigma(\psi_1) \cup Appox_\sigma(\psi_2) \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}) \wedge (A^{cf}_{\psi_1} \vee A^{ct}_{\psi_2})$$

Proof. This is obvious from the fact that the difference between $Approx_\sigma(\psi_1 \Rightarrow \psi_2)$ and $Approx_\sigma(\psi_1) \cup Appox_\sigma(\psi_2)$ is precisely a set of rules that ensure that A^{ct}_{ψ} is equivalent to $A^{cf}_{\psi_1} \vee A^{ct}_{\psi_2}$. ∎

Our approximation for $\exists\forall SO(ID)$ now essentially consists of just replacing $Approx_\sigma(\psi_1) \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}$ and $A^{cf}_{\psi_1}$ respectively by $\mathcal{D}^{\Delta\wedge compl(\Delta)\wedge\varphi}_{\sigma}$ and A^{cf}_{φ} in the above form.

Proposition 5.5. *Given an $\exists\forall SO(ID)$ formula $F = \exists\bar{P}\forall\bar{Q}\,(\Delta \wedge \varphi) \Rightarrow \psi_2$. We define $\mathcal{APP}^{wf}(F)$ as the following $\exists SO(ID)$ formula.*

$$\exists\bar{P}\bar{R} :\ (\mathcal{D}^{\Delta\wedge compl(\Delta)\wedge\varphi}_{\sigma} \cup Approx_\sigma(\psi_2)) \wedge (A^{cf}_{\varphi} \vee A^{ct}_{\psi_2}).$$

Then $\mathcal{APP}^{wf}(F)$ is a sound approximation of F.

In some cases, the approximating $\mathcal{D}^{\Delta\wedge compl(\Delta)\wedge\varphi}_{\sigma}$ instead of $\mathcal{D}^{\Delta\wedge\varphi}_{\sigma}$ will not gain us anything. For instance, consider an $\exists\bar{P}\forall\bar{Q}(\Delta \wedge \varphi) \Rightarrow \psi_2$ problem, where φ only contains *open predicates* of Δ (as is the case for the conformant planning problems we consider in the next sections). In this case, we will never need head-to-body propagation, and therefore $\mathcal{D}_\sigma^{\Delta\wedge\varphi}$ is just as complete as $\mathcal{D}_\sigma^{\Delta\wedge compl(\Delta)\wedge\varphi}$, and we are therefore better off using the former.

As was the case for our approximation method for $\exists\forall SO$, here as well not all rules from the definition in $\mathcal{APP}^{wf}$ are necessary. Indeed, only the bottom-up rules from $Approx_\sigma(\psi_2)$ are needed, and can be unfolded into a single rule. Therefore, below we define two more variants of Definition 5.5.

Definition 5.7 ($\mathcal{APP}^{wf}_{BU}(F)$)**.** Given an $\exists\forall SO(ID)$ formula $F = \exists\bar{P}\forall\bar{Q}\,(\Delta\wedge\varphi)\Rightarrow\psi_2$.

We define $\mathcal{APP}^{wf}_{BU}(F)$ as the following $\exists SO(ID)$ formula,

$$\exists\bar{P}\bar{R}:\ (\mathcal{D}_\sigma^{\Delta\wedge compl(\Delta)\wedge\varphi}\cup Approx_\sigma^{BU}(\psi_2))\wedge(A_\varphi^{cf}\vee A_{\psi_2}^{ct}),$$

and we define $\mathcal{APP}^{wf}_{BU,Unf}(F)$ as the following $\exists SO(ID)$ formula,

$$\exists\bar{P}\bar{R}:\ (\mathcal{D}_\sigma^{\Delta\wedge compl(\Delta)\wedge\varphi}\cup\{A_{\psi_2}^{ct}\leftarrow(\psi_2)^{ct}\})\wedge(A_\varphi^{cf}\vee A_{\psi_2}^{ct}).$$

6. Experimental Evaluation

In this paper we have seen a number of methods to approximate $\exists\forall SO$ and $\exists\forall SO(ID)$ satisfiability problems. In this subsection, we explore, through a number of experiments, how we can use these methods to solve practically useful problems as fast as possible. We performed these experiments on a number of conformant planning benchmarks from the paper of Son et al. (2005). As we show in Section 7, all these benchmarks are of the form $F = \exists\forall\Delta\Rightarrow\psi$, where Δ is a stratified definition, and is therefore equivalent to its completion. Therefore, F is equivalent to the $\exists\forall SO$ formula $\exists\forall\, compl(\Delta)\Rightarrow\psi$, which we denote by F^{cp}. All experiments were run on a dual core 2.4 Ghz CPU, 2.8 Gb RAM Linux machine, using the IDP model expansion system for FO(ID) (Mariën et al., 2006). A time-out of twenty minutes was used.

A first question we want to answer is whether for these definitions, the completion based approximation is faster than the ct/pt approximation. It is not hard to see that, even though $Approx(compl(\Delta))$ is linear in the size of the parse tree of $compl(\Delta)$, this definition may contain more rules than $App^{ct/pt}(\Delta)$, and moreover, these rules may contain a lot of recursion. This can pose a challenge for current solvers, and suggests that it is likely to be more efficient to use the ct/pt approximation for definitions. The first column in Table 2 shows times for using the completion of the definition Δ, that is, $\mathcal{APP}^{\Rightarrow}(F^{cp})$, while for the second column, the ct/pt-approximation of Δ was used, that is, $\mathcal{APP}^{wf}(F)$. As expected, the ct/pt-approximation was consistently faster.

Table 2 also compares solving times of the full completion-based approximative definition (in the first column) with the approximation $\mathcal{APP}^{\Rightarrow}_{BU}(F^{cp})$ (Def. 4.5), from which the top-down propagation rules for ψ have been removed (third column). We see that in the BT and BTC benchmarks we get an order of magnitude improvement. The fourth column of Table 2 shows timings for the unfolded approximation of ψ_2, $\mathcal{APP}^{\Rightarrow}_{BU,Unf}(F^{cp})$, in which the intermediate Tseitin predicates have been removed (Def. 4.6). We see that this unfolding consistently provides a speed-up.

These results suggest that combining the above techniques, that is, using the ct/pt approximation for Δ and the unfolding of the bottom-up approximation of ψ_2 together, will

Problem	$\mathcal{APP}^{\Rightarrow}(F^{cp})$	$\mathcal{APP}^{wf}(F)$	$\mathcal{APP}^{\Rightarrow}_{BU}(F^{cp})$	$\mathcal{APP}^{\Rightarrow}_{BU,Unf}(F^{cp})$	$\mathcal{APP}^{wf}_{BU,Unf}(F)$
BT(2,2)	0,151	0,109	0,115	0,065	0,031
BT(4,2)	3,404	3,493	0,312	0,153	0,064
BT(6,2)	38,93	14,76	0,876	0,409	0,113
BT(8,4)	-	-	32,91	1,774	0,462
BT(10,4)	-	-	-	-	1,643
BTC(2,2)	0,210	0,131	0,171	0,116	0,037
BTC(4,2)	-	-	40,081	8,408	0,109
BTC(6,2)	-	-	-	-	0,335
BTC(8,4)	-	-	-	-	41,894
Domino(100)	0,390	0,026	0,507	0,473	0,049
Domino(200)	1,101	0,036	1,250	1,266	0,052
Domino(500)	6,597	0,067	8,995	6,997	0,128
Domino(1000)	31,275	0,120	42,583	28,387	0,396
Domino(2000)	-	0,231	-	-	1,530
Ring(2)	7,023	0,374	5,217	2,792	0,100
Ring(4)	-	-	-	-	0,358
Ring(6)	-	-	-	-	6,650
Ring(8)	-	-	-	-	193,290
Ring(10)	-	-	-	-	2485

Table 2: The first column gives the name of the benchmark and the other ones different execution times. The second column gives the execution time for the approximation of the completion and the third for the cp/pt approximation. The fourth and fifth column use variants of the completion approximation. For the fourth column, the top-down rules for ψ_2 are removed while in addition, for the fifth column, the remaining bottom-up rules are unfolded. The last column combines the cp/pt approximation with both other changes. "-" means the execution was interrupted after 20 minutes.

give us the fastest way of approximating $\exists\forall(\Delta \wedge \psi_1) \Rightarrow \psi_2$ satisfiability problems. Indeed, this is what formula $\mathcal{APP}^{wf}_{BU,Unf}(F)$ (Definition 5.7) does, and the results of this method are shown in the last column of Table 2. As expected, this is by far the fastest method.

7. Applications and Related Work

In the literature, many examples can be found of approaches that perform some kind of approximate reasoning about the models of a logical theory. Often, these approaches, which are specific to the problem at hand, seem to boil down to an instantiation of the general methods presented here. In this section we give some examples.

7.1 Conformant Planning

In general, a *conformant planning problem* is a planning problem in a non-deterministic domain where the initial state may be not fully known. The goal is to come up with a plan (i.e., a sequence of actions) that is nevertheless guaranteed to work. This is a hard problem: the decision problem of deciding whether a conformant plan with a fixed length k exists is Σ_2^P-complete[3] (Baral, Kreinovich, & Trejo, 2000; Turner, 2002). Therefore, one

3. For planning domains where the executability of actions in a given state cannot be determined polynomially, this is even Σ_3^P (Turner, 2002)

typically attempts to solve it approximately. In this section, we show how we can apply our approximative methods to solve conformant planning problems.

Example 7.1. Let us consider the Clogged Bombs in the Toilet domain (McDermott, 1987; Son et al., 2005). There are a number of packages and a toilet. Each of the packages may contain a bomb which can be disarmed by dunking the package in the toilet. Dunking a package into a toilet also clogs the toilet and we cannot throw a package in a clogged toilet. Flushing the toilet unclogs it. The effects of the actions on the fluents are modeled by the following definition Δ_{act}, and the preconditions by the conjunction ψ_{prec} of sentences in T_{prec}.

$$\Delta_{act} = \left\{ \begin{array}{r@{\;}l} Clogged(0) & \leftarrow Init\_Clogged. \\ Clogged(t+1) & \leftarrow \exists p : Dunk(p,t) \vee (Clogged(t) \wedge \neg Flush(t)). \\ Armed(p,0) & \leftarrow Init\_Armed(p). \\ Armed(p,t+1) & \leftarrow Armed(p,t) \wedge \neg Dunk(p,t). \end{array} \right\}$$

$$T_{prec} = \begin{array}{l} \forall p\, t : Dunk(p,t) \Rightarrow \neg Clogged(t). \\ \neg(\exists p\, t : Dunk(p,t) \wedge Flush(t)). \\ \forall p\, p_2\, t : Dunk(p,t) \wedge Dunk(p_2,t) \Rightarrow p = p_2. \\ \forall p\, t\, t_2 : Dunk(p,t) \wedge Dunk(p,t_2) \Rightarrow t = t_2. \end{array}$$

Now consider the following regular planning problem: given a completely specified initial situation (specified by a formula ψ_{init}), find a plan such that all packages are disarmed. We can formulate this problem as the following formula:

$$\exists \bar{A}, \bar{F}, \bar{I} : \Delta_{act} \wedge \psi_{prec} \wedge \psi_{init} \wedge (\exists t \forall p \neg Armed(p,t)),$$

where with $\bar{A}$, we denote the action predicates $\{Dunk/2, Flush/1\}$, with $\bar{F}$ we denote the fluent predicates $\{Armed/2, Clogged/1\}$ and with $\bar{I}$ we denote the predicates used to describe the initial situation $\{Init\_Clogged/0, Init\_Armed/1\}$. Now imagine that initial situation is not specified, and we want to find a plan that works for all possible initial situations, in other words a *conformant plan*. We can formulate the problem of finding such a plan as follows.

$$\exists \bar{A} \forall \bar{F}, \bar{I} : \Delta_{act} \Rightarrow (\psi_{prec} \wedge \exists t \forall p \neg Armed(p,t)).$$

All this can be formalized in general as follows.

Definition 7.1 (Conformant planning)**.** Let Σ be a vocabulary, consisting of a set of predicates $\overline{A}$, denoting *actions*, $\overline{I}$, denoting *initial fluents*, and $\overline{F}$ denoting *fluents*. Let T_{act} be an FO(ID) theory and T_{init}, T_{prec} and T_{goal} FO theories, all over Σ, such that T_{act} specifies the values of the fluents given an interpretation for the actions and initial fluents, T_{init} is a theory specifying the initial situation, T_{prec} contains preconditions for the actions, and T_{goal} specifies the goal of the planning problem. With ψ_{act} we denote the conjunction of the sentences and possibly definitions in T_{act} and similarly for the other theories. The problem of conformant planning is then to decide the satisfiability of the following formula:

$$\exists \overline{A} \forall \overline{I} \forall \overline{F} : (\psi_{act} \wedge \psi_{init}) \Rightarrow (\psi_{prec} \wedge \psi_{goal}). \tag{6}$$

$$
\exists \bar{A}\bar{R} : \left\{ \begin{array}{rcl}
Clogged^{ct}(0) & \leftarrow & Init\_Clogged^{ct}. \\
Clogged^{ct}(t+1) & \leftarrow & \exists p : Dunk(p,t) \lor (Clogged^{ct}(t) \land \neg Flush(t)). \\
Armed^{ct}(p,0) & \leftarrow & Init\_Armed^{ct}(p). \\
Armed^{ct}(p,t+1) & \leftarrow & Armed(p,t) \land \neg Dunk(p,t). \\
Clogged^{pt}(0) & \leftarrow & Init\_Clogged^{pt}. \\
Clogged^{pt}(t+1) & \leftarrow & \exists p : Dunk(p,t) \lor (Clogged^{pt}(t) \land \neg Flush(t)). \\
Armed^{pt}(p,0) & \leftarrow & Init\_Armed^{pt}(p). \\
Armed^{pt}(p,t+1) & \leftarrow & Armed(p,t) \land \neg Dunk(p,t). \\
Init\_Clogged^{ct} & \leftarrow & \mathbf{f}. \\
Init\_Armed^{ct}(p) & \leftarrow & \mathbf{f}. \\
Init\_Clogged^{pt} & \leftarrow & \neg Init\_Clogged^{cf}. \\
Init\_Clogged^{cf} & \leftarrow & \mathbf{f}. \\
Init\_Armed^{pt}(p) & \leftarrow & \neg Init\_Armed^{cf}(p). \\
Init\_Armed^{cf} & \leftarrow & \mathbf{f}. \\
Clogged^{cf}(t) & \leftarrow & \neg Clogged^{pt}(t). \\
Armed^{cf}(p,t) & \leftarrow & \neg Armed^{pt}(p,t). \\
A^{ct}_{\psi_2} & \leftarrow & \forall pt : Dunk(p,t) \Rightarrow Clogged^{cf}(t) \\
 & & \land \neg(\exists pt : Dunk(p,t) \land Flush(t)) \\
 & & \land \forall p_1 p_2 t : Dunk(p,t) \land Dunk(p_2,t) \Rightarrow p_1 = p_2 \\
 & & \land \forall p t_1 t_2 : Dunk(p,t_1) \land Dunk(p,t_2) \Rightarrow t_1 = t_2 \\
 & & \land \exists t \forall p : Armed^{cf}(p,t).
\end{array} \right\}
$$

$$\land$$

$$A^{ct}_{\psi_2}.$$

Figure 4: The complete approximation of the Clogged Bombs in the Toilet example.

In words, there must be a plan ($\exists \bar{A}$), such that no matter how the nondeterministic aspects turn out ($\forall \bar{I}, \bar{F}$), as long as the specification of the effects of the actions (ψ_{act}) and the (partial) specification of the initial situation (ψ_{init}) are obeyed, the plan will be executable (ψ_{prec}) and achieve the goal (ψ_{goal}).

Formula 6 is now exactly of the form we assumed above, and we can thus use one of our methods to approximate conformant planning problems.

Example 7.1. (continued) Continuing the Clogged Bombs in the Toilet example, by using the ct/pt-approximation for the definition, and unfolding the constraint $(\psi_{prec} \land \exists t \forall p \neg Armed(p,t))^{ct}$, we get the approximating $\exists SO$ formula $\mathcal{APP}^{wf}_{BU,Unf}$ (Definition 5.7), shown in Figure 4, where $\bar{R}$ are the ct- and cf-predicates introduced by the approximation method.

The result of applying our general approximation method to a conformant planning problem, as specified by $T_{act}, T_{prec}, T_{goal}$ and T_{init} as above, is very similar to the approximation of an $\mathcal{AL}$ action theory by a logic program as in the work of Son et al. (2005). However, there are some small differences in the details that make it difficult to formally compare the two. Nevertheless, for all experiments discussed in this section, our method always finds a correct solution (unless it times out), as does the method of Son et al. Moreover, the two approaches also found these solutions in comparable execution times.

In more detail, Table 3 presents the following results. We implemented a conformant planner by iteratively calling the IDP model generator for FO(ID) (Mariën et al., 2006) on our approximation, giving it an increasing number of timesteps until either a plan is

Problem	IDP	Smodels	Cmodels
BT(2,2)	0.438	0.199	**0.145**
BT(4,2)	0.513	0.219	**0.212**
BT(6,2)	1.050	0.587	**0.425**
BT(8,4)	**1.55**	30.9	2.39
BT(10,4)	**2.80**	-	5.80
BTC(2,2)	0.273	**0.136**	0.139
BTC(4,2)	0.844	0.412	**0.389**
BTC(6,2)	1.60	3.88	**1.23**
BTC(8,4)	**43.7**	-	102
Cleaner(2,2)	0.644	**0.226**	0.376
Cleaner(2,5)	1.57	72.5	**1.36**
Cleaner(2,10)	-	-	-
Cleaner(4,2)	1.55	13.8	**1.13**
Cleaner(4,5)	**2460**	-	-
Cleaner(4,10)	-	-	-
Cleaner(6,2)	8.30	-	**6.16**
Cleaner(6,5)	-	-	-
Domino(100)	0.176	0.096	**0.090**
Domino(200)	0.181	**0.114**	0.151
Domino(500)	**0.212**	0.324	0.354
Domino(1000)	**0.236**	0.618	0.660
Domino(2000)	**0.339**	1.22	1.32
Ring(2)	0.655	**0.285**	0.296
Ring(4)	1.56	2.092	**0.937**
Ring(6)	7.35	19.1	**3.542**
Ring(8)	157	-	**19.860**
Ring(10)	1537	-	**232**

Table 3: Comparison IDP vs Cmodels vs Smodels

found or a maximum number of timesteps is reached. We then compared this planner to the CP_{ASP} conformant planner (Son et al., 2005), using the same experimental setup as in Section 6. CP_{ASP} takes an action theory in the action language $\mathcal{AL}$, and encodes an approximation of the transition diagram corresponding to that action theory, by means of an answer set program. Then any answer set solver can be used to find conformant plans. As Son et al., we used as the ASP solver behind CP_{ASP} both CModels (E. Giunchiglia & Maratea, 2011) and SModels (Niemelä, Simons, & Syrjänen, 2000). As Table 3 shows, the combination of our approximation and the IDP system is comparable to, but overall slightly worse, than the combination of CModels and Son et al.'s approximation. When compared to the same approximation given to SModels, our method tends to be a bit better. These results are in line with results from the ASP competition (Denecker et al., 2009) concerning the performance of SModels, CModels and IDP in general, suggesting that, for conformant planning, our approximation and that of Son et al. are of comparable quality.

Another approximative method for solving conformant planning problems can be found in the work of Palacios and Geffner (2009). In their paper, the authors consider conformant planning problems, specified in the language Strips extended with conditional effects and negation. They define a transformation K_0 that transforms such a conformant planning problem into a classical planning problem in a sound but incomplete way. For each fluent literal L in the conformant planning specification, two new literals KL and $K\neg L$ are created, denoting that L is known to be true, resp. known to be not true, and the initial situation, action preconditions and effects are translated into an initial situation, preconditions and effects with reference to these new knowledge literals. It is not hard to verify that our approximation method generalizes this transformation: if we take an $\exists\forall SO$ encoding of a conformant planning problem P, the $\exists SO$ approximation obtained by our method can be interpreted as a classical planning problem in the ct/cf vocabulary. This planning problem will exactly be the planning problem specified by $K_0(P)$ (i.e., the action preconditions and effects correspond), apart from the initial situation. The K_0 transformation does not do propagation on knowledge about the initial situation: given an initial situation I (specified as a set of clauses), then $K_0(I)$ consists of only those literals KL where L is a *unit clause* in I. This means that, e.g., for an initial situation $I = \{P \vee Q, \neg P\}$, $K_0(I)$ will not include the literal KQ, while our method will be able to infer that Q^{ct} holds (which means our approximation method will be more complete than the K_0 transformation).

Being a general method, ours does not only allow for solving conformant planning problems, but also allows for approximating a number of related problems in temporal domains. Consider, for example, the following problem: 'Given that a certain action A happens at timepoint t, will this certainly lead to a property ψ being true'? This can be formalized as the following $\forall SO$ satisfiability problem, to which our method applies again.

$$\forall \bar{A}\bar{I}\bar{F} : ((\Delta_{act} \wedge \psi_{init} \wedge \psi_{prec} \wedge A(t)) \Rightarrow \psi).$$

This formula is true if for *all possible plans* in which $A(t)$ happens, the property ψ holds. A variant on this problem is the so-called *projection problem*: 'Given that we exactly know which actions happened (we can thus assume that the preconditions were satisfied), does the property ψ hold'? In order to formulate this problem as a SO satisfiability problem, we need to express *that these and only these* actions happened. This can be done, for example, by using an inductive definition $\Delta_{\bar{A}}$. The projection problem can then be expressed as the $\forall \bar{A}\bar{I}\bar{F} : ((\Delta_{act} \wedge \psi_{init} \wedge \Delta_{\bar{A}}) \Rightarrow \psi)$ satisfiability problem. Another variant is the following problem: 'If the property ψ_1 holds for a certain plan, does property ψ_2 also hold?', which can be expressed as the $\forall \bar{A}\bar{I}\bar{F}((\Delta_{act} \wedge \psi_{init} \wedge \psi_{prec} \wedge \psi_1) \Rightarrow \psi_2)$ satisfiability problem.

7.2 Querying and Reasoning in Open Databases

Approximate methods similar to ours have been used in the context of databases without complete information, in particular in databases without CWA, such as *open* databases (Baral et al., 1998; Liu & Levesque, 1998) or databases that make forms of *local closed world assumptions* (Denecker et al., 2010; Doherty et al., 2006). In most of these papers the goal is to compute certain or possible answers to queries. Because this task has a high complexity (from CoNP for a locally closed database without integrity constraints to possibly Δ_2^P for databases with first-order constraints - assuming a given finite domain), approximate methods are presented which translate an FO query into an approximate FO or FO(FP)[4] query that can be solved directly against the database tables using standard (polynomial) query methods.

The method presented in this paper can provide a similar functionality. Let DB be a set of ground literals, representing an incomplete database. Let Ψ be a background theory:

4. FO(FP) is the extension of FO with least and greatest fixpoint constructs.

it may contain integrity constraints, view definitions (datalog view programs are a special case of FO(ID) definitions), local closed world statements expressed in FO, etc. For a given FO query $\varphi_Q[\bar{x}]$, the goal is to find all tuples $\bar{d}$ such that $\varphi_Q[\bar{d}]$ holds in all Herbrand models of $DB \cup \Psi$. The problem of deciding whether a *given* tuple $\bar{d}$ is an answer corresponds to the satisfiability problem of the formula

$$\forall \bar{R}(DB \wedge \Psi \Rightarrow \varphi_Q[\bar{d}]), \tag{7}$$

and we can directly use our approximation method on this problem. While this allows us to answer yes/no queries as well as to decide whether a given tuple $\bar{d}$ is a certain answer to a query, our approximation method does not directly provide a method to compute (an approximation of) all such tuples.

However, let us look at the following $\exists SO$ satisfiability problem.

$$\exists \bar{R}' : \mathcal{D}^{DB \wedge \Psi} \cup Approx^{BU}(\varphi_Q[\bar{x}]),$$

It looks very much like our approximation of $\exists\forall(\Delta \wedge \psi_1) \Rightarrow \psi_2$ satisfiability problems (as formulated in Proposition 5.5). We again have the definition $\mathcal{D}^{DB \wedge \Psi}$ approximating the database DB and background knowledge Ψ (note that Ψ possibly contains definitions), and the bottom up evaluation of the query, only now the constraint A_Q^{ct} has been dropped.

The definition $\mathcal{D}^{DB \wedge \Psi} \cup Approx^{BU}(\varphi_Q[\bar{x}])$ consists of rules describing propagations allowed by the database and the theory Ψ, and rules defining the predicate symbol $A_{\varphi_Q}^{ct}$, where A_{φ_Q} is the Tseitin predicate representing the query $\varphi_Q[\bar{x}]$. In the unique Herbrand model of this definition, the interpretation of $A_{\varphi_Q}^{ct}$ contains those tuples for which our propagation can derive that they certainly satisfy the query – a sound approximation of the full set of answers!

In the work of Denecker et al. (2010), a *locally closed database* $LCDB$ is assumed. Such a locally closed database consists of a standard database DB, i.e. a set of atoms, together with a set of *local closed world assumptions* $\mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])$. Each of these LCWA statements expresses that the database's knowledge about P is complete for those tuples $\bar{x}$ that satisfy the formula $\Psi[\bar{x}]$. An atom $P(\bar{d})$ is therefore true if it is in DB and it is false if it is not in DB and there is a $\mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])$ such that $\Psi[\bar{d}]$ holds in the domain of discourse; otherwise it is unknown. The authors then present an approximate reasoning method for query answering in locally closed databases and show how this approximate query answering can be formulated as a fixpoint query. Basically, this boils down to the following. One constructs the following definition

$$\Delta_{\mathcal{LCWA}} = \left\{ \begin{array}{c} \dots \\ P^{ct}(\bar{x}) \leftarrow P(\bar{x}) \\ P^{cf}(\bar{x}) \leftarrow \neg P^{ct}(\bar{x}) \wedge \Psi_P^{ct}[\bar{x}] \\ \dots \end{array} \right\},$$

for every relation P and every local closed world assumption $\mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])$. Although the authors do not phrase it in this form, their method for finding an approximation of the certain answers to a query $\varphi_Q[\bar{x}]$ actually boils down to solving the following satisfiability problem:

$$\exists \bar{R}' : DB \wedge CWA(DB) \wedge \Delta_{\mathcal{LCWA}} \wedge Approx^{BU}(\varphi_Q[\bar{x}]),$$

Here $\bar{R}'$ denotes all predicates and auxiliary predicates occurring in the body of the existential formula. With $CWA(DB)$, we denote the formula expressing *closed world assumption* for the database DB. The presence of this closed world assumption might seem strange at first sight, since the whole idea behind locally closed world databases is to not assume CWA per default. However, in order to correctly apply the local closed world assumptions, we need an exact specification of what is in the database and what is not, and this is precisely what is expressed by $DB \wedge CWA(DB)$. Indeed, given $DB \wedge CWA(DB)$, $\Delta_{\mathcal{LCWA}}$ can be seen as an approximative definition of what is certainly true and false in the context of the locally closed world assumptions. Again the predicate $A^{ct}_{\varphi_Q}$ will contain an approximate answer to the query $\varphi_Q[\bar{x}]$, i.e., a lower bound on all the tuples for which the query $\varphi_Q[\bar{x}]$ is certainly true. Similarly, the predicate $A^{cf}_{\varphi_Q}$ will contain a lower bound on all the tuples for which the query is false.

A limitation of the approach by Denecker et al. is that they extend the above method for only one type of integrity constraints, namely functional dependencies. The way these functional dependencies are handled is by extending $\Delta_{\mathcal{LCWA}}$ with extra propagation rules taking these functional dependencies into account. By contrast, our more general method can be used to easily extend this to *arbitrary* integrity constraints. This works as follows. Let T_{int} be a set of first-order integrity constraints. We can then approximate the problem of finding certain queries by the following satisfiability problem.

$$\exists \bar{R}' : Approx(T_{int}) \cup \{A_{T^{ct}_{int}} \leftarrow \mathbf{t}\} \wedge DB \wedge CWA(DB) \wedge \Delta_{\mathcal{LCWA}} \wedge Approx^{BU}(\varphi_Q[\bar{t}]).$$

Again, the predicate $A^{ct}_{\varphi_Q}$ will contain an approximate answer to the query $\varphi_Q[\bar{x}]$.

Doherty et al. (2006), propose yet another approach to asking queries to an incomplete database. The authors use the term *approximate database* to denote a database, consisting of two layers: an extentional and an intensional layer. Both of these layers have an external representation towards the user, and an internal representation.

The *extentional* database consists of positive and negative literals, and is internally stored as a classical database, using the Feferman transformation. For example, the extentional database (EDB), as entered by a user,

$$Color(Car1, Black), \neg Color(Car1, Red), Color(Car2, Red),$$

is internally stored as

$$Color^{ct}(Car1, Black), Color^{cf}(Car1, Red), Color^{ct}(Car2, Red).$$

The *intentional* database consists of rules to infer additional information from the facts in the EDB. The user can write down rules of the form $(\neg)P_1(\bar{x}_1) \wedge \ldots \wedge (\neg)P_n(\bar{x}_n) \rightarrow (\neg)P(\bar{x}))$, which are then internally stored as $((\neg)P_1(\bar{x}_1))^{ct} \wedge \ldots \wedge ((\neg)P_n(\bar{x}_n))^{ct} \rightarrow ((\neg)P(\bar{x})))^{ct}$. An example of such a IDB rule is the following rule

$$Color(x, y_1) \wedge y_1 \neq y_2 \rightarrow \neg Color(x, y_2),$$

which is internally stored as

$$Color^{ct}(x, y_1) \wedge y_1 \neq y_2 \rightarrow Color^{cf}(x, y_2).$$

To evaluate a query, a naive algorithm based on exhaustively applying all rules on the EDB is used.

The rules in the IDB resemble our INF formulas in the sense that both describe *valid inferences* that can be made based on incomplete information. The internal representation of the IDB is indeed similar to our representation of INF formulas as definitional rules. However, a key difference is that in the approach of Doherty et al., when a user wants to add a property ψ to the database (e.g., 'a car can only have one color'), he has to write down all inferences that are valid according to that property, while in our approach these inference rules are automatically generated from the property itself. Manually writing down all valid inferences sanctioned by a property is not an easy task. For example, take the property "a car has to be inspected if and only if it was suspect and black" from the paper by Doherty et al.. This can be expressed in FO as the formula $\psi = \forall c(Suspect(c) \wedge Color(c, Black) \Leftrightarrow Investigate(c))$. While, in our method, $Approx(\psi)$ constructs an approximation of all valid inferences that can be made from this formula, the user has to write down the following rules in Doherty et al.'s approach:

$$
\begin{gathered}
Suspect(c) \wedge Color(c, Black) \rightarrow Investigate(c) \\
Suspect(c) \wedge \neg Investigate(c) \rightarrow \neg Color(c, Black) \\
\neg Suspect(c) \rightarrow \neg Investigate(c) \\
\ldots
\end{gathered}
$$

Our method therefore generalizes the work of Doherty et al. by deriving these rules automatically from a general first-order theory.

Liu and Levesque (1998) propose another type of reasoning in open databases. They only consider a simple form of first order knowledge bases, called *proper knowledge bases*. An interesting feature of these knowledge bases is that it is easy to obtain a *complete* characterization of what is certainly true, resp. certainly false. In our terminology, that means that one can construct a definition Δ, such that $KB \models P(\bar{d})$ if and only if $\Delta \models P^{ct}(\bar{d})$ and $KB \models \neg P(\bar{d})$ if and only if $\Delta \models P^{cf}$. It holds that every *two valued extension* of the three valued interpretation encoded by Δ is a model of the KB. Then Levesque et al. use an evaluation procedure based on the three-valued Kleene-evaluation to check whether a query holds in the knowledge base. As mentioned earlier, they also define a normal form $\mathcal{NF}$ for queries, for which they prove that the Kleene-evaluation is complete. Our work extends their work, in the sense that we can take a general first order knowledge base and approximately solve queries, as we have shown above. Of course, since in general we can no longer guarantee a complete characterization of what is certainly true/false, we can no longer guarantee completeness, even if the query is in the normal form $\mathcal{NF}$. Another difference between the work of Liu and Levesque and our work here, is that they assume a fixed countable *infinite* domain, while we assume a fixed finite domain. While this is indeed a theoretical difference, in practice it does not make any difference, since their evaluation method only considers a *finite* set of domain elements that can be determined up-front.

8. Conclusions and Future Work

Even if a problem is computationally hard in general, specific instances of it might still be solved efficiently. This is why approximate methods are important: they cannot solve every instance, but the instances they can solve, they solve quickly. In computational logic, hard problems arise quite readily. It is therefore not surprising that the literature contains numerous examples of algorithms that perform approximate reasoning tasks for various logical formalisms in various specific contexts. Since many of these algorithms share common ideas, it is a natural question whether they can be seen as instances of some more general method for a more general language.

This paper presents such a method. We start from the propagation method for FO($\cdot$) developed by Wittocx, Mariën, and Denecker (2008) and its symbolic expression (Wittocx, 2010) and generalize this to a method for approximating the Σ_2^P-complete $\exists\forall SO(ID)$ satisfiability problem by solving an NP problem. Importantly, this is a syntactic method that transforms the $\exists\forall SO(ID)$ formula into an $\exists SO(ID)$ formula. This affords us the freedom to use any off-the-shelf solver for such a language to perform the approximative reasoning. Moreover, it also makes it significantly easier to update the method by adding (or removing) specific propagations.

Since our method is an approximation, it is necessarily incomplete. Nevertheless, our experiments have shown that, in practice, it often does manage to find a solution. An interesting topic for future work is to determine classes of problems, for which our method can be shown to be complete.

In summary, the contributions of this paper are that (1) we have extended the logical representation describing the propagation process to a general method for approximating $SAT(\exists\forall SO)$ problems; (2) we have shown how to approximate inductive definitions, and use this to approximate a class of useful $SAT(\exists\forall SO(ID))$-problems; and (3) we have examined how existing approximation methods fit into our general framework.

Acknowledgments

This work is supported by Research Foundation - Flanders FWO-Vlaanderen, projects G.0489.10 and G.035712, and by Research Fund KULeuven, project GOA/08/008. Hanne Vlaeminck is supported by IWT-Vlaanderen.

Appendix A. An Example of an Approximation

Figure 5 shows the full approximation of ψ_{act} as in Example 1.1.

Appendix B. Proof of Theorem 3.1

Proof. First, remark that Feferman (1984) showed that the four-valued evaluation of a formula φ in an interpretation $\mathcal{I}$ can be simulated by computing the standard two-valued evaluation of φ^{ct} and φ^{cf} in $\mathcal{I}^{tf}$. It is easy to verify that the bottom-up rules in $Approx(\psi)$ inductively encode this evaluation. We split the proof in two parts. First we assume that $\mathcal{I}$ is three-valued. We show that in this case only the bottom-up rules are used, i.e., leaving out the top-down rules does not change the model of the definition. This proves the first part of the theorem, and together with the above remark also proves the second part of the theorem for the case that $\mathcal{I}$ is three-valued. Then, all that is left to prove, is that the second part of the theorem also holds for four-valued $\mathcal{I}$.

$$
\left\{
\begin{array}{lcl lcl}
A^{ct}_{\psi_{act}} & \leftarrow & A^{ct}_0 \wedge A^{ct}_8. & A^{cf}_8 & \leftarrow & A^{cf}_9. \\
A^{cf}_{\psi_{act}} & \leftarrow & A^{cf}_0. & A^{cf}_8 & \leftarrow & A^{cf}_{11}. \\
A^{cf}_{\psi_{act}} & \leftarrow & A^{cf}_8. & A^{ct}_8 & \leftarrow & (A^{ct}_9 \wedge A^{ct}_{11}). \\
A^{ct}_0 & \leftarrow & A^{ct}_{\psi_{act}}. & A^{ct}_9 & \leftarrow & A^{ct}_8. \\
A^{ct}_8 & \leftarrow & A^{ct}_{\psi_{act}}. & A^{cf}_9 & \leftarrow & (A^{cf}_8 \wedge A^{ct}_{11}). \\
A^{cf}_0 & \leftarrow & A^{cf}_{\psi_{act}} \wedge A^{ct}_8. & A^{ct}_{11} & \leftarrow & A^{ct}_8. \\
A^{cf}_8 & \leftarrow & A^{cf}_{\psi_{act}} \wedge A^{ct}_0. & A^{cf}_{11} & \leftarrow & (A^{cf}_8 \wedge A^{ct}_9). \\
 & & & A^{ct}_9 & \leftarrow & Clean^{cf}(0). \\
A^{ct}_0 & \leftarrow & (\forall t : A^{ct}_1(t)). & A^{ct}_9 & \leftarrow & InitiallyClean^{ct}. \\
A^{cf}_0 & \leftarrow & (\exists t : A^{cf}_1(t)). & A^{cf}_9 & \leftarrow & (Clean^{ct}(0) \wedge InitiallyClean^{cf}). \\
A^{ct}_1(t) & \leftarrow & A^{ct}_0. & Clean^{ct}(0) & \leftarrow & A^{cf}_9. \\
A^{cf}_1(t) & \leftarrow & (A^{cf}_0 \wedge (\forall t_1 : (t_1 = t \vee A^{ct}_1(t_1)))). & Clean^{cf}(0) & \leftarrow & (A^{ct}_9 \wedge InitiallyClean^{cf}). \\
A^{cf}_1(t) & \leftarrow & A^{cf}_2(t). & InitiallyClean^{cf} & \leftarrow & A^{cf}_9. \\
A^{cf}_1(t) & \leftarrow & A^{cf}_5(t). & InitiallyClean^{ct} & \leftarrow & (A^{ct}_9 \wedge Clean^{ct}(0). \\
A^{ct}_1(t) & \leftarrow & (A^{ct}_2(t) \wedge A^{ct}_5(t)). & A^{ct}_{11} & \leftarrow & A^{ct}_{12}. \\
A^{ct}_2(t) & \leftarrow & A^{ct}_1(t). & A^{ct}_{11} & \leftarrow & Clean^{ct}(0). \\
A^{cf}_2(t) & \leftarrow & (A^{cf}_1(t) \wedge A^{ct}_5(t)). & A^{cf}_{11} & \leftarrow & (A^{cf}_{12} \wedge Clean^{cf}(0)). \\
A^{ct}_5(t) & \leftarrow & A^{ct}_1(t). & A^{cf}_{12} & \leftarrow & A^{cf}_{11}. \\
A^{cf}_5(t) & \leftarrow & (A^{cf}_1(t) \wedge A^{ct}_2(t)). & A^{ct}_{12} & \leftarrow & (A^{ct}_{11} \wedge Clean^{cf}(0)). \\
A^{ct}_2(t) & \leftarrow & Clean^{cf}((t+1)). & Clean^{cf}(0) & \leftarrow & A^{cf}_{11}. \\
A^{ct}_2(t) & \leftarrow & A^{ct}_4(t). & Clean^{ct}(0) & \leftarrow & (A^{ct}_{11} \wedge A^{cf}_{12}). \\
A^{cf}_2(t) & \leftarrow & (Clean^{ct}(t+1)) \wedge A^{cf}_4(t)). & A^{ct}_{12} & \leftarrow & InitiallyClean^{cf}. \\
Clean^{ct}(t+1) & \leftarrow & A^{cf}_2(t). & A^{cf}_{12} & \leftarrow & InitiallyClean^{ct}. \\
Clean^{cf}(t+1) & \leftarrow & (A^{ct}_2(t) \wedge A^{cf}_4(t)). & InitiallyClean^{cf} & \leftarrow & A^{ct}_{12}. \\
A^{cf}_4(t) & \leftarrow & A^{cf}_2(t). & InitiallyClean^{ct} & \leftarrow & A^{cf}_{12}. \\
A^{ct}_4(t) & \leftarrow & (A^{ct}_2(t) \wedge Clean^{ct}(t+1)). & & & \\
A^{ct}_4(t) & \leftarrow & Clean^{ct}(t). & & & \\
A^{ct}_4(t) & \leftarrow & Wipe(t). & & & \\
A^{cf}_4(t) & \leftarrow & (Clean^{cf}(t) \wedge \neg Wipe(t)). & & & \\
Clean^{cf}(t) & \leftarrow & A^{cf}_4(t). & & & \\
Clean^{ct}(t) & \leftarrow & (A^{ct}_4(t) \wedge \neg Wipe(t)). & & & \\
A^{ct}_5(t) & \leftarrow & A^{ct}_6(t). & & & \\
A^{ct}_5(t) & \leftarrow & Clean^{ct}(t+1). & & & \\
A^{cf}_5(t) & \leftarrow & (A^{cf}_6(t) \wedge Clean^{cf}(t+1)). & & & \\
A^{cf}_6(t) & \leftarrow & A^{cf}_5(t). & & & \\
A^{ct}_6(t) & \leftarrow & (A^{ct}_5(t) \wedge Clean^{cf}(t+1)). & & & \\
Clean^{cf}((t+1)) & \leftarrow & A^{cf}_5(t). & & & \\
Clean^{ct}((t+1)) & \leftarrow & (A^{ct}_5(t) \wedge A^{cf}_6(t)). & & & \\
A^{cf}_6(t) & \leftarrow & Clean^{ct}(t). & & & \\
A^{cf}_6(t) & \leftarrow & Wipe(t). & & & \\
A^{ct}_6(t) & \leftarrow & (Clean^{cf}(t) \wedge \neg Wipe(t)). & & & \\
Clean^{cf}(t) & \leftarrow & A^{ct}_6(t). & & & \\
Clean^{ct}(t) & \leftarrow & (A^{cf}_6(t) \wedge \neg Wipe(t)). & & &
\end{array}
\right\}
$$

Figure 5: $Approx_{\{Wipe\}}(\psi_{act})$, with ψ_{act} taken from Example 1.1.

So let us assume that $\mathcal{I}$ is three-valued. We prove that $Mod(Approx^{BU}(\psi) \cup \Delta_{\mathcal{I}}) = Mod(Approx(\psi) \cup \Delta_{\mathcal{I}})$ by contradiction. Assume there is a predicate A^{ct}_{φ} (the proof goes analogously for A^{cf}_{φ}) such that $Mod(Approx(\psi) \cup \Delta_{\mathcal{I}}) \models A^{ct}_{\varphi}$ but $Mod(Approx^{BU}(\psi) \cup \Delta_{\mathcal{I}}) \not\models A^{ct}_{\varphi}$. In the preliminaries we recalled that the model of a positive inductive definition is the least-fixpoint of the immediate consequence operator T^{O}_{Δ}. The model of such a definition is thus the limit of a sequence of applications of the immediate consequence operator. One can prove (see, e.g., Denecker & Ternovska, 2004) that we do not have to apply the immediate consequence operator of the *complete* definition in every step. I.e., applying the immediate consequence operator of a subset of the definition, until there no longer exists such a immediate consequence operator that will give something new, gives the same model. Suppose we take such a sequence where we first apply all the bottom-up rules, and only after no bottom-up rules are applicable we try to apply the top-down rules. Suppose A^{ct}_{φ} is the first atom we infer with the top down rules in this sequence. Obviously A^{ct}_{φ} cannot be the top-level atom A^{ct}_{ψ}, since there are no top-down rules for this. We can now do a case study on the type of (sub)formula φ occurs in, e.g., assume that φ is a subformula of the formula ψ where $\psi = \varphi \vee \varphi'$. From the fact that A^{ct}_{φ} is true, it follows that the body of the top down rule $A^{ct}_{\varphi} \leftarrow A^{ct}_{\psi} \wedge A^{cf}_{\varphi'}$ has to be true, and thus that A^{ct}_{ψ} and $A^{cf}_{\varphi'}$ are true. Since A^{ct}_{φ} was the first atom to be inferred by a top-down rule, we have that since A^{ct}_{ψ} is true, also $A^{ct}_{\varphi} \vee A^{ct}_{\varphi'}$ must be true. Now since A^{ct}_{φ} only became true in the last step of the sequence, we have that $A^{ct}_{\varphi'}$ must have been true already. This means that after applying the bottom-up rules $A^{ct}_{\varphi'}$ and $A^{cf}_{\varphi'}$ are both true, which is a contradiction with the fact that $\mathcal{I}$ was three-valued and that the bottom-up rules encode the four-valued evaluation. The proof is analogous for the other types of subformulas.

Now in the case that $\mathcal{I}$ is four-valued, and not three-valued it is no longer the case that only the bottom-up rules contribute to the model (i.e., $Mod(Approx^{BU}(\psi) \cup \Delta_{\mathcal{I}}) \neq Mod(Approx(\psi) \cup \Delta_{\mathcal{I}})$). To see this, consider the following formula $P \wedge Q$, and take for $\mathcal{I}$ the four-valued interpretation such that $P = \mathbf{i}$ and $Q = \mathbf{t}$. Then one can verify that the bottom-up rules in $Approx(\psi) \cup \Delta_{\mathcal{I}}$ will infer that both $A^{ct}_{P \wedge Q}$ and $A^{cf}_{P \wedge Q}$ are true. However, now the top-down rules can also infer that Q^{cf} has to be true. What happens is that once an inconsistency is inferred for a certain subformula, this propagates back down the parse-tree. However, similar to above, we can again do a case study on the structure of ψ to prove that (for the 'top' formula ψ) $Mod(Approx^{BU}(\psi) \cup \Delta_{\mathcal{I}}) \models A^{ct}_{\psi} \wedge A^{cf}_{\psi}$ if and only if $Mod(Approx(\psi) \cup \Delta_{\mathcal{I}}) \models A^{ct}_{\psi} \wedge A^{cf}_{\psi}$. Now since since $Approx^{BU}(\psi)$ is clearly a direct encoding of the four-valued evaluation, this concludes the proof. □

Appendix C. Proof of Proposition 4.3

Proof. Take a witness I for the satisfiability of $\mathcal{APP}^{\Rightarrow}(F)$. First let us remark that $Open(Approx_{\sigma}(\psi_1 \Rightarrow \psi_2) \cup \{A^{ct}_{\psi_1} \leftarrow \mathbf{t}\}) = \sigma$. From the fact that I is a witness of the satisfiability of $\mathcal{APP}^{\Rightarrow}(F)$ we know that the model M of this definition extending I contains A^{ct}_{ψ} and by construction of $Approx_{\sigma}(\psi_1 \Rightarrow \psi_2)$ it must also contain either $A^{cf}_{\psi_1}$ or $A^{ct}_{\psi_2}$.

Assume first that $A^{cf}_{\psi_1}$ is true in M. Then application of Theorem 3.2 (where we take $\Phi = \{\psi_1\}$ and $\varphi' = \psi_1$) gives: if M extends I and $M \models \psi_1$, then $M \not\models \psi_1$, so the assumption that $M \models \psi_1$ results in a contradiction and hence $M \not\models \psi_1$, in which case $\psi_1 \Rightarrow \psi_2$ holds for every M extending I, and thus is I a witness for the satisfiability of F.

Next, assume that $A^{ct}_{\psi_2}$ is true in M. Again applying Theorem 3.2 (where this time $\Phi = \{\psi_1\}$ and $\varphi' = \psi_2$) gives: if M extends I and $M \models \psi_1$ then $M \models \psi_2$, hence also in this case does $\psi_1 \Rightarrow \psi_2$ hold for every M extending I, and again this means that I is a witness of the satisfiability of F.

$\square$

Appendix D. Proof of Theorem 5.2

A key ingredient in the proof of Theorem 5.2 is the following property of $App^{ct/pt}(\Delta)$. Its immediate consequence operator $T^{O^{ct/pt}}_{App^{ct/pt}(\Delta)}$ on two-valued interpretations simulates the immediate consequence operator $\mathcal{T}^{(O_1,O_2)}_{\Delta}$ on four-valued interpretations of the original definition. This is made more precise in the following lemma.

Definition D.1. For a pair of Σ-interpretations (I, J), we use $t(I, J)$ to denote the $\Sigma^{ct/pt}$-interpretation $(I, J)^{ct/pt}$.

Lemma D.1. *For each* (O_1, O_2) *and* (I, J)*,*

$$\mathcal{T}^{(O_1,O_2)}_{\Delta}(I, J) = t^{-1}(T^{t(O_1,O_2)}_{App^{ct/pt}(\Delta)}(t(I, J))).$$

Proof. Let (I', J') be $\mathcal{T}^{(O_1,O_2)}_{\Delta}(I, J)$ and let $F = T^{t(O_1,O_2)}_{App^{ct/pt}(\Delta)}(t(I, J))$. We first show that $F|_{\Sigma^{ct}} = I'$. Since $F|_{\Sigma^{ct}}$ depends only on the rules of $App^{ct/pt}(\Delta)$ with a predicate from Σ^{ct}, we can discard all rules with a head from Σ^{pt}. As a result, we are left with a single copy of Δ in which positive occurrences of atoms have been replaced by their $\cdot^{ct}$ variant and negative ones by their $\cdot^{pt}$ variant. This implies that the evaluation of the bodies of these remaining rules according to $t(O_1 \cup I, O_2 \cup J)$ will be identical to the evaluation of the bodies of the original rules by $(O_1 \cup I, O_2 \cup J)$ in the construction of I', thus proving the equality. The proof of the remaining equality $F|_{\Sigma^{pt}} = J'$ is analoguous. $\square$

Proof of Theorem 5.2. First, recall that, given some partial knowledge (O_1, O_2), the three-valued well-founded models of Δ, resp. $App^{ct/pt}(\Delta)$ are the least fixpoints of operators $\mathcal{ST}^{(O_1,O_2)}_{\Delta}$ resp. $\mathcal{ST}^{t(O_1,O_2)}_{App^{ct/pt}(\Delta)}$ (note that since $t(O_1, O_2)$ is two-valued, we abuse notation here and in the rest of the proof and denote the two-valued pair $(t(O_1, O_2), t(O_1, O_2))$ by $t(O_1, O_2)$).

Now, the latter operator is rather peculiar, in the sense that it is actually juggling four different interpretations of the original alphabet Σ. In more detail, each element in its domain looks like this:

$$(I, J) = \begin{pmatrix} I_{ct} & & J_{ct} \\ \cup & , & \cup \\ I_{pt} & & J_{pt} \end{pmatrix}.$$

where I_{ct} and J_{ct} interpret the alphabet Σ^{ct}, and I_{pt} and J_{pt} interpret Σ^{pt}. If we now apply the operator $\mathcal{ST}^{t(O_1,O_2)}_{App^{ct/pt}(\Delta)}$, we obtain a new such pair:

$$(I', J') = \begin{pmatrix} I'_{ct} & & J'_{ct} \\ \cup & , & \cup \\ I'_{pt} & & J'_{pt} \end{pmatrix}.$$

From the general definition of the $\mathcal{ST}^{\mathcal{O}}_{\Delta}$ construction, it is obvious that $I'_{ct} \cup I'_{pt}$ depends only on $J_{ct} \cup J_{pt}$. However, in this particular case, the operator exhibits even more structure. The operator $ST^{(O_1,O_2)}_{App^{ct/pt}(\Delta)}(J)$ uses its argument J as a *fixed* interpretation for the negative occurrences, which remains constant throughout the least fixpoint computation over the positive occurrences. Now, $App^{ct/pt}(\Delta)$ contains two copies of Δ which interact only through negative occurrences (that is, all occurrences of a $\cdot^{pt}$ predicate in the body of a rule with a $\cdot^{ct}$ predicate in its head are always negative ones, and vice versa). This means that as long as we keep the interpretation of the negative occurrence fixed to a constant value J, these two copies of Δ do not interact at all. Consequently, to construct I'_{ct}, we can discard all rules with a $\cdot^{pt}$ predicate in its head. This means we are left with rules whose head is a $\cdot^{ct}$ predicate and whose body contains only positive occurrences of $\cdot^{ct}$ predicates and negative occurrences of $\cdot^{pt}$ predicates. Therefore, I'_{ct} depends only on J_{pt}. Moreover, the value of I'_{ct} will be such that if we map its symbols back to the original alphabet Σ (let $orig(\sigma^{ct}) = orig(\sigma^{pt}) = \sigma$ for all $\sigma \in \Sigma$), then $orig(I'_{ct}) = ST^{(O_1,O_2)}_{\Delta}(orig(J_{pt}))$. Similarly, we also obtain that:

$$\begin{aligned}
orig(I'_{pt}) &= ST^{(O_1,O_2)}_{\Delta}(orig(J_{ct})),\\
orig(J'_{ct}) &= ST^{(O_1,O_2)}_{\Delta}(orig(I_{pt})),\\
orig(J'_{pt}) &= ST^{(O_1,O_2)}_{\Delta}(orig(I_{ct})).
\end{aligned}$$

In other words,

$$\begin{aligned}
(orig(I'_{ct}), orig(J'_{pt})) &= \mathcal{ST}^{(O_1,O_2)}_{\Delta}(orig(I_{ct}), orig(J_{pt})),\\
(orig(I'_{pt}), orig(J'_{ct})) &= \mathcal{ST}^{(O_1,O_2)}_{\Delta}(orig(I_{pt}), orig(J_{ct})).
\end{aligned}$$

Now, if we consider the construction of the well-founded model of $App^{ct/pt}(\Delta)$, we have a sequence of this form:

$$\begin{pmatrix} I^0_{ct} & & J^0_{ct} \\ \cup & , & \cup \\ I^0_{pt} & & J^0_{pt} \end{pmatrix} \mapsto \begin{pmatrix} I^1_{ct} & & J^1_{ct} \\ \cup & , & \cup \\ I^1_{pt} & & J^1_{pt} \end{pmatrix} \mapsto \begin{pmatrix} I^2_{ct} & & J^2_{ct} \\ \cup & , & \cup \\ I^2_{pt} & & J^2_{pt} \end{pmatrix} \mapsto \cdots \mapsto \begin{pmatrix} I^\infty_{ct} & & J^\infty_{ct} \\ \cup & , & \cup \\ I^\infty_{pt} & & J^\infty_{pt} \end{pmatrix}.$$

Let $(I^i, J^i)_{i \geq 0}$ be the well-founded model construction for the original definition Δ, i.e., for all predicates $P/n \in Def(\Delta)$ and domain tuples $\bar{d} \in D^n$ we have that $(P(\bar{d}))^{I^0} = \mathbf{f}$ and $(P(\bar{d}))^{J^0} = \mathbf{t}$, and $(I^{n+1}, J^{n+1}) = \mathcal{ST}^{(O_1,O_2)}_{\Delta}(I^n, J^n)$ for $n \geq 0$. It is easy to see that we have that $t^{-1}(I^0_{ct} \cup J^0_{pt}) = (I_0, J_0)$. This provides a base case, while the above equation provides the inductive step to prove that, for each i, $(I^i, J^i) = t^{-1}(I^i_{ct} \cup J^i_{pt})$. In other words, the well-founded model construction for the original definition Δ is tracked by these elements of the well-founded model construction for $App^{ct/pt}(\Delta)$:

$$\begin{pmatrix} I^0_{ct} & & \cdot \\ \cup & , & \cup \\ \cdot & & J^0_{pt} \end{pmatrix} \mapsto \begin{pmatrix} I^1_{ct} & & \cdot \\ \cup & , & \cup \\ \cdot & & J^1_{pt} \end{pmatrix} \mapsto \begin{pmatrix} I^2_{ct} & & \cdot \\ \cup & , & \cup \\ \cdot & & J^2_{pt} \end{pmatrix} \mapsto \cdots \mapsto \begin{pmatrix} I^\infty_{ct} & & \cdot \\ \cup & , & \cup \\ \cdot & & J^\infty_{pt} \end{pmatrix}.$$

What about the other diagonal? There we have that $t^{-1}(J^{ct}_0 \cup I^{ct}_0) = (\top, \bot)$, which is the most precise element $\top_p$ in the lattice of pairs of interpretations. Therefore, we find that the other diagonal actually tracks the construction of the *greatest* fixpoint of $\mathcal{ST}_\Delta^{(O_1,O_2)}$. Combining these two results, we see that if (L_1, L_2) and (G_1, G_2) are these least and greatest fixpoint, respectively, the well-founded model of $App^{ct/pt}(\Delta)$ looks like this:

$$\begin{pmatrix} L_1 & & G_1 \\ \cup & , & \cup \\ G_2 & & L_2 \end{pmatrix}.$$

Note that a unique four-valued stable fixpoint is both the least and greatest stable fixpoint and hence also the well-founded fixpoint. This immediately concludes the proof.

□

Appendix E. Proof of Proposition 5.3

Lemma E.1. *Given a Σ-definition Δ, and a FO-formula ψ. Let σ be a subset of Σ' (a renamed copy of Σ). Consider the definition*

$$\mathcal{D} = App^{ct/pt}(\Delta) \cup \{P^{ct} \leftarrow P'^{ct}\}_{P\in\sigma} \cup \{O^{pt} \leftarrow O'^{pt}\}_{O\in\sigma\cap Open(\Delta')}.$$

Assume that we have a three-valued interpretation $\mathcal{I}$ that approximates all models of $\Delta \wedge \psi$. Then it holds that $WFM_{\mathcal{I}^{ct/pt}}(\mathcal{D})$ also approximates $\Delta \wedge \psi$, i.e.,

- *if $P^{ct}(\bar{d}) \in WFM_{\mathcal{I}^{ct/pt}}(\mathcal{D})$, then $\models (\Delta \wedge \psi) \Rightarrow P(\bar{d})$,*
- *if $P^{pt}(\bar{d}) \notin WFM_{\mathcal{I}^{ct/pt}}(\mathcal{D})$, then $\models (\Delta \wedge \psi) \Rightarrow \neg P(\bar{d})$.*

Proof. We prove this through induction over a well-founded model construction alternative to the one we described in this paper, which can be found in the work by Denecker and Vennekens (2007). Assume we have an induction sequence $(\mathcal{I}_i)_{0\leq i\leq n}$, of four-valued $Def(\mathcal{D})$-interpretations such that $\mathcal{I}_0$ is the interpretation in which everything is completely unknown, and $\mathcal{I}_n = WFM_{\mathcal{I}^{ct/pt}}(\mathcal{D})$. We prove for every i that $\mathcal{I}_i$ is a sound approximation of $\Delta \wedge \psi$. This is trivially the case for $n = 0$. So now assume that $\mathcal{I}_k$ is a sound approximation of $\Delta \wedge \psi$. We have to prove that this is also the case for $\mathcal{I}_{k+1}$. We need to prove two things for this: first, an atom $P^{ct}(\bar{d})$ cannot be true in $\mathcal{I}_{k+1}$ if $P(\bar{d})$ is not true in all models of $\Delta \wedge \psi$, and second, that an atom $P^{pt}(\bar{d})$ cannot be false in $\mathcal{I}_{k+1}$ if it is not false in all models of $\Delta \wedge \psi$. We prove both cases by contradiction.

We start with the first case. So assume that in the k-th well-founded induction step, for a certain predicate P and domain tuple $\bar{D}$, $P^{ct}(\bar{d})$ is incorrectly deduced, i.e., there exists a model $M \models \Delta \wedge \psi$, s.t. $M \not\models P(\bar{d})$. Since $P^{ct}(\bar{d})$ was inferred in the k-th step, this means that the body of the rule defining $P^{ct}(\bar{d})$, that is, $(\varphi_P)^{ct}[\bar{d}]$ was already made true in a previous step. The induction hypothesis then tells us that in every model M of $\Delta \wedge \psi$ it holds that $M \models \varphi_P[\bar{d}]$, but then the semantics of inductive definitions says that also $M \models P$, which is a contradiction with our assumption.

Next, we consider the second case. This time, assume that in the k-th well-founded induction step, $P^{pt}(\bar{d})$ is inferred to be false, while there exists a model M of $\Delta \wedge \psi$, s.t. $M \models \Delta \wedge \psi \wedge P(\bar{d})$. Now, using this alternative version of the well-founded semantics, there are two ways for a domain atom to become false. Indeed, a domain atom can become false because the body of the defining rule was already false, or because it is part of an *unfounded set*.

If $P^{pt}(\bar{d})$ was made false because the body of the defining rule $(\varphi_P)^{pt}$ was already false, an argument completely analogous to the one above - using the induction hypothesis - again gives a contradiction with the assumption. So now, all that is left to prove, is that $P^{pt}(\bar{d})$ cannot be incorrectly made false through the application of an unfounded set rule. If $P^{pt}(\bar{d})$ is made false through the application of the unfounded set rule, this means that there is a set US of atoms, that are unknown in $\mathcal{I}_k$, such that when made false in the bodies of the rules defining these atoms, the Kleene evaluation of these bodies returns false. It is possible to verify that we can always find an US such that it only contains $\cdot^{pt}$-atoms.

Now let us take such a model M of $\Delta \wedge \psi$. Such an M is obviously also a model of Δ. Consider the corresponding set US', consisting of domain atoms $P(\bar{d})$, such that $P^{pt}(\bar{d}) \in US$. For every atom $Q^{pt}[\bar{d}]$ in the body $(\varphi_P)^{pt}$ not in the set US, the induction hypothesis actually tells us that $(Q[\bar{d}])^M \leq_t (Q^{pt}[\bar{d}])^{\mathcal{I}_k}$. Similarly, for every atom Q^{ct} in the body of $(\varphi_P)^{pt}$, it says that $(Q^{ct}[\bar{d}])^{\mathcal{I}_k} \leq_t (Q[\bar{d}])^M$. Now, since Q^{pt} atoms occur only positively and and Q^{ct} only negatively in $(\varphi_P)^{pt}$, it follows that M interprets literals that are not in US' in a more 'false' way than $\mathcal{I}_k$. Thus, US' is also an unfounded set (indeed, turning all the atoms in US' to false will make all the bodies of the defining rules false), and thus $M \models \neg P(\bar{d})$, which is again a contradiction with the assumption, and which concludes the proof of this lemma.

□

Proof of Proposition 5.3. The proof of this proposition is now an easy proof over induction on the construction of the well-founded model of $\mathcal{D}^{\Delta \wedge \psi}$, using the lemma above, and the soundness of $Approx(\psi)$. □

References

Baral, C. (2003). *Knowledge Representation, Reasoning, and Declarative Problem Solving.* Cambridge University Press, New York, NY, USA.

Baral, C., Gelfond, M., & Kosheleva, O. (1998). Expanding queries to incomplete databases by interpolating general logic programs. *J. Log. Program.*, *35*(3), 195–230.

Baral, C., Kreinovich, V., & Trejo, R. (2000). Computational complexity of planning and approximate planning in the presence of incompleteness. *Artif. Intell.*, *122*(1-2), 241–267.

Belnap, N. D. (1977). A useful four-valued logic. In Dunn, J. M., & Epstein, G. (Eds.), *Modern Uses of Multiple-Valued Logic*, pp. 8–37. Reidel, Dordrecht. Invited papers from the Fifth International Symposium on Multiple-Valued Logic, held at Indiana University, Bloomington, Indiana, May 13-16, 1975.

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

Denecker, M., Cortés Calabuig, Á., Bruynooghe, M., & Arieli, O. (2010). Towards a logical reconstruction of a theory for locally closed databases. *ACM Transactions on Database Systems*, *35*(3), 22:1–22:60.

Denecker, M., & Ternovska, E. (2004). A logic of non-monotone inductive definitions and its modularity properties. In Lifschitz, V., & Niemelä, I. (Eds.), *LPNMR*, Vol. 2923 of *LNCS*, pp. 47–60. Springer.

Denecker, M., & Ternovska, E. (2007). Inductive situation calculus. *Artificial Intelligence*, *171*(5-6), 332–360.

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

Denecker, M., & Vennekens, J. (2007). Well-founded semantics and the algebraic theory of non-monotone inductive definitions. In Baral, C., Brewka, G., & Schlipf, J. S. (Eds.), *LPNMR*, Vol. 4483 of *LNCS*, pp. 84–96. Springer.

Denecker, M., Vennekens, J., Bond, S., Gebser, M., & Truszczyński, M. (2009). The second Answer Set Programming competition. In Erdem, E., Lin, F., & Schaub, T. (Eds.), *LPNMR*, Vol. 5753 of *LNCS*, pp. 637–654. Springer.

Doherty, P., Magnusson, M., & Szalas, A. (2006). Approximate databases: a support tool for approximate reasoning. *Journal of Applied Non-Classical Logics*, *16*(1-2), 87–118.

E. Giunchiglia, Y. L., & Maratea, M. (2011). Cmodels homepage. `http://www.cs.utexas.edu/users/tag/cmodels.html`.

Fagin, R. (1974). Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of Computation*, *7*, 43–74.

Feferman, S. (1984). Toward useful type-free theories. *Journal of Symbolic Logic*, *49*(1), 75–111.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In Kowalski, R. A., & Bowen, K. A. (Eds.), *ICLP/SLP*, pp. 1070–1080. MIT Press.

Immerman, N. (1998). *Descriptive Complexity*. Springer Verlag.

Kleene, S. C. (1952). *Introduction to Metamathematics*. Van Nostrand.

Liu, Y., & Levesque, H. J. (1998). A completeness result for reasoning with incomplete first-order knowledge bases. In *KR*, pp. 14–23.

Mariën, M., Wittocx, J., & Denecker, M. (2006). The IDP framework for declarative problem solving. In *Search and Logic: Answer Set Programming and SAT*, pp. 19–34.

McDermott, D. (1987). A critique of pure reason. *Computational Intelligence*, *3*, 151–160.

Mitchell, D. G., & Ternovska, E. (2005). A framework for representing and solving NP search problems. In Veloso, M. M., & Kambhampati, S. (Eds.), *AAAI*, pp. 430–435. AAAI Press / The MIT Press.

Niemelä, I., Simons, P., & Syrjänen, T. (2000). Smodels: A system for answer set programming. In *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning*, Breckenridge, Colorado, USA. CoRR, cs.AI/0003033.

Palacios, H., & Geffner, H. (2009). Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research (JAIR)*, *35*, 623–675.

Son, T. C., Tu, P. H., Gelfond, M., & Morales, A. R. (2005). An approximation of action theories of and its application to conformant planning. In Baral, C., Greco, G., Leone, N., & Terracina, G. (Eds.), *LPNMR*, Vol. 3662 of *LNCS*, pp. 172–184. Springer.

Tamaki, H., & Sato, T. (1984). Unfold/fold transformations of logic programs. In *ICLP*, pp. 127–138.

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

Turner, H. (2002). Polynomial-length planning spans the polynomial hierarchy. In *JELIA*, pp. 111–124.

van Fraassen, B. (1966). Singular terms, truth-value gaps and free logic. *Journal of Philosophy*, *63*(17), 481–495.

Van Gelder, A. (1993). The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, *47*(1), 185–221.

Vlaeminck, H., Wittocx, J., Vennekens, J., Denecker, M., & Bruynooghe, M. (2010). An approximate method for solving ∃∀SO problems. In Fisher, M., van der Hoek, W., Konev, B., & Lisitsa, A. (Eds.), *JELIA*, Lecture Notes in Computer Science, pp. 326–338. Springer.

Wittocx, J. (2010). *Finite Domain and Symbolic Inference Methods for Extensions of First-Order Logic*. Ph.D. thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium.

Wittocx, J., Denecker, M., & Bruynooghe, M. (2010). Constraint propagation for extended first-order logic. *CoRR*, *abs/1008.2121*.

Wittocx, J., Mariën, M., & Denecker, M. (2008). Approximate reasoning in first-order logic theories. In Brewka, G., & Lang, J. (Eds.), *KR*, pp. 103–112. AAAI Press.