Ultimate approximation and its application in nonmonotonic knowledge representation systems ⋆

Marc Denecker [a]

[a] *Department of Computer Science, K.U.Leuven*
Celestijnenlaan 200A, B-3001 Heverlee
Département d'Informatique, U.L.Bruxelles
Bld du Triomphe CP 212, B-1050 Brussels
Belgium

Victor W. Marek [b]

[b] *Department of Computer Science*
University of Kentucky
Lexington, KY 40506-0046
USA

Mirosław Truszczyński [c]

[c] *Department of Computer Science*
University of Kentucky
Lexington, KY 40506-0046
USA

Abstract

In this paper we study fixpoints of operators on lattices and bilattices in a systematic and principled way. The key concept is that of an *approximating* operator, a monotone operator on the *product bilattice*, which gives approximate information on the original operator in an intuitive and well-defined way. With any given approximating operator our theory associates several different types of fixpoints, including the Kripke-Kleene fixpoint, stable fixpoints and the well-founded fixpoint, and relates them to fixpoints of operators being approximated. Compared to our earlier work on approximation theory, the contribution of this paper is that we provide an alternative, more intuitive and better motivated construction of the well-founded and stable fixpoints. In addition, we study the space of approximating operators by means of a *precision* ordering and show that each lattice operator O has a unique most precise — we call it *ultimate* — approximation. We demonstrate that fixpoints of this ultimate approximation provide useful insights into fixpoints of the operator O. We then discuss applications of these results in logic programming.

Preprint submitted to Elsevier Science *29 March 2004*

Key words: Knowledge Representation, Bilattices, Approximation

1 Introduction

This paper presents a fixpoint theory of lattice operators. It has its origin and applications in knowledge representation and declarative programming, and is particularly useful in the study of semantics of nonmonotonic logics. It casts major nonmonotonic modes of reasoning in an abstract algebraic framework, reveals constructive principles behind them, and provides a clear understanding of how nonmonotonic logics are related to each other. Our theory can also be viewed as an extension of Tarski's least-fixpoint theory of *monotone* lattice operators [Tar55] to the case of arbitrary ones, computing fixpoints by *iterated induction* [Acz77], rather than monotone induction.

It is well known that semantic objects such as interpretations and possible-world structures form complete lattices. A theory in a nonmonotonic logic determines a *characteristic* operator on the complete lattice of semantic objects appropriate for the logic. That operator formalizes a view of the theory as a device for *revising* interpretations. A semantics of the theory is defined by the set of fixpoints of its characteristic operator, or by some subset of these fixpoints. Arguably, the most representative example of a characteristic operator is the operator T_P associated with a normal logic program P and defined on the lattice of interpretations [vEK76]. We study it in Section 6, where we show how techniques and results of our paper apply to logic programming. Other noteworthy examples of characteristic operators include the operators D_T [DMT99] associated with a modal theory T [DMT99] (implicitly defined in [Moo84]), and E_Δ, associated with a default theory Δ [DMT99], both defined on the lattice of possible-world structures.

Consequently, an abstract framework of lattices and operators on lattices has emerged as an effective tool in investigations of nonmonotonic reasoning. This algebraic approach can be traced back to studies of semantics of logic programs [vEK76,AvE82,Fit85,Prz90,Van93] and of applications of lattices and bilattices in knowledge representation [Gin88]. Fitting [Fit85,Fit91,Fit02] used this approach to characterize all major 2-, 3- and 4-valued semantics of logic programs, specifically, supported-model semantics [Cla78], stable-model semantics [GL88], Kripke-Kleene semantics [Fit85,Kun87] and well-founded semantics [VRS91], in terms of fixpoints of operators on the bilattice of 4-valued

⋆ This is a full version of the extended abstract published in the Proceedings of KR'2002 as [DMT02].

interpretations [Fit02]. Similar methods were used in [Lif90,BS91,DMT99] to study the semantics of default logic [Rei80] and autoepistemic logic [Moo84].

In [DMT00a], we studied fixpoint principles behind semantics of these logics in a more abstract algebraic setting of *arbitrary* complete lattices. To get information on fixpoints of an operator O on a complete lattice L, we introduced and studied the notion of an *approximating operator* for O. Approximating operators are operators on the product bilattice L^2. They are designed to approximate the behavior of O. Using purely algebraic techniques, given an approximating operator A for O, we introduced the stable operator and the concepts of the Kripke-Kleene, well-founded and stable fixpoints of A. We also showed how these fixpoints provide information about fixpoints of the operator O.

We discussed two key applications of the theory of approximating operators and their fixpoints. First, we showed that it generalizes the results on semantics of logic programs described in [Fit02]. Specifically, we observed that the 4-valued immediate consequence operator $\mathcal{T}_P$ of a logic program is an approximating operator for the 2-valued immediate consequence operator T_P and showed that all the semantics considered by Fitting can be derived from $\mathcal{T}_P$ by means of general algebraic constructions described in [DMT00a].

We used the *same* algebraic principles also for default and autoepistemic logics. In [DMT00b,DMT03], for a given default or autoepistemic theory we defined its characteristic approximating operator on the bilattice of possible-world structures. By applying our theory of fixpoints of approximating operators, we obtained families of different semantics for both logics, including all major known semantics for these logics and, in addition, some new ones. However, the approximation theory did more than just provide a uniform approach to existing and new semantics for default and autoepistemic logics. It also provided insights into fundamental relations between them. We showed that a default theory Δ and the autoepistemic theory $\pi(\Delta)$, obtained by applying Konolige's interpretation of defaults as modal formulas [Kon88], have *identical* characteristic operators. Consequently, there is a one-to-one correspondence between *families* of fixpoint semantics of default and autoepistemic logics and, under this correspondence, Konolige's interpretation is equivalence preserving. Different semantics formalize different "dialects" of default or autoepistemic reasoning. These dialects can be aligned so that formalizations of nonmonotonic reasoning in default and autoepistemic logics coincide. In other words, Konolige's mapping establishes a perfect match between default and autoepistemic logics, *once proper semantics on each side are identified and correctly aligned*. Our paper [DMT03] provides a detailed study and discussion of the relationship between the two logics.

These results demonstrate that the algebraic framework developed in [DMT00a]

is an effective tool in studies of semantics of knowledge representation formalisms. However, there are at least two directions in which the theory of [DMT00a] can be improved.

First, one should explore the issue of motivations and intuitions behind the approximation theory as presented in [DMT00a]. The problem is that only a fraction of the lattice L^2 (of pairs of elements of L) has a natural interpretation as approximations. Specifically, we regard a pair $(x, y) \in L^2$ as an approximation to all those elements $z \in L$ for which $x \leq z \leq y$. That is, x is a lower estimate and y is an upper estimate for each such z. The problem is that this interpretation makes sense only for *consistent* pairs (x, y), that is, pairs such that $x \leq y$. Inconsistent pairs lack such semantic intuitions [1]. In Section 3, we show that as far as stable and well-founded fixpoints are concerned, the theory developed in [DMT00a] can be reformulated in an equivalent way in terms of the so called *consistent approximating operators* concerned with consistent pairs only. Not only this result completes the theory from [DMT00a] but it also provides a more natural setting for the study of the precision of approximating operators, the main focus of the remaining part of this paper.

Second, the use of the theory developed in [DMT00a] in studies of fixpoints of a lattice operator $O : L \to L$ requires that one is *given* some approximating operator $A : L^2 \to L^2$ for O. In the context of logic programming and default and autoepistemic logic, this approximating operator turns up rather naturally, by using a 4-valued truth evaluation scheme rather than the 2-valued one. There is no *a priori* reason that such natural approximations will emerge in other applications. Given an operator $O : L \to L$, the theory described in [DMT00a] does not provide any principled way for selection of an approximating operator. Thus, the following questions arise:

- Does every lattice operator has an approximation?
- If an operator has several different approximations, how do their fixpoints relate to each other?
- What criteria to use to discriminate between approximations? Is there a natural way to (partially) order approximations? In particular, does there exist a *best* approximation?

In order to turn approximation theory into a full-fledged algebraic fixpoint theory, these questions need to be resolved. This is the other major goal of this paper. To this end, we will study the family of all approximations of a lattice operator and introduce the notion of the *precision* of an approximation. We will show that more precise approximations have more precise Kripke-Kleene and well-founded fixpoints, and have more stable models. We will show that

[1] It is important to note, though, that under other interpretations of bilattice elements, for instance, when they represent information coming from multiple sources, inconsistent bilattice elements can be given a natural intuitive account [Fit91].

the family of approximations of each lattice operator O contains a unique *most precise* approximation of O. We will call it the *ultimate approximation* of O. Since the ultimate approximation is determined by O, it is well suited for investigations of fixpoints of O and yields concepts of ultimate stable fixpoints, the ultimate Kripke-Kleene fixpoint and the ultimate well-founded fixpoint. To define these fixpoints we only need to know O. There is no need to specify any particular approximating operator.

The two goals outlined above form the core of our paper. To show the applicability of the concept of an ultimate approximation we investigate the consequences of our expanded theory in logic programming. We compare our new ultimate semantics of logic programs with the corresponding "standard" semantics. In particular, we show that the ultimate Kripke-Kleene and the ultimate well-founded semantics are more precise and have more attractive properties from the logic perspective than the standard Kripke-Kleene and well-founded semantics. The higher accuracy comes, however, at a cost. We show that ultimate semantics are in general computationally more complex. However, we demonstrate that for wide classes of programs, including programs likely to occur in practice, the complexity of main computational problems remains the same as in the case of corresponding standard semantics.

In summary, our contributions are as follows. We demonstrate that the theory of approximating operators can be developed entirely in terms of consistent approximations. Within that context, we develop a *principled* way of deriving an approximation to a lattice operator. In this way, we obtain concepts of Kripke-Kleene fixpoint, well-founded fixpoint and stable fixpoints that are determined by the operator O and not by the choice of an approximation. In the specific context of logic programming with negation we obtain new semantics with desirable logical properties and possible computational applications.

2 Preliminaries

In this section we recall basic algebraic concepts underlying our work. We assume that the reader is familiar with the concept of a partially ordered set (poset). For a poset $\langle L, \leq \rangle$ and a set of elements $X \subseteq L$, by $\bigvee X$ and $\bigwedge X$ we denote the least upper bound and the greatest lower bound of X in L, respectively. We note that, in general, these bounds are not guaranteed to exist. We also recall that posets are, by definition, nonempty.

A lattice is a poset $\langle L, \leq \rangle$ such that every pair of elements $x, y \in L$ has a unique greatest lower bound and least upper bound, denoted $x \wedge y$ and

$x \vee y$, respectively [2]. A lattice is *complete* if *every* subset has a greatest lower bound and a least upper bound. If S is a subset of a complete lattice L, we denote the least upper bound and the greatest lower bound of S by $\bigvee S$ and $\bigwedge S$, respectively. It is clear that a complete lattice has a least and a greatest element, denoted $\bot$ and $\top$, respectively.

Let L be a poset. For any two elements $x, y \in L$, we define $[x, y] = \{z \in L\colon x \leq z \leq y\}$. We note that this set is non-empty if and only if $x \leq y$. If L is a complete lattice and $x \leq y$, then $[x, y]$ forms a complete lattice, too (under the ordering relation obtained by restricting the relation $\leq$ on L to $[x, y]$).

Let L be a poset. By an operator on L we mean any function $O : L \to L$. For an operator O on L and for a subset $S \subseteq L$, we define $O(S) = \{O(x) : x \in S\}$. An operator O on a poset L is monotone if for every $x, y \in L$ such that $x \leq y$, $O(x) \leq O(y)$. An operator O on a poset L is *increasing* if for every $x \in L$, $x \leq O(x)$.

Let O be an operator on a poset L. An element x of an operator O is a *pre-fixpoint* of O if $O(x) \leq x$; x is a *post-fixpoint* of O if $O(x) \geq x$; x is a *fixpoint* of O if $O(x) = x$. If the set of fixpoints of O has a least element, we call this element the *least fixpoint of* O and denote it by $lfp(O)$.

One of the most fundamental results used in studies of semantics of logics and programming languages is a theorem of Tarski and Knaster on fixpoints of monotone operators on complete lattices [Tar55]. Among other things, it is concerned with the existence of the least fixpoint and the least pre-fixpoint of such operators. We recall here this fragment of the Tarski-Knaster result.

Theorem 2.1 *Let L be a complete lattice and let O be a monotone operator on L. Then O has a least fixpoint and a least pre-fixpoint, and these two elements of L coincide. That is, we have $lfp(O) = \bigwedge\{x \in L\colon O(x) \leq x\}$.*

In this paper, we need a generalization of Theorem 2.1 to a broader class of algebraic structures: chain-complete posets. A chain C in a poset L is a linearly ordered subset of L. A poset L is *chain-complete* if it contains a least element $\bot$ and if every chain C of elements of L has a least upper bound. Clearly, each complete lattice is a chain-complete poset. The converse, in general, does not hold. Chain-complete posets are discussed in detail in [Mar76].

The Tarski-Knaster result (to be precise, the fragment we stated above) holds also in the context of chain-complete posets. Let O be a monotone operator on a chain-complete poset L. Let us define a sequence of elements of L by transfinite induction as follows:

[2] Throughout the paper, whenever the ordering relation $\leq$ of a poset (lattice) is clear from the context, we drop it from the notation.

(1) $c_0 = \bot$
(2) $c_{\alpha+1} = O(c_\alpha)$
(3) $c_\alpha = \bigvee\{c_\beta \colon \beta < \alpha\}$, for a limit ordinal α.

One can show that this sequence is well defined, that it has in L a least upper bound, and that this least upper bound is the least fixpoint and the least pre-fixpoint of O. The argument can be summarized in the following constructive generalization of Theorem 2.1.

Theorem 2.2 ([Mar76]) *Let L be a chain-complete poset and let O be a monotone operator on L. Then the sequence $\{c_\alpha\}_{\alpha \in Ord}$ has a least upper bound and it is the least fixpoint and the least pre-fixpoint of O.*

Let L be a complete lattice. By the *product bilattice* [Gin88] of L we mean the set $L^2 = L \times L$ with the following two orderings $\leq_p$ and $\leq$:

(1) $(x, y) \leq_p (x', y')$ if $x \leq x'$ and $y' \leq y$
(2) $(x, y) \leq (x', y')$ if $x \leq x'$ and $y \leq y'$.

Both orderings are complete lattice orderings in L^2. In this paper we are mostly concerned with the ordering $\leq_p$.

An element $(x, y) \in L^2$ is *consistent* if $x \leq y$. We can think of a consistent element $(x, y) \in L^2$ as an *approximation* to every $z \in L$ such that $x \leq z \leq y$. With this interpretation in mind, the ordering $\leq_p$, when restricted to consistent elements, can be viewed as a *precision* ordering. Consistent pairs that are "higher" in the ordering $\leq_p$ provide tighter approximations. Maximal consistent elements with respect to $\leq_p$ are pairs of the form (x, x). We call approximations of the form (x, x) — *exact*. We note that the consistent elements of L^2 are exactly those for which $[x, y]$ is non-empty.

The *approximation* interpretation of bilattice elements and the corresponding intuition of the *precision* ordering guide our work here. This is the reason why we depart from the more common notation for the bilattice orderings, $\leq_k$ and $\leq_t$, respectively [Fit91]. As an aside, we note that elements of bilattices can be given several alternative interpretations [Fit91], in which they describe "degree of belief" and "degree of doubt" ("evidence for" and "against") or represent information coming from multiple agents (sources).

For a pair $(x, y) \in L^2$, we define its *projections* as:

$$(x, y)_1 = x \qquad \text{and} \qquad (x, y)_2 = y.$$

Given a set $C \subseteq L^2$, we define two subsets of L, $C_1 = \{z_1 : z \in C\}$ and $C_2 = \{z_2 : z \in C\}$ as the *projections* of C on the first and the second coordinate, respectively. When an operator A maps the bilattice L^2 into L^2 we simplify

notation and denote the first projection of the value of the operator A on the pair (x, y) by $A(x, y)_1$ instead of more formal $(A(x, y))_1$. Likewise, we write $A(x, y)_2$ instead of $(A(x, y))_2$.

An operator $A : L^2 \to A^2$ induces two families of operators from L to L as follows. Given an element $b \in A$, we define the operator $A(\cdot, b)_1$ as follows. To every element $a \in L$, the operator $A(\cdot, b)_1$ assigns the value $A(a, b)_1$. Analogously, with a fixed $a \in L$, we define the operator $A(a, \cdot)_2$ as the operator that assigns the value $A(a, b)_2$ to every element $b \in L$.

We denote the set of all consistent pairs in L^2 by L^c. The set $\langle L^c, \leq_p \rangle$ is not a lattice. In particular, a pair of different exact elements has no upper bound in L^c. In fact, exact pairs are maximal elements in L^c.

The structure $\langle L^c, \leq_p \rangle$ is, however, a chain-complete poset. Indeed, the element $(\bot, \top)$ is the least element in L^c and the following straightforward result shows that every chain in L^c has a least upper bound in L^c.

Proposition 2.3 *Let L be a complete lattice and let C be a chain in L^c (ordered by the relation $\leq_p$). Let C_1, C_2 be the projections of C. Then:*

(1) $\bigvee C_1 \leq \bigwedge C_2$
(2) The least upper bound of C exists, and is equal to $(\bigvee C_1, \bigwedge C_2)$.

3 Consistent approximations

In this section we develop a new formalization of the approximation theory based on the structure L^c rather than on L^2. In Section 4, we will show that this new approach is equivalent to the original one as long as we are interested in consistent fixpoints only.

An operator $A : L^c \to L^c$ is a *consistent approximating* operator if it is $\leq_p$-monotone and for every $x \in L$, $A(x, x)_1 = A(x, x)_2$, that is, A assigns exact pairs to exact pairs. To simplify the notation, in the next two sections we will use the term approximating operator rather than *consistent* approximating operator.

We denote the set of all consistent approximating operators on L^c by $Appx(L^c)$. Let $A \in Appx(L^c)$. Since A is $\leq_p$-monotone and L^c is chain-complete, A has a least fixpoint, called the *Kripke-Kleene fixpoint* of A ($k(A)$, in symbols) [3].

[3] We use the term Kripke-Kleene fixpoint to acknowledge an analogy with a Kripke-Kleene model of a logic program (a least fixpoint of a 3-valued van Emden-Kowalski operator).

Directly from the definition it follows that $k(A)$ approximates all fixpoints of A. That is, for every fixpoint (x, y) of A we have $k(A) \leq_p (x, y)$.

An operator $A \in Appx(L^c)$ approximates an operator $O\colon L \to L$ (is an approximation of O) if for every $x \in L$, $A(x, x)_1 = O(x) = A(x, x)_2$. It is easy to see that when A approximates O, then for each $(x, y) \in L^c$ and for each $z \in [x, y]$, $A(x, y)_1 \leq O(z) \leq A(x, y)_2$. That is, $O([x, y]) \subseteq [A(x, y)_1, A(x, y)_2]$. We denote the set of all consistent approximations of O by $Appx^c(O)$.

Properties of the fixpoints of an operator O can be studied by considering fixpoints of approximations of O. Indeed, we have the following two simple results.

Proposition 3.1 *If O is an operator on a complete lattice L and A is an approximation of O, then $x \in L$ is a fixpoint of O if and only if (x, x) is a fixpoint of A.*

Corollary 3.2 *If O is an operator on a complete lattice L and A is an approximation of O, then for every fixpoint x of O, $k(A)_1 \leq x \leq k(A)_2$ (that is, $k(A)$ approximates x).*

In applications, we are usually not interested in all fixpoints of the operator O. For instance, if O is a monotone operator on L, it is common to focus attention on the least fixpoint of O, as it can be given an effective characterization and captures intuitions behind inductive definitions and computational processes. In the case of operators determined by theories in nonmonotonic logics, we are often interested in fixpoints that satisfy some minimality condition. While the Kripke-Kleene fixpoint of an approximating operator of O approximates all fixpoints of O, when we are interested in special classes of fixpoints only, better approximations are possible. Below, based on constructiveness and minimality principles developed in knowledge representation and logic programming, we identify an important class of *stable* fixpoints of an operator O and introduce techniques to obtain more refined approximations of these fixpoints.

An operator A from $Appx(L^c)$ provides means to revise consistent approximations: given a pair $(a, b) \in L^c$, $A(a, b)$ can be viewed as a revision of (a, b). Of particular interest are those pairs whose revisions are at least as accurate. We call an approximation (a, b) *A-reliable* if it is a post-fixpoint of A, that is, if $(a, b) \leq_p A(a, b)$. When A is an approximating operator for a lattice operator O, then A-reliable pairs (a, b) are especially useful for studying fixpoints of O. Indeed, as we will show below in Proposition 3.5, such pairs represent intervals where O behaves in a local way: for each $z \in [a, b]$, $O(z) \in [a, b]$. Consequently, we can obtain a tighter bound on the fixpoints of O in $[a, b]$, simply by the iterated application of A on (a, b). By the $\leq_p$-monotonicity of A, $A(a, b)$ is also A-reliable and approximates all fixpoints of O in $[a, b]$. Iterating the operator A over (a, b) yields a (transfinite) sequence

$(a,b) \leq_p A^1(a,b) \leq_p \ldots \leq_p A^\alpha(a,b) \leq_p \ldots$ of approximations of increasing precision approximating the fixpoints of O in $[a,b]$. In particular, $(\bot,\top)$ is A-reliable and approximates all fixpoints of O. The corresponding increasing sequence of approximations generates the least fixpoint $k(A)$ of A, which is a better approximation to the fixpoints of O than $(\bot,\top)$.

Proposition 3.3 *Let L be a complete lattice and $A \in Appx(L^c)$. If $(a,b) \in L^c$ is A-reliable then, for every $x \in [\bot,b]$, $A(x,b)_1 \in [\bot,b]$ and, for every $x \in [a,\top]$, $A(a,x)_2 \in [a,\top]$.*

Proof: Let $x \in [\bot,b]$. Then $(x,b) \leq_p (b,b)$. By the $\leq_p$-monotonicity of A,

$$A(x,b)_1 \leq A(b,b)_1 = A(b,b)_2 \leq A(a,b)_2 \leq b.$$

The last inequality follows from the fact that (a,b) is A-reliable. The second part of the assertion can be proved in a similar manner. □

Proposition 3.3 implies that for every A-reliable pair (a,b), the restrictions of $A(\cdot,b)_1$ to $[\bot,b]$ and $A(a,\cdot)_2$ to $[a,\top]$ are in fact operators on $[\bot,b]$ and $[a,\top]$, respectively. Moreover, they are $\leq$-monotone operators on the posets $\langle[\bot,b],\leq\rangle$ and $\langle[a,\top],\leq\rangle$. Since $\langle[\bot,b],\leq\rangle$ and $\langle[a,\top],\leq\rangle$ are complete lattices, the operators $A(\cdot,b)_1$ and $A(a,\cdot)_2$ have least fixpoints in the lattices $\langle[\bot,b],\leq\rangle$ and $\langle[a,\top],\leq\rangle$, respectively. We define:

$$b^{A\downarrow} = lfp(A(\cdot,b)_1) \quad \text{and} \quad a^{A\uparrow} = lfp(A(a,\cdot)_2).$$

We call the mapping $(a,b) \mapsto (b^{A\downarrow}, a^{A\uparrow})$, the *stable revision operator for A*. When A is clear from the context, we will drop the reference to A from the notation.

The stable revision operator for A provides a different (but related) way to revise A-reliable approximations than that given by A. However, there is a caveat. The image $(b^{A\downarrow}, a^{A\uparrow})$ of (a,b) under the stable operator does not have, in general, to be a refinement of (a,b).

We will now discuss intuitions behind the definition of the stable revision operator for A. First, we motivate the computation of $b^\downarrow$. Our goal here is to produce a lower bound to all fixpoints of O that are smaller than or equal to b. To this end, we use the approximating operator A. Clearly, $b^0 = \bot$ provides such a bound. By $\leq_p$-monotonicity of A, the operator $A(\cdot,b)_1$ is $\leq$-monotone. Thus, for every fixpoint $x \in [\bot,b]$ of O

$$b^1 = A(\bot,b)_1 \leq A(x,b)_1 \leq A(x,x)_1 = O(x) = x.$$

and every fixpoint $x \in [\bot,b]$ of O belongs, in fact, to the interval $[b^1,b]$. A similar reasoning shows that $b^2 = A(b^1,b)_1$ is a further improvement on the bound for all fixpoints $x \in [\bot,b]$ of O. Continuing this construction we

find that the limit $b^{\downarrow}$ is also a bound. This argument provides a proof of the following result.

Proposition 3.4 *Let A be an approximating operator for an operator O and let (a, b) be A-reliable. For every fixpoint c of O, if $c \leq b$ then $b^{\downarrow} \leq c$.*

The new upper bound $a^{\uparrow}$ is computed to reflect a different intuitive requirement. In the computation of the new upper bound, we are not concerned with preserving all fixpoints greater than a because in this way, we could never eliminate non-minimal fixpoints. On the contrary, we want to select a new upper bound that is as small (tight) as possible. However, any new upper bound b' associated to a should at least satisfy the requirement that the interval $[a, b']$ be closed under application of O. This is the case, in particular, when (a, b') is reliable. Indeed, we have the following simple proposition.

Proposition 3.5 *Let A be an approximating operator for an operator O and let (a, b) be A-reliable. For every $x \in [a, b]$, $O(x) \in [a, b]$. In other words, $[a, b]$ is closed under O.*

Proof: Since $(a, b) \leq_p A(a, b)$, we have $a \leq A(a, b)_1$. Next, since $(a, b) \leq_p (x, x)$, $A(a, b)_1 \leq A(x, x)_1 = O(x)$. The inequality $O(x) \leq b$ can be argued in the same way. □

Interestingly, the set of elements b' of L such that (a, b') is reliable, has a least element and it is easy to show that this element is $lfp(A(a, \cdot)_2) = a^{\uparrow}$. We select this element as the new upper estimate produced by the stable operator when applied to (a, b).

We now study the properties of the stable revision operator.

Proposition 3.6 *Let $A \in Appx(L^c)$. For every A-reliable pair (a, b), $b^{\downarrow} \leq b$, $a \leq a^{\uparrow} \leq b$, and the pair $(b^{\downarrow}, a^{\uparrow})$ is consistent.*

Proof: Inequalities $b^{\downarrow} \leq b$ and $a \leq a^{\uparrow}$ follow directly from the definition of the stable revision operator.

By A-reliability of (a, b), we also have that $A(a, b)_2 \leq b$. Since $a \leq b$, b is in the domain of the operator $A(a, \cdot)_2$ and, moreover, it is a pre-fixpoint of this operator. Thus, since $a^{\uparrow}$ is the least pre-fixpoint of $A(a, \cdot)_2$, $a^{\uparrow} \leq b$.

In particular, we have that $a^{\uparrow}$ is in the domain of the operator $A(\cdot, b)_1$. By the $\leq_p$-monotonicity of A we obtain:

$$A(a^{\uparrow}, b)_1 \leq A(a^{\uparrow}, a^{\uparrow})_1 = A(a^{\uparrow}, a^{\uparrow})_2 \leq A(a, a^{\uparrow})_2 = a^{\uparrow}. \quad (1)$$

It follows that $a^{\uparrow}$ is a pre-fixpoint of the operator $A(\cdot, b)_1$. Thus, $b^{\downarrow} = lfp(A(\cdot, b)_1) \leq a^{\uparrow}$ and so, $(b^{\downarrow}, a^{\uparrow})$ is consistent. □

The notion of A-reliability is not strong enough to guarantee desirable properties of the stable revision operator. For instance, if $(a, b) \in L^c$ is A-reliable, it is not true in general that $(a, b) \leq_p (b^{\downarrow}, a^{\uparrow})$. There is, however, a class of A-reliable pairs for which this property holds. An A-reliable approximation (a, b) is *A-prudent* if $a \leq b^{\downarrow}$. Proposition 3.4 implies that if (a, b) is *A-prudent*, then $[a, b]$ is guaranteed to contain all fixpoints of the lattice operator O approximated by A that belong to $[\bot, b]$. In particular, it contains all minimal fixpoints that belong to $[\bot, b]$. We will now prove several basic properties of A-prudent approximations.

Proposition 3.7 *Let L be a complete lattice, $A \in Appx(L^c)$ and let $(a, b) \in L^c$ be A-prudent. Then, $(b^{\downarrow}, a^{\uparrow})$ is A-prudent and $(a, b) \leq_p (b^{\downarrow}, a^{\uparrow})$.*

Proof: By Proposition 3.6, we have that $b^{\downarrow} \leq b$ and $a \leq a^{\uparrow}$. Since (a, b) is A-prudent, it follows that $a \leq b^{\downarrow}$. From Proposition 3.6 it also follows that $a^{\uparrow} \leq b$. Thus, $(a, b) \leq_p (b^{\downarrow}, a^{\uparrow})$.

Next, let us observe that $b^{\downarrow} = A(b^{\downarrow}, b)_1 \leq A(b^{\downarrow}, a^{\uparrow})_1$. Similarly, $a^{\uparrow} = A(a, a^{\uparrow})_2 \geq A(b^{\downarrow}, a^{\uparrow})_2$. Thus, the pair $(b^{\downarrow}, a^{\uparrow})$ is reliable.

Lastly, we note that for every $x \in [\bot, a^{\uparrow}]$, $A(x, b)_1 \leq A(x, a^{\uparrow})_1$. It follows that each pre-fixpoint of $A(\cdot, a^{\uparrow})_1$ is a pre-fixpoint of $A(\cdot, b)_1$. By (1) (cf. the proof of Proposition 3.6), $A(a^{\uparrow}, a^{\uparrow})_1 \leq a^{\uparrow}$. Thus, the set of pre-fixpoints of $A(\cdot, a^{\uparrow})_1$ is non-empty. Consequently, $b^{\downarrow} = \mathit{lfp}(A(\cdot, b)_1) \leq \mathit{lfp}(A(\cdot, a^{\uparrow})_1) = (a^{\uparrow})^{\downarrow}$ and $(b^{\downarrow}, a^{\uparrow})$ is A-prudent. □

We recall that an A-reliable pair (a, b) is revised by an operator A into a more precise approximation $A(a, b)$. An A-prudent pair (a, b) can be "revised even more" by the stable revision operator.

Proposition 3.8 *Let $A \in Appx(L^c)$. If (a, b) is A-prudent then $A(a, b) \leq_p (b^{\downarrow}, a^{\uparrow})$.*

Proof. It is easy to see that $A(a, b)_1 \leq A(b^{\downarrow}, b)_1 = b^{\downarrow}$ and $a^{\uparrow} = A(a, a^{\uparrow})_2 \leq A(a, b)_2$. Thus, the assertion follows. □

The stable revision operator satisfies a useful monotonicity property.

Proposition 3.9 *Let L be a complete lattice, $A \in Appx(L^c)$ and let $(a, b), (c, d) \in L^c$. If (a, b) is A-reliable, (c, d) is A-prudent and if $(a, b) \leq_p (c, d)$, then $(b^{\downarrow}, a^{\uparrow}) \leq_p (d^{\downarrow}, c^{\uparrow})$.*

Proof: Clearly, we have $d^{\downarrow} \leq c^{\uparrow} \leq d \leq b$. By the $\leq_p$-monotonicity of A, it follows that $A(d^{\downarrow}, b)_1 \leq A(d^{\downarrow}, d)_1 = d^{\downarrow}$. Thus, $d^{\downarrow}$ is a pre-fixpoint of $A(\cdot, b)_1$. Since $b^{\downarrow}$ is the least fixpoint of $\mathit{lfp}(A(\cdot, b)_1)$, it follows that $b^{\downarrow} \leq d^{\downarrow}$.

To prove the assertion, it now suffices to show that $c^\uparrow \leq a^\uparrow$. Let $u = a^\uparrow \wedge d^\downarrow$. By Proposition 3.7, $(c,d) \leq_p (d^\downarrow, c^\uparrow)$. Since $(a,b) \leq_p (c,d)$, it follows that $a \leq d^\downarrow$. Further, by the A-reliability of (a,b) and (c,d), we have $a \leq a^\uparrow$ and $d^\downarrow \leq d$ (Proposition 3.6). Thus, $a \leq u \leq a^\uparrow$ and $u \leq d^\downarrow \leq d$. Consequently,

$$A(u,d)_1 \leq A(u,u)_1 = A(u,u)_2 \leq A(a,a^\uparrow)_2 = a^\uparrow$$

and

$$A(u,d)_1 \leq A(d^\downarrow,d)_1 = d^\downarrow.$$

It follows that $A(u,d)_1 \leq a^\uparrow \wedge d^\downarrow = u$. In particular, u is a pre-fixpoint of $A(\cdot,d)_1$. Since $d^\downarrow$ is the least fixpoint of $A(\cdot,d)_1$, $d^\downarrow \leq u$. Hence, $d^\downarrow \leq a^\uparrow$.

We now have $a \leq c \leq d^\downarrow \leq a^\uparrow$ (the first inequality follows from the assumption $(a,b) \leq_p (c,d)$, the second one follows by Proposition 3.7 from the assumption that (c,d) is A-prudent). Since $a \leq c \leq a^\uparrow$, the $\leq_p$-monotonicity of A implies

$$A(c,a^\uparrow)_2 \leq A(a,a^\uparrow)_2 = a^\uparrow.$$

Hence, $a^\uparrow$ is a pre-fixpoint of $A(c,\cdot)_2$. Since $c^\uparrow$ is the least fixpoint of $A(c,\cdot)_2$, it follows that $c^\uparrow \leq a^\uparrow$. □

The next result states that the limit of a chain of A-prudent pairs is A-prudent.

Proposition 3.10 *Let L be a complete lattice, $A \in Appx(L^c)$ and let C be a chain of A-prudent pairs from L^c. Then, $\bigvee C$ is A-prudent.*

Proof: Let $C_1 = \{p_1 : p \in C\}$ and $C_2 = \{p_2 : p \in C\}$ be the projections of C. We define $a^\infty = \bigvee C_1$ and $b^\infty = \bigwedge C_2$. By Proposition 2.3, (a^∞, b^∞) is consistent and $(a^\infty, b^\infty) = \bigvee C$. Let $(x,y) \in C$. Then, (x,y) is A-reliable (since it is A-prudent) and $(x,y) \leq_p (a^\infty, b^\infty)$. Combining these two observations and using the $\leq_p$-monotonicity of A, we obtain:

$$(x,y) \leq_p A(x,y) \leq_p A(a^\infty, b^\infty).$$

By the fact that (x,y) is an arbitrary element in C, we have

$$(a^\infty, b^\infty) = \bigvee C \leq_p A(a^\infty, b^\infty),$$

and it follows that (a^∞, b^∞) is A-reliable.

Let us now consider an element $x \in C_1$. Then there is an element $y \in C_2$ such that $(x,y) \in C$. Clearly, $b^\infty \leq y$ and, by the $\leq_p$-monotonicity of A, for every $z \leq b^\infty$

$$A(z,y)_1 \leq A(z,b^\infty)_1.$$

Thus, if $z \in [\bot, b^\infty]$ is a pre-fixpoint of the operator $A(\cdot,b^\infty)_1$, it is also a pre-fixpoint of the operator $A(\cdot,y)_1$. Moreover, the set of pre-fixpoints of the operator $A(\cdot,b^\infty)_1$ is nonempty (since (a^∞,b^∞) is A-reliable, it is in the domain

of the stable revision operator for A and $(b^\infty)^\downarrow$ is a pre-fixpoint of $A(\cdot, b^\infty)_1)$. Thus,

$$lfp(A(\cdot, y)_1) \leq lfp(A(\cdot, b^\infty)_1).$$

Since (x, y) is A-prudent, we have $x \leq y^\uparrow = lfp(A(\cdot, y)_1)$. Thus, $x \leq lfp(A(\cdot, b^\infty)_1)$. Since x is an arbitrary element of C_1, $a^\infty \leq lfp(A(\cdot, b^\infty)_1)$. It follows that the pair (a^∞, b^∞) is A-prudent. □

The following theorem summarizes the results on the properties of the stable revision operator.

Theorem 3.11 *Let L be a complete lattice, $A \in Appx(L^c)$. The set of A-prudent elements of L^c is a chain-complete poset under the precision order $\leq_p$, with least element $(\bot, \top)$. The stable revision operator is a well-defined, increasing and monotone operator in this poset.*

It follows that the stable revision operator has fixpoints and a least fixpoint. Let L be a complete lattice and let $A \in Appx(L^c)$. We say that $(x, y) \in L^c$ is a *stable fixpoint* of A if (x, y) is A-reliable (hence, it belongs to the domain of the stable revision operator) and if (x, y) is a fixpoint of the stable revision operator (that is, $x = y^\downarrow$ and $y = x^\uparrow$). We note that a stable fixpoint of A is A-prudent. Moreover, as we show next, stable fixpoints of an approximating operator A are, in particular, fixpoints of A.

Proposition 3.12 *Let L be a complete lattice and let $A \in Appx(L^c)$. If (x, y) is a stable fixpoint of A then (x, y) is a fixpoint of A.*

Proof: Since (x, y) is stable, $x = lfp(A(\cdot, y)_1)$. In particular, $x = A(x, y)_1$. Similarly, $y = A(x, y)_2$. Thus, $A(x, y) = (x, y)$. □

Let O be an operator on a complete lattice L and let $A \in Appx^c(O)$. We say that x is an *A-stable* fixpoint of O if (x, x) is a stable fixpoint of A. The notation is justified as, by Proposition 3.12 and our earlier remarks, every A-stable fixpoint of O is, in particular, a fixpoint of O. The following proposition gives several simple characterizations of A-stable fixpoints of O.

Proposition 3.13 *Let A be an approximating operator for an operator O on a complete lattice L and let $x \in L$. The following conditions are equivalent:*

(1) x is an A-stable fixpoint of O
(2) x is a fixpoint of O and $x = x^\downarrow$
(3) x is a fixpoint of O and $x \leq x^\downarrow$
(4) x is a fixpoint of O and (x, x) is A-prudent.

Proof: We note that each of these conditions guarantees that x is a fixpoint of O and (x, x) a fixpoint of A. Therefore, in each case (x, x) is A-reliable and hence, $x^\downarrow$ is well-defined.

(1)⇒(2) If x is an A-stable fixpoint of O then it is a fixpoint of O. Moreover, (x, x) is a stable fixpoint of A and, consequently, $x = x^{\downarrow}$.
(2)⇒(3) This implication is straightforward.
(3)⇒(4) Since x is a fixpoint of O, $(x, x) = (O(x), O(x)) = A(x, x)$. Hence, (x, x) is A-reliable. Moreover, we have $x \leq x^{\downarrow}$. Consequently, (x, x) is A-prudent.
(4)⇒(1) By Theorem 3.11, (x, x) is a fixpoint of a stable revision operator for A. Since (x, x) is A-prudent and the stable revision operator for A is increasing on the set of A-prudent pairs, (x, x) is a stable fixpoint of A. Thus, x is a A-stable fixpoint of O. □

The next proposition shows that A-stable fixpoints of O are minimal fixpoints of O.

Proposition 3.14 *An A-stable fixpoint x of O is a minimal fixpoint of O. More, generally, a stable fixpoint (x, y) of A is a minimal fixpoint of A with respect to ordering $\leq$ of the product bilattice (the second ordering of the bilattice).*

Proof: Let us assume that (x, y) is a stable fixpoint of A and let $(x', y') \leq_p (x, y)$ be a fixpoint of A. Since $x' \leq y' \leq y$, $A(x', y)_1 \leq A(x', y')_1 = x'$. Thus, x' is a pre-fixpoint of $A(\cdot, y)_1$. Since (x, y) is a stable fixpoint of A, x is the least pre-fixpoint of $A(\cdot, y)_1$. Thus, $x \leq x'$ and, consequently, $x = x'$ (we recall that by our assumption, $x' \leq x$). In a similar way, we argue that $y' = y$. □

Since A-stable fixpoints are A-prudent, we obtain the following corollary to Proposition 3.9.

Corollary 3.15 *Let L be a complete lattice, $A \in Appx(L^c)$ and let $(c, d) \in L^c$ be a stable fixpoint of A. If $(a, b) \in L^c$ is A-reliable and $(a, b) \leq_p (c, d)$ then $(b^{\downarrow}, a^{\uparrow}) \leq_p (c, d)$.* □

The stable revision operator is a monotone operator on the chain-complete poset of A-prudent pairs and has a least fixpoint. Therefore A has a least precise stable fixpoint. We call this least stable fixpoint the *well-founded fixpoint* of A and denote it by $w(A)$. This fixpoint is the limit of the sequence $\{(a^{\alpha}, b^{\alpha})\}_{\alpha \in Ord}$ of elements of L^c defined in by transfinite induction:

(1) $(a^0, b^0) = (\bot, \top)$
(2) $a^{\alpha+1} = (b^{\alpha})^{\downarrow}$ and $b^{\alpha+1} = (a^{\alpha})^{\uparrow}$
(3) $(a^{\alpha}, b^{\alpha}) = \bigvee\{(a^{\beta}, b^{\beta}) \colon \beta < \alpha\}$ for limit ordinals α.

The well-founded fixpoint approximates all stable fixpoints of A. In particular, it approximates all A-stable fixpoints of the operator O. That is, for every A-stable fixpoint x of O, $w(A) \leq_p (x, x)$ or, equivalently, $w(A)_1 \leq x \leq w(A)_2$. Moreover, the well-founded fixpoint is more precise than the Kripke-Kleene

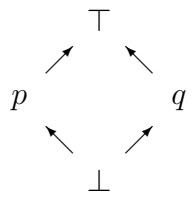

Fig. 1. Lattice L

fixpoint.

Proposition 3.16 *For any approximating operator $A \in Appx(L^c)$, $k(A) \leq_p w(A)$.*

Proof: Since $k(A)$ approximates all fixpoints of A and since $w(A)$ is a fixpoint of A (by Proposition 3.12), the assertion follows. □

We will now illustrate the concepts of A-reliable and A-prudent approximations and stable and well-founded fixpoints. We will also demonstrate that not all minimal fixpoints are stable. Let L be the lattice shown in Figure 1. Let O be an operator in L defined by: $O(\bot) = q, O(q) = q, O(p) = p, O(\top) = p$. It is evident that O has two (minimal) fixpoints: p and q.

We define $A(x, y) = (\bigwedge O([x, y]), \bigvee O([x, y]))$. It is easy to see that A is an approximating operator for O (we will study this construction in a more general setting later in the paper). We give the explicit definition of the operator A in Table 1.

(x, y)	$(\bot, \top)$	$(\bot, p)$	$(\bot, q)$	$(\bot, \bot)$	$(p, \top)$	(p, p)	$(q, \top)$	(q, q)	$(\top, \top)$
$A(x, y)$	$(\bot, \top)$	$(\bot, \top)$	(q, q)	(q, q)	(p, p)	(p, p)	$(\bot, \top)$	(q, q)	(p, p)

Table 1
Operator A.

From Table 1 it is evident that there are five A-reliable pairs in L^c: $(\bot, \top)$, $(\bot, q)$, (q, q), $(p, \top)$ and (p, p). Thus, values $x^\downarrow$ can be computed for $x = p$, q and $\top$, while values $x^\uparrow$ can be computed for $x = \bot$, p and q. These values are given in two tables in Figure 2.

x	p	q	$\top$
$x^\downarrow$	$\bot$	q	$\bot$

x	$\bot$	p	q
$x^\uparrow$	q	p	q

Fig. 2. Values $x^\downarrow$ and $x^\uparrow$

It follows that there are three A-prudent pairs: $(\bot, \top)$, $(\bot, q)$ and (q, q). One can also verify that: $(\top^\downarrow, \bot^\uparrow) = (\bot, q)$, $(q^\downarrow, \bot^\uparrow) = (q, q)$ and $(q^\downarrow, q^\uparrow) = (q, q)$. Thus, (q, q) is the well-founded fixpoint of A (we reached (q, q) by iterating the stable revision operator over $(\bot, \top)$). Since the well-founded fixpoint (q, q)

is complete, (q, q) is also the unique stable fixpoint of A and q is the unique A-stable fixpoint of O. We also note that p, the other minimal fixpoint of O, is not an A-stable fixpoint of O.

4 Approximations in the bilattice

We will now show that the theory of *consistent* approximations captures all results of the theory of approximations in the product bilattice developed in [DMT00a], as long as we restrict our attention to consistent pairs. We start by recalling some basic concepts defined in [DMT00a].

Let L be a complete lattice. An operator $A\colon L^2 \to L^2$ is *symmetric* if for every $(x, y) \in L^2$, $A(x, y)_1 = A(y, x)_2$ (as before, $(\cdot)_1$ and $(\cdot)_2$ are the two projection functions). Further, A is *approximating* if A is symmetric and $\leq_p$-monotone [4]. While it is possible to develop a generalization of the theory presented in this paper without the symmetry assumption, we chose to adopt it because the motivating examples, that is, operators occurring in knowledge representation, are symmetric.

Every approximating operator A on L^2 maps exact pairs to exact pairs (indeed, $A(x, x) = (A(x, x)_1, A(x, x)_2)$ and, by the symmetry of A, $A(x, x)_1 = A(x, x)_2$). If A is an approximating operator and O is an operator on L such that for every $x \in L$ $A(x, x) = (O(x), O(x))$, then A is an *approximating operator for* O. We denote the set of all approximating operators on L^2 by $Appx(L^2)$ and the set of all approximating operators for an operator O on L by $Appx(O)$.

Let $A\colon L^2 \to L^2$ be an approximating operator. It is easy to see that for every $y \in L$, the operator $A(\cdot, y)_1$ (*defined on* L) is $\leq$-monotone. Thus, it has a least fixpoint. For every $y \in L$, we define $C_A(y) = lfp(A(\cdot, y)_1)$ or, equivalently (as A is symmetric), $C_A(y) = lfp(A(y, \cdot)_2)$. We call the operator $\mathcal{C}_A(x, y) = (C_A(y), C_A(x))$ the *stable operator for* A.

In [DMT00a], we proved that all fixpoints of $\mathcal{C}_A$ are also fixpoints of A and called them *stable fixpoints* of A. Furthermore, we proved that $\mathcal{C}_A$ is $\leq_p$-monotone. Thus, it has a least fixpoint (with respect to $\leq_p$). We called this fixpoint the *well-founded* fixpoint of A.

We showed in [DMT00a] that if $A\colon L^2 \to L^2$ is an approximating operator then for every $(x, y) \in L^c$, $A(x, y) \in L^c$. It follows that the restriction of A to

[4] In this section we will always use the term *consistent approximating operator* for approximating operators on L^c and *approximating operator* for approximating operators on L^2.

L^c is an operator on L^c. We will denote this operator by A^c. It follows directly from the relevant definitions that if $A\colon L^2 \to L^2$ is an approximating operator, then A^c is a consistent approximating operator and, since $A(x,x) = A^c(x,x)$, the two operators approximate the same operator on the lattice L.

We will now establish a correspondence between consistent fixpoints of A and fixpoints of A^c. We start with a lemma which shows that on A^c-prudent pairs, the stable operator of A and the stable revision operator of A^c coincide.

Lemma 4.1 *Let A be an approximating operator. A consistent pair (x,y) is A^c-prudent if and only if $(x,y) \leq_p A(x,y)$ and $x \leq C_A(y)$. Moreover, if (x,y) is A^c-prudent then $\mathcal{C}_A(x,y) = (y^{A^c\downarrow}, x^{A^c\uparrow})$.*

Proof: Let us assume that (x,y) is A^c-prudent. Then (x,y) is A^c-reliable and, since $A^c(x,y) = A(x,y)$ for $(x,y) \in L^c$, $(x,y) \leq_p A(x,y)$. Further, $A^c(\cdot,y)_1$ is a monotone operator on $[\bot, y]$. Since for $(x,y) \in L^c$, $A^c(x,y) = A(x,y)$, $lfp(A^c(\cdot,y)_1) = lfp(A(\cdot,y)_1)$. Thus, $x \leq y^{A^c\downarrow} = lfp(A^c(\cdot,y)_1) = lfp(A(\cdot,y)_1) = C_A(y)$.

Conversely, let us assume that $(x,y) \leq_p A(x,y)$ and $x \leq C_A(y)$. Since, $A(x,y) = A^c(x,y)$, it follows that (x,y) is A^c-reliable. Thus, $y^{A^c\downarrow}$ is well defined and, reasoning as before, one can show that $y^{A^c\downarrow} = lfp(A^c(\cdot,y)_1) = lfp(A(\cdot,y)_1) = C_A(y)$. Thus, $x \leq y^{A^c\downarrow}$ and, since (x,y) is A^c-reliable, (x,y) is A^c-prudent.

Next we will prove the second part of the assertion. We proved earlier that if (x,y) is A^c-prudent, then $y^{A^c\downarrow} = C_A(y)$. Let us now observe that, by A^c-reliability of (x,y), $A^c(x,\cdot)_2$ is a monotone operator on $[x, \top]$. Since $x^{A^c\uparrow}$ is a fixpoint of $A^c(x,\cdot)_2$ (viewed as an operator on $[x,\top]$), $(x, x^{A^c\uparrow})$ is consistent and $x^{A^c\uparrow}$ is a fixpoint of $A(x,\cdot)_2$. Thus, $lfp(A(x,\cdot)_2) \leq x^{A^c\uparrow}$ and $C_A(x) = lfp(A(\cdot,x)_1) = lfp(A(x,\cdot)_2) \leq x^{A^c\uparrow}$.

We will now show that $x^{A^c\uparrow} \leq C_A(x)$. By the $\leq_p$-monotonicity of A and since $x \leq y$, for every $u \in L$, $A(u,y)_1 \leq A(u,x)_1$. In particular, if u is a pre-fixpoint of $A(\cdot,x)_1$ (that is, if $A(u,x)_1 \leq u$), u is a pre-fixpoint of $A(\cdot,y)_1$. Thus, $C_A(x)$, which is a pre-fixpoint of $A(\cdot,x)_1$ (the least pre-fixpoint, in fact), is a pre-fixpoint of $A(\cdot,y)_1$. Since $C_A(y)$ is the least pre-fixpoint of $A(\cdot,y)_1$, $C_A(y) \leq C_A(x)$.

Since (x,y) is A^c-prudent, by the first part of the assertion, $x \leq C_A(y) \leq C_A(x)$. Thus, $(x, C_A(x))$ is consistent. By the definition of $C_A(x)$, $C_A(x) = A(C_A(x),x)_1 = A(x, C_A(x))_2 = A^c(x, C_A(x))_2$, the last equality follows by the consistency of $(x, C_A(x))$. Thus, $C_A(x)$ is a fixpoint of $A^c(x,\cdot)_2$ and, consequently, $x^{A^c\uparrow} \leq C_A(x)$. □

The next theorem summarizes the relationship between consistent fixpoints of

an approximating operator A on L^2 and fixpoints of a consistent approximating operator A^c on L^c. In our discussion when we refer to stable fixpoints of an approximating operator, we treat them according to their definition given in [DMT00a] and reviewed above. Similarly, when we refer to stable fixpoints of a consistent approximating operator, we understand them according to their definitions specified in this paper. In addition, with some abuse of notation we write $w(A)$ and $w(A^c)$ to denote well-founded fixpoints of A and A^c, even though formally the definitions are different.

Theorem 4.2 *Let L be a complete lattice and let $A\colon L^2 \to L^2$ be an approximating operator. Then,*

(1) A consistent pair (x, y) is a fixpoint of A if and only if (x, y) is a fixpoint of A^c.
(2) The least fixpoints of A and A^c coincide. In other words, $k(A) = k(A^c)$.
(3) A consistent pair (x, y) is a stable fixpoint of A if and only if it is a stable fixpoint of A^c.
(4) The well-founded fixpoints of A and A^c coincide. In other words, $w(A) = w(A^c)$.

Proof: It is a straightforward consequence of the definitions that the set of consistent fixpoints of A and the set of fixpoints of A^c (which are consistent by definition) coincide. It follows that $lfp(A) \leq_p lfp(A^c)$. Since the set of consistent pairs, L^c forms an initial segment of $\langle L^2, \leq_p \rangle$, $lfp(A)$ is consistent. Consequently, the least fixpoints of A and A^c coincide as well (we recall that we refer to these least fixpoints as Kripke-Kleene fixpoints). Thus, the assertions (1) and (2) follow.
(3) If a consistent pair (x, y) is a stable fixpoint of A, $x = C_A(y)$. By Lemma 4.1, (x, y) is A^c-prudent and a fixpoint of the stable revision operator of A^c. Conversely, if (x, y) is a stable fixpoint of A^c, then it is A^c-prudent and, by Lemma 4.1, it is a stable fixpoint of A.
(4) Clearly, $w(A^c)$ is consistent and so, by (3), it is a stable fixpoint of A. Since $w(A)$ is a least stable fixpoint of A, $w(A) \leq_p w(A^c)$. Consequently, $w(A)$ is consistent. Again by (3), $w(A)$ is a stable fixpoint of A^c. Therefore, $w(A^c) \leq_p w(A)$. Thus, $w(A^c) = w(A)$. □

It follows from the results presented so far that the concept of a consistent stable fixpoint of an approximating operator can be given an equivalent characterization in terms of the definition of a stable fixpoint of the consistent restriction of A, the operator A^c. We will now study the converse problem: can stable fixpoints of a consistent approximating operator (including its well-founded fixpoint) be studied and characterized in the setting of the theory of approximating operators developed in [DMT00a]? To resolve this question we will show that *every* consistent approximating operator A can be viewed as a restriction of some approximating operator B to L^c.

Let $A \in Appx(L^c)$. We say that an operator B on L^2 *extends* A if $A = B^c$, that is, if A is the restriction of B to L^c. We will now study two fundamental questions concerning consistent approximating operators: given a consistent approximating operator A, (1) can A be extended to an approximating operator on the bilattice, and (2) is the extension unique?

We will start by constructing, given a consistent approximating operator C on L^c, two operators C^+ and C^- on L^2, and by showing that each of them is an approximating operator that extends C. To this end, for an element $(x, y) \in L^2$ we define

$$Cons(x, y) = \{(a, b) \in L^c \colon (a, b) \leq_p (x, y)\}.$$

Next, for every $(x, y) \in L^2$, we define $C^+(x, y) = (C^+(x, y)_1, C^+(x, y)_2)$ as follows:

$$C^+(x, y)_1 = \begin{cases} C(x, y)_1 & \text{if } x \leq y \\ \bigwedge\{C(a, b)_2 \colon (a, b) \in Cons(y, x)\} & \text{otherwise.} \end{cases}$$

We complete the definition by setting $C^+(x, y)_2 = C^+(y, x)_1$.

Similarly, we define an operator C^- on L^2 as follows:

$$C^-(x, y)_2 = \begin{cases} C(x, y)_2 & \text{if } x \leq y \\ \bigvee\{C(a, b)_1 \colon (a, b) \in Cons(y, x)\} & \text{otherwise.} \end{cases}$$

As before, we complete the definition by setting $C^-(x, y)_1 = C^-(y, x)_2$. We have the following result.

Theorem 4.3 *Let $C \colon L^c \to L^c$ be a consistent approximating operator. The operators C^- and C^+ are approximating operators on L^2 and each of them extends C.*

Proof: We will prove the result for the operator C^+ only. The case of the operator C^- can be established by a similar argument.

It follows directly from the definition that C^+ is symmetric. We will now show that C^+ is $\leq_p$-monotone. Let $(x, y), (x', y')$ be two elements of L^2 such that $(x, y) \leq_p (x', y')$. Since C^+ is symmetric, to prove $\leq_p$-monotonicity of C^+ it is enough to show that $C^+(x, y)_1 \leq C^+(x', y')_1$. To this end, we will consider the following three cases.
Case 1. Both pairs (x, y) and (x', y') are consistent. In this case,

$$C^+(x, y)_1 = C(x, y)_1 \leq C(x', y')_1 = C^+(x', y')_1.$$

Case 2. The pair (x, y) is consistent and the pair (x', y') is inconsistent. Let $(a, b) \in Cons(y', x')$. Then, since $(x, y) \leq_p (x', y')$,

$$a \leq y' \leq y \quad \text{and} \quad x \leq x' \leq b.$$

Since $x \leq y$ and $a \leq b$, it follows that $a \vee x \leq y$ and $a \vee x \leq b$. Thus,

$$(x, y) \leq_p (a \vee x, a \vee x) \quad \text{and} \quad (a, b) \leq_p (a \vee x, a \vee x).$$

Consequently,

$$C^+(x, y)_1 = C(x, y)_1 \leq C(x \vee a, x \vee a)_1 = C(x \vee a, x \vee a)_2 \leq C(a, b)_2.$$

Thus, since (a, b) is an arbitrary element of $Cons(y', x')$, we have

$$C^+(x, y)_1 \leq \bigwedge\{C(a, b)_2 \colon (a, b) \in Cons(y', x')\} = C^+(x', y')_1.$$

Case 3. Both pairs (x, y) and (x', y') are inconsistent. Since $(x, y) \leq_p (x', y')$, we have $(y', x') \leq_p (y, x)$. Thus, $Cons(y', x') \subseteq Cons(y, x)$ and

$$\begin{aligned} C^+(x, y)_1 &= \bigwedge\{C(a, b)_2 \colon (a, b) \in Cons(y, x)\} \\ &\leq \bigwedge\{C(a, b)_2 \colon (a, b) \in Cons(y', x')\} = C^+(x', y')_1. \end{aligned}$$

The cases (1) - (3) exhaust all possibilities for pairs (x, y) and (x', y'), where $(x, y) \leq_p (x', y')$. Thus, C^+ is $\leq_p$-monotone. To complete the proof, it remains to show that C^+ extends C. Let $(x, y) \in L^c$. By the definition of C^+, we have $C(x, y)_1 = C^+(x, y)_1$. Next, we observe that (x, y) is the greatest element in $Cons(x, y)$. By $\leq_p$-monotonicity of C,

$$C^+(y, x)_1 = \bigwedge\{C(a, b)_2 \colon (a, b) \in Cons(x, y)\} = C(x, y)_2.$$

Thus, $C^+(x, y)_2 = C^+(y, x)_1 = C(x, y)_2$ and consequently, for every $(x, y) \in L^c$, $C^+(x, y) = (C^+(x, y)_1, C^+(x, y)_2) = (C(x, y)_1, C(x, y)_2) = C(x, y)$. □

The two approximating operators C^- and C^+ are not arbitrary. They provide boundaries for the space of approximating operators extending a consistent approximating operator C with respect to the second ordering in the product bilattice L^2, that is, the componentwise extension of the lattice ordering $\leq$ to L^2 (cf. Section 2). By somewhat abusing the notation, we use the same symbol, $\leq$, to denote this ordering of L^2. We now have the following result.

Theorem 4.4 *If C is a consistent approximating operator and A is an approximating operator extending C, then for each $(x, y) \in L^2$, $C^-(x, y)_1 \leq A(x, y)_1 \leq C^+(x, y)_1$ and $C^-(x, y)_2 \leq A(x, y)_2 \leq C^+(x, y)_2$. In other words, in the notation introduced above: $C^-(x, y) \leq A(x, y) \leq C^+(x, y)$.*

Proof: If $x \leq y$, the assertion follows from the fact that C^-, C^+ and A all extend C and, consequently, they all coincide on (x, y). If $x \geq y$ then, by symmetry, C^-, C^+ and A all coincide on (x, y) and the assertion follows in this case, as well.

To complete the proof, let us consider a pair $(x, y) \in L^2$ such that neither $x \leq y$ nor $x \geq y$ holds. Let $(a, b) \in Cons(y, x)$. Since $(a, b) \leq_p (y, x)$ and A is symmetric and $\leq_p$-monotone, we have:

$$A(x, y)_1 = A(y, x)_2 \leq A(a, b)_2 = C^+(a, b)_2$$

(the last equality follows by the fact that A and C^+ coincide on consistent pairs). Since (a, b) is an arbitrary element of $Cons(y, x)$, we obtain

$$A(x, y)_1 \leq \bigwedge\{C^+(a, b)_2 \colon (a, b) \in Cons(y, x)\} = C^+(x, y)_1.$$

The inequality $C^-(x, y)_1 \leq A(x, y)_1$ can be proved in the same way.

The inequalities involving $A(x, y)_2$ follow directly from those of $A(x, y)_1$ and the symmetry of C^-, A and C^+. □

We now turn to the question of the uniqueness of the extension. Theorem 4.4 and its proof imply the following result.

Corollary 4.5 *Let $C \in Appx(L^c)$. If $C^- = C^+$ then there is only one approximating operator in $Appx(L^2)$ that extends C. In particular, if the ordering $\leq$ on L is linear, then $C^- = C^+$ and C has a unique extension to an approximating operator in $Appx(L^2)$.*

Corollary 4.5 exhibits conditions under which a consistent approximating operator admits a unique extension to an approximating operator. In general, however, the extension is not unique. Let L be the four-element lattice shown in Figure 1. We define an operator on L^c as follows:

$$C(x, y) = \begin{cases} (\bot, \top) & \text{if } x < y \\ (x, x) & \text{if } x = y \end{cases}$$

It is evident that the operator C is $\leq_p$-monotone and exact. Thus, C is a consistent approximating operator on L^c.

We will show that the operators C^- and C^+ do not coincide. To this end, we will compute $C^-(p, q)$ and $C^+(p, q)$. First, we observe that $Cons(q, p) = \{(\bot, \top), (\bot, p), (q, \top)\}$. By the definition,

$$C^-(p, q)_2 = C(\bot, \top)_1 \vee C(\bot, p)_1 \vee C(q, \top)_1 = \bot.$$

Similarly, $C^-(p,q)_1 = C^-(q,p)_2 = \bot$. Thus, $C^-(p,q) = (\bot, \bot)$. In the same way,

$$C^+(p,q)_1 = C(\bot, \top)_2 \wedge C(\bot, p)_2 \wedge C(q, \top)_2 = \top$$

and $C^+(p,q)_2 = \top$. Thus, $C^+(p,q) = (\top, \top)$ and, consequently, $C^-(p,q) \neq C^+(p,q)$.

Lack of uniqueness of an extension is not a problem as long as we are interested in consistent fixpoints and stable fixpoints. Since for every two extensions A and B of a consistent approximating operator C, A and B coincide on L^c ($A^c = B^c = C$), Theorem 4.2 implies that consistent fixpoints (including the Kripke-Kleene fixpoints) and consistent stable fixpoints (including the well-founded fixpoints) of A and B coincide. Thus, the choice of a particular extending approximating operator is not essential, as long as we limit our interest to consistent fixpoints. In other words, it follows from Theorems 4.2 and 4.3 that the theories of fixpoints and stable fixpoints developed in this paper and in [DMT00a] are equivalent for the most interesting and important case when consistency of fixpoints is required. Consequently, they are equivalent in their ability to approximate fixpoints of lattice operators.

We end this section with a reflection on differences between the two approaches to the theory of approximating operators. When considering consistent approximations, that is, operators defined on L^c only, we introduce one revision operator to compute a suitable upper bound from a given lower bound, and *another* revision operator, with different underlying intuitions, to compute a suitable lower bounds from a given upper bound. These two operators are partial, that is, defined on only some pairs in L^c. Consequently, proofs often require tedious analysis concerning whether a revision of an approximation is well-defined, or whether it is consistent. The strength of this approach is that all notions have a compelling intuitive appeal.

When we consider approximating operators defined on the bilattice L^2, all technical difficulties mentioned above disappear. We define a *single* stable operator and use it when revising *both* the lower and the upper bounds of the present approximation. As a result the whole theory and, in particular, proofs, get significantly simpler. However, on the intuitive level, this theory gives little insight. It does not explain what motivates the way the stable operator is defined.

The existence of two equivalent theories providing trade-offs between mathematical elegance and intuitive appeal is useful — it allows us to proceed on an intuitive level, knowing that a natural and elegant mathematical constructions can be provided in a properly constructed superstructure.

5 Ultimate approximations

In this section we will study ways to order approximations. To this end, we will again restrict our attention to consistent approximations only. Let $A, B \in Appx(L^c)$. We say that A is *less precise* than B ($A \leq_p B$, in symbols) if for each pair $(x, y) \in L^c$, $A(x, y) \leq_p B(x, y)$. It is easy to see that if $A \leq_p B$ then they approximate the same operator O on the lattice L.

Lemma 5.1 *Let L be a complete lattice and $A, B \in Appx(L^c)$. If $A \leq_p B$ and $(a, b) \in L^c$ is A-prudent then (a, b) is B-prudent and $(b^{A\downarrow}, a^{A\uparrow}) \leq_p (b^{B\downarrow}, a^{B\uparrow})$.*

Proof: Clearly, $(a, b) \leq_p A(a, b) \leq_p B(a, b)$ (the first inequality follows by A-reliability of (a, b)). Thus, (a, b) is B-reliable.

Since (a, b) is both A- and B-reliable, the least fixpoints $b^{A\downarrow}$ and $b^{B\downarrow}$ are well-defined. For each $x \in L$ such that $x \leq b$, if x is a pre-fixpoint of $B(\cdot, b)_1$, then $A(x, b)_1 \leq B(x, b)_1 \leq x$. Consequently, x is a pre-fixpoint of $A(\cdot, b)_1$. It follows that $b^{A\downarrow} \leq b^{B\downarrow}$. Since $a \leq b^{A\downarrow}$, $a \leq b^{B\downarrow}$. Thus (a, b) is B-prudent.

In a similar way one can prove that $a^{B\uparrow} \leq a^{A\uparrow}$. We already proved above that $b^{A\downarrow} \leq b^{B\downarrow}$. Thus, it follows that $(b^{A\downarrow}, a^{A\uparrow}) \leq_p (b^{B\downarrow}, a^{B\uparrow})$. □

More precise approximations have more precise Kripke-Kleene and well-founded fixpoints.

Theorem 5.2 *Let L be a complete lattice and let $A, B \in Appx(L^c)$. If $A \leq_p B$ then $k(A) \leq_p k(B)$ and $w(A) \leq_p w(B)$.*

Proof: Since $A \leq_p B$,

$$A(k(B)) \leq_p B(k(B)) = k(B).$$

Therefore, $k(B)$ is a pre-fixpoint of A. Since $k(A)$ is the least pre-fixpoint of A, it follows that $k(A) \leq_p k(B)$.

To prove the second part of the assertion, let us consider the sequences $\{(a_A^\alpha, b_A^\alpha)\}_{\alpha \in Ord}$ and $\{(a_B^\alpha, b_B^\alpha)\}_{\alpha \in Ord}$ used to define he well-founded fixpoints of A and B, respectively. To prove the assertion, it suffices to show that for every ordinal α, $(a_A^\alpha, b_A^\alpha) \leq_p (a_B^\alpha, b_B^\alpha)$.

Clearly, $(a_A^0, b_A^0) \leq_p (a_B^0, b_B^0)$. Let us assume that $\alpha = \beta{+}1$ and that $(a_A^\beta, b_A^\beta) \leq_p (a_B^\beta, b_B^\beta)$.

By Proposition 3.9,

$$((b_A^\beta)^{B\downarrow}, (a_A^\beta)^{B\uparrow}) \leq_p ((b_B^\beta)^{B\downarrow}, (a_B^\beta)^{B\uparrow}) = (a_B^\alpha, b_B^\alpha).$$

Since (a_A^β, b_A^β) is A-prudent, Lemma 5.1 entails that it is B-prudent and

$$(a_A^\alpha, b_A^\alpha) = ((b_A^\beta)^{A\downarrow}, (a_A^\beta)^{A\uparrow}) \leq_p ((b_A^\beta)^{B\downarrow}, (a_A^\beta)^{B\uparrow}).$$

The case of the limit ordinal α is straightforward. □

The next result shows that as the precision of an approximation grows, all exact fixpoints and exact stable fixpoints are preserved.

Theorem 5.3 *Let L be a complete lattice and let $A, B \in Appx(L^c)$. If $A \leq_p B$ then every exact fixpoint of A is an exact fixpoint of B, and every exact stable fixpoint of A is an exact stable fixpoint of B.*

Proof: Since for every $x \in L$, $A(x, x) = B(x, x)$, the first part of the assertion follows. Let us now assume that (x, x) is an exact stable fixpoint of A. In particular, it follows that (x, x) is a fixpoint of A and is A-prudent. By Lemma 5.1, (x, x) is B-prudent. Consequently, by Proposition 3.13(4), x is a B-stable fixpoint of O. □

Corollary 5.4 *Let L be a complete lattice and let $A, B \in Appx^c(O)$. If $A \leq_p B$. Then every A-stable fixpoint of O is a B-stable fixpoint of O.*

Non-exact fixpoints are not preserved, in general. For instance, let us consider two consistent approximations A and B such that $A \leq_p B$. Let us also assume that $w(A) <_p w(B)$. That is, A has a *strictly* less precise well-founded fixpoint than B. Then, clearly, $w(A)$ is no longer a stable fixpoint of B.

We note that a well-founded fixpoint of an approximation yields a bound on its exact stable fixpoints. It is worth noting that more precise approximations yield a larger set of exact stable fixpoints (Theorem 5.3) and, at the same time, more precise bounds on this set in terms of the well-founded fixpoint (Theorem 5.2). Moreover, the well-founded fixpoint of a more precise operator also provides correct approximations for exact stable fixpoints of a less precise operator.

It is an important question whether there exists an *ultimate approximation* of O, that is, a consistent approximation most precise with respect to the ordering $\leq_p$. This ultimate approximation, if existed, would have a most precise Kripke-Kleene and well-founded fixpoints and a largest set of exact stable fixpoints. Such ultimate approximation, being a distinguished object in the collection of all approximations of O can be viewed as determined by O itself. Consequently, fixpoints of the ultimate approximation of O (including stable, Kripke-Kleene and well-founded fixpoints) could be regarded as determined by O and would be associated with it. We will show that the answer to this key question is positive. That is, we will show that for every operator O on a complete lattice, the set $Appx^c(O)$ has the greatest element with respect to $\leq_p$.

We start by providing a non-constructive argument for the existence of ultimate approximations. Let us note that the set $Appx^c(O)$ is not empty. Indeed, let us define $A_O(x,y) = (O(x), O(x))$, if $x = y$, and $A_O(x,y) = (\bot, \top)$, otherwise. It is easy to see that $A_O \in Appx^c(O)$ and that it is the least precise element in $Appx^c(O)$. Next, we observe that $Appx^c(O)$ with the ordering $\leq_p$ forms a complete lattice, as the set $Appx^c(O)$ is closed under the operations of taking greatest lower bounds and least upper bounds. It follows that $Appx^c(O)$ has a greatest element (most precise approximation). We call this consistent approximation the *(consistent) ultimate approximation* of O and denote it by U_O.

We call the Kripke-Kleene and the well-founded fixpoints of U_O, the *ultimate Kripke-Kleene* and the *ultimate well-founded fixpoint* of O. We denote them by $k(O)$ and $w(O)$, respectively. We call an *exact* stable fixpoint of U_O an *ultimate stable fixpoint* of O. Exact fixpoints of all consistent approximations are the same and correspond to fixpoints of O. Thus, there is no need to introduce the concept of an ultimate exact fixpoint of O. We have the following corollary to Theorems 5.2 and 5.3.

Corollary 5.5 *Let O be an operator on a complete lattice L. For every $A \in Appx^c(O)$ we have:*

(1) $k(A) \leq_p k(O)$ and $w(A) \leq_p w(O)$,
(2) For every A-stable fixpoint x of O, x is an ultimate stable fixpoint of O and $w(O)_1 \leq x \leq w(O)_2$.

We will now provide a constructive characterization of the notion of ultimate approximation. To state the result, for every $x, y \in L$ such that $x \leq y$, we define $O([x,y]) = \{O(z) \colon z \in [x,y]\}$.

Theorem 5.6 *Let O be an operator on a complete lattice L. Then, for every $(x,y) \in L^c$, $U_O(x,y) = (\bigwedge O([x,y]), \bigvee O([x,y]))$.*

Proof: We define an operator $C : L^c \to L^2$ by setting

$$C(x,y) = (\bigwedge O([x,y]), \bigvee O([x,y])).$$

First, let us notice that since $\bigwedge O([x,y]) \leq \bigvee O([x,y])$, the operator C maps L^c into L^c. Moreover, it is easy to see that C is $\leq_p$-monotone. Lastly, since $O([x,x]) = \{O(x)\}$,

$$\bigwedge O([x,x]) = \bigvee O([x,x]) = O(x)$$

and, consequently, $C(x,x) = (O(x), O(x))$. Thus, it follows that C is a consistent approximation of O. Since U_O is the most precise approximation, we have $C \leq_p U_O$.

On the other hand, let $(x, y) \in L^c$ and let $z \in [x, y]$. Then, since U_O is an approximating operator for O, $U_O(x, y) \leq_p (O(z), O(z))$. Thus, $U_O(x, y)_1 \leq O(z)$ and, consequently, $U_O(x, y)_1 \leq \bigwedge O([x, y])$. Similarly, $\bigvee O([x, y]) \leq U_O(x, y)_2$. Since $x \leq y$ are arbitrary, $U_O \leq_p C$, as desired. □

We will now provide characterizations of U_O-reliable pairs and of the stable revision operator for U_O.

Theorem 5.7 *Let O be an operator on a complete lattice L. Then:*

(1) A pair $(a, b) \in L^c$ is U_O-reliable if and only if $O([a, b]) \subseteq [a, b]$
(2) If $(a, b) \in L^c$ is U_O-reliable, $b^{U_O\downarrow}$ is the limit of the following sequence: $a_0 = \bot$, $a_{\alpha+1} = \bigwedge O([a_\alpha, b])$ and $a_\alpha = \bigvee\{a_\beta : \beta < \alpha\}$, for limit α; and $a^{U_O\uparrow}$ is the least b such that $[a, b]$ is U_O-reliable (or, equivalently, such that $O([a, b]) \subseteq [a, b]$).

As a consequence to Theorem 5.6 and Proposition 3.13, we obtain the following characterization of ultimate stable fixpoints of an operator O.

Corollary 5.8 *Let L be a complete lattice. An element $x \in L$ is an ultimate stable fixpoint of an operator $O : L \to L$ if and only if $x = lfp(\bigwedge O([\cdot, x]))$, where we regard $\bigwedge O([\cdot, x])$ as an operator on $[\bot, x]$.*

The following proposition describes the ultimate approximations for monotone and antimonotone operators on L.

Proposition 5.9 *If O is a monotone operator on a complete lattice L then for every $(x, y) \in L^c$, $U_O(x, y) = (O(x), O(y))$. If O is antimonotone then for every $(x, y) \in L^c$, $U_O(x, y) = (O(y), O(x))$.*

Proof: By Theorem 5.6,

$$U_O(x, y) = (\bigwedge O([x, y]), \bigvee O([x, y])).$$

Now, it is easy to see that if O is monotone, then $\bigwedge O([x, y]) = O(x)$ and $\bigvee O([x, y]) = O(y)$. If O is antimonotone, then $\bigwedge O([x, y]) = O(y)$ and $\bigvee O([x, y]) = O(x)$. Thus the proposition follows. □

Using Proposition 5.9 we now obtain the following corollary. The first part of the assertion provides support to our earlier claim that the theory of ultimate approximations generalizes Tarski's least-fixpoint theory of *monotone* lattice operators to the case of arbitrary ones.

Corollary 5.10 *Let O be an operator on a complete lattice L. If O is monotone, then the least fixpoint of O is the ultimate well-founded fixpoint of O and the unique ultimate stable fixpoint of O. If O is antimonotone, then $k(O) = w(O)$ and every fixpoint of O is an ultimate stable fixpoint of O.*

Ultimate approximation operator for a lattice operator O provides us with the most precise estimates on fixpoints O. The question arises then about the role of less precise approximations. We will now argue that in several situations they may be useful. For instance, it may be the case that computing the well-founded and stable fixpoints of some less precise approximation, say A, is more tractable than computing their ultimate counterparts. In such case, in order to compute the ultimate well-founded fixpoint of O we can first compute the well-founded fixpoint of A. The well-founded fixpoint of A is, in general, less precise than the ultimate well-founded fixpoint. However, in some cases the two may coincide. Even if they are not the same, knowing the well-founded fixpoint of A may speed up computation of the ultimate well-founded fixpoint (as the well-founded fixpoint of A provides some information about the ultimate well-founded fixpoint). Similarly, if we just want to find a single ultimate stable fixpoint of O, we might first try to search for an A-stable fixpoint of O. If we find one, it is also an ultimate stable fixpoint of O and we are done. Only if A-stable fixpoints do not exist, we would have to deal with the more complex task of directly computing the ultimate stable fixpoints of O.

The following result provides some sufficient conditions under which it is enough to consider a less precise approximation in order to compute ultimate fixpoints of and operator O.

Corollary 5.11 *Let A be any element in the family of approximations of a lattice operator O.*

(1) If $k(A)$ is exact then it is the ultimate Kripke-Kleene, the ultimate well-founded and the unique ultimate stable fixpoint of O.
(2) If $w(A)$ is exact then it is the ultimate well-founded and the unique ultimate stable fixpoint of O.

The next corollary shows that when different approximations are used to study an operator O, the results will not be inconsistent with each other.

Corollary 5.12 *Let A, B be two approximations of a lattice operator O.*

(1) $k(A) \vee k(B)$ is consistent and approximates all fixpoints of O.
(2) $w(A) \vee w(B)$ is consistent.
(3) Each of $w(A)$, $w(B)$ and $w(A) \vee w(B)$ approximates all elements of L^c that are stable fixpoints of both A and B, and all ultimate stable fixpoints of O.

We end this section with a discussion of the notion of ultimate approximation in the context of bilattice approximation operators (rather than consistent approximating operators). Let us consider two approximating operators A and B (defined on L^2). We say that B is more precise than A ($A \leq_p B$, in symbols) if for every *consistent* pair (x, y), $A(x, y) \leq_p B(x, y)$. The restriction

to consistent pairs is essential. Due to symmetry of approximating operators, if $x \leq y$ then

$$A(x,y) \leq_p B(x,y) \text{ if and only if } B(y,x) \leq_p A(y,x).$$

Thus, we cannot require that $A(x,y) \leq_p B(x,y)$ hold for *every* pair (x,y). We also note that the precision ordering $\leq_p$ on $Appx(L^2)$ is reflexive and transitive but not antisymmetric. It is then a pre-order relation and not an order relation, as in the case of the precision ordering on *consistent* approximations. This difference is not essential. The pre-order on the set of approximating operators gives rise to an order relation on the set of equivalence classes of the relation $\equiv$, defined as follows: an approximating operator A is equivalent to an approximating operator B ($A \equiv B$) if and only if $A^c = B^c$. In fact, the resulting partially ordered set is isomorphic to the partially ordered set of consistent approximating operators considered earlier in this section.

Let $O : L \to L$ be an operator on L. We call an operator U on L^2 an *ultimate* approximating operator for O if U is an approximating operator for O and if for every approximating operator B for O such that $U \leq_p B$, $U^c = B^c$. It is easy to see that any approximating operator extending the (consistent) ultimate approximating operator, as defined earlier in the section, is an ultimate approximating operator. The following result is a straightforward consequence of the fact that approximating operators are, by definition, symmetric.

Theorem 5.13 *An operator $U : L^2 \to L^2$ is an ultimate approximating operator for an operator $O : L \to L$ if and only if U is an extension of the (consistent) ultimate approximation for O.*

Theorem 5.13 implies that ultimate fixpoints and ultimate stable fixpoints can be studied in the setting of approximating operators by means of algebraic techniques from [DMT00a].

6 Ultimate semantics for logic programming

In this section, we apply ultimate approximations to propositional logic programming. We derive explicit characterizations of ultimate semantics for logic programs, study their properties and establish the complexity of decision problems concerned with the existence and computing of ultimate models.

In order for the paper to be self-contained, we briefly review basic concepts pertaining to logic programming. A *propositional logic program* P is a finite set of *rules* of the form $p \leftarrow B$, where p is a propositional atom called the *head* of the rule, and B is a finite conjunction of *literals*, that is, atoms and their

negations. We call this conjunction the *body* of the rule. We denote the set of atoms that appear in P by $At(P)$.

We often write logic programs in their *normal* form [Cla78,Fit85]. Let P be a logic program and let p be an atom appearing in P. By $B_P(p)$ we mean a *disjunction* of the bodies of all rules in P with the head p (when p does not appear as the head of a rule, this disjunction is empty, and so contradictory). A collection of rules $p \leftarrow B_P(p)$, where p ranges over all atoms of P, is the *normal form* of P.

A 2-*valued interpretation* (or, simply, an *interpretation*) is a total function from the set of atoms occurring in a program P to the set of truth values $\{\mathbf{t}, \mathbf{f}\}$. A 3-*valued interpretation* is a total function from the atoms to the set $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. These truth values are ordered by the *truth order* $\mathbf{f} \leq_t \mathbf{u} \leq_t \mathbf{t}$ and by the *precision order* $\mathbf{u} \leq_p \mathbf{f}$, $\mathbf{u} \leq_p \mathbf{t}$. Both orders have point-wise extensions to the collection of all 3-valued interpretations.

The complements of the three truth values are defined as $\mathbf{f}^{-1} = \mathbf{t}, \mathbf{t}^{-1} = \mathbf{f}$ and $\mathbf{u}^{-1} = \mathbf{u}$. A 3-valued interpretation $\mathcal{K}$ can be extended, by standard recursion, to all formulas:

$$v_{\mathcal{K}}(\varphi) = \begin{cases} \mathcal{K}(\varphi) & \text{if } \varphi \text{ is an atom} \\ \min_{\leq_t}\{v_{\mathcal{K}}(\psi_1), v_{\mathcal{K}}(\psi_2)\} & \text{if } \varphi = \psi_1 \wedge \psi_2 \\ \max_{\leq_t}\{v_{\mathcal{K}}(\psi_1), v_{\mathcal{K}}(\psi_2)\} & \text{if } \varphi = \psi_1 \vee \psi_2 \\ v_{\mathcal{K}}(\psi)^{-1} & \text{if } \varphi = \neg\psi. \end{cases}$$

We observe that if $\mathcal{K}$ is 2-valued, then $v_{\mathcal{K}}$ is also 2-valued (and coincides with the standard extension of $\mathcal{K}$ to all formulas in the 2-valued case). We define $\mathcal{K} \models \varphi$ if $v_{\mathcal{K}}(\varphi) = \mathbf{t}$. The following proposition is well known.

Proposition 6.1 *If $\mathcal{K} \leq_p \mathcal{K}'$, then for each formula φ, $v_{\mathcal{K}}(\varphi) \leq_p v_{\mathcal{K}'}(\varphi)$.*

The collections of 2-valued and 3-valued interpretations can be considered within the lattice-based framework. Following a common convention, we identify a 2-valued interpretation I with the set of atoms that are true in I. The set of interpretations, with the order defined by the inclusion relation, forms a complete lattice. In particular, $\emptyset$ and the set of all atoms from P are the least and the greatest elements in this lattice. Furthermore, the greatest lower bound and the least upper bound of a set S of interpretations are given by the intersection $\bigcap S$ and the union $\bigcup S$, respectively.

We identify a 3-valued interpretation $\mathcal{K}$ with a pair (I, J) of 2-valued interpretations of P, where

$$I = \{a\colon \mathcal{K}(a) = \mathbf{t}\} \quad \text{and} \quad J = \{a\colon \mathcal{K}(a) = \mathbf{t} \text{ or } \mathbf{u}\}.$$

Clearly, $I \subseteq J$. Moreover, given a pair of 2-valued interpretations (I, J), where $I \subseteq J$ (a consistent pair (I, J)). One can define

$$\mathcal{K}(a) = \begin{cases} \mathbf{t} \text{ if } a \in I \\ \mathbf{f} \text{ if } a \notin J \\ \mathbf{u} \text{ otherwise.} \end{cases}$$

Thus, each 3-valued interpretation $\mathcal{K}$ determines a unique pair (I, J) such that $I \subseteq J$ and conversely. It follows that the set of 3-valued interpretations can be identified with the set L^c, where L is the lattice of 2-valued interpretations (with the inclusion as the lattice order). Moreover, the precision order $\subseteq_p$ on L^c coincides with the precision order $\leq_p$ on 3-valued interpretations (that is, the point-wise extension of the precision order on the set of truth values $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$). Since L is a complete lattice, $\langle L^c, \subseteq_p \rangle$ is a chain-complete poset.

Each logic program P determines a special operator on the lattice of interpretations. We call this operator the *immediate consequence operator* or the van Emden-Kowalski operator and denote it by T_P [vEK76,AvE82]. For an interpretation I, we define $T_P(I)$ as follows:

$$T_P(I) = \{p\colon \quad p \leftarrow B \in P \text{ and } \ I \models B\}.$$

All major 2-valued semantics of logic programming, including the supported-model [Cla78,AvE82] and stable-model semantics [GL88], are defined as classes of fixpoints of the operator T_P.

Some logic programs do not have any 2-valued supported or stable models. To handle such programs, researchers introduced 3-valued semantics [Fit85,Kun87,VRS91]. These semantics can be described in terms of fixpoints of the 3-valued immediate consequence operator $\mathcal{T}_P$ of P , defined on the poset L^c of 3-valued interpretations [Kun87,Fit85]. For an interpretation (I, J), where $I \subseteq J$, we set $\mathcal{T}_P(I, J) = (I', J')$, where

$$I' = \{p\colon \text{for some } \ p \leftarrow B \in P, \quad v_{(I,J)}(B) = \mathbf{t}\}$$

and

$$J' = \{p\colon \text{for some } \ p \leftarrow B \in P, \quad v_{(I,J)}(B) = \mathbf{t} \text{ or } \mathbf{u}\}.$$

It is evident that $I' \subseteq J'$, that is, $\mathcal{T}_P$ is indeed an operator on L^c. This definition extends to programs given in the normal form. If Q is a normal form of a program P, then $T_Q = T_P$ and $\mathcal{T}_Q = \mathcal{T}_P$.

It follows directly from the definitions that $\mathcal{T}_P$ is exact and $\subseteq_p$-monotone. Moreover, for every 2-valued interpretation I, $\mathcal{T}_P(I, I) = (T_P(I), T_P(I))$. Thus, $\mathcal{T}_P$ is an approximating operator for T_P. Moreover, it turns out that its Kripke-

Kleene, supported, well-founded and stable fixpoints determine precisely the corresponding semantics of the program P [DMT00a].

The operator $\mathcal{T}_P$ does not depend on T_P but on P and there is no algebraic derivation of $\mathcal{T}_P$ from T_P. We will now consider the ultimate approximation to the operator T_P, which can be constructed in a purely algebraic way from T_P, following the technique we described in Section 5. We will then introduce the corresponding ultimate semantics and study their properties.

Let P be a logic program. We denote by U_P the ultimate approximating operator for the operator T_P. It is an operator on pairs (I, J) of 2-valued interpretations such that $I \subseteq J$ or equivalently on 3-valued interpretations. By specializing Theorem 5.6 to the operator T_P we obtain that for every two interpretations I and J such that $I \subseteq J$,

$$U_P(I, J) = (\bigcap T_P([I, J]), \bigcup T_P([I, J])).$$

Replacing the ultimate approximating operator U_O in the definitions of ultimate Kripke-Kleene, well-founded and stable fixpoints with U_P results in the corresponding notions of ultimate Kripke-Kleene, well-founded and stable models (semantics) of a program P.

The operator U_P is defined algebraically in terms of the T_P operator. It turns out that this operator and its $\leq_p$-precise fixpoint were introduced earlier in [Fit94] in a more standard way using an alternative truth valuation. For any consistent pair (I, J), the *supervaluation* $v^{sv}_{(I,J)}$ is defined as follows [vFr66]:

$$v^{sv}_{(I,J)}(\varphi) = \begin{cases} \mathbf{t} & \text{if for each } K \in [I, J],\ K \models \varphi \\ \mathbf{f} & \text{if for each } K \in [I, J],\ K \not\models \varphi \\ \mathbf{u} & \text{otherwise.} \end{cases}$$

Clearly, for every atom a, $v_{(I,J)}(a) = v^{sv}_{(I,J)}(a)$. Furthermore, since each $K \in [I, J]$, viewed as a 3-valued interpretation, is more precise than (I, J), it follows from Proposition 6.1 that $v_{(I,J)}(\varphi) \leq_p v_K(\varphi)$. Consequently, we obtain the following result.

Proposition 6.2 *For every consistent pair (I, J) and every formula φ, $v_{(I,J)}(\varphi) \leq_p v^{sv}_{(I,J)}(\varphi)$.*

In [Fit94], Fitting used supervaluations to define the following operator Φ_P: for every consistent pair (I, J), $\Phi_P(I, J) = (I', J')$, where

$$I' = \{a\colon v^{sv}_{(I,J)}(B_P(a)) = \mathbf{t}\} \quad \text{and} \quad J' = \{a\colon v^{sv}_{(I,J)}(B_P(a)) = \mathbf{t} \text{ or } \mathbf{u}\}.$$

This operator is the same as the ultimate approximating operator U_P. This is proven in the following proposition.

Proposition 6.3 *For every consistent pair* (I, J), $U_P(I, J) = \Phi_P(I, J)$.

Proof: For every atom a, $a \in \bigcap T_P([I, J])$ if and only if $a \in T_P(K)$, for every $K \in [I, J]$. This is equivalent to the statement that $K \models B_P(a)$, for every $K \in [I, J]$ which, in turn, is equivalent to $v^{sv}_{(I,J)}(B_P(a)) = \mathbf{t}$. Similarly, one can show that $a \in \bigcup T_P([I, J])$ if and only if $v^{sv}_{(I,J)}(B_P(a)) \neq \mathbf{f}$. □

We are now in a position to discuss commonsense reasoning intuitions underlying abstract algebraic concepts of ultimate approximation and its fixpoints. We view consistent pairs (I, J) of interpretations as approximations of interpretations K such that $I \subseteq K \subseteq J$. Let K be an interpretation known to be approximated by the pair (I, J). The interpretation I represents a lower bound and specifies atoms that are definitely true in K, while J represents an upper bound and specifies the atoms that are possibly true in K. If an atom p is derived by applying the operator T_P to *every* interpretation in $[I, J]$, it can safely be assumed to be true in $T_P(K)$. Consequently, $I' = \bigcap T_P([I, J])$ is a safe lower bound for $T_P(K)$. Similarly, if p can be derived by the operator T_P from *at least* one interpretation in $[I, J]$, it might be true in $T_P(K)$. Thus, the set $J' = \bigcup T_P([I, J])$, consisting of all such atoms, can be regarded as a safe upper bound of $T_P(K)$. It follows that $(I', J') = U_P(I, J)$ is indeed well-motivated as an approximation of $T_P(K)$.

In the case that interests us most, when K is a fixpoint of T_P, it follows that if K is approximated by (I, J), then it is also approximated by $U_P(I, J)$. *Good* approximations of K are those consistent pairs (I, J) that are U_P-reliable (that is, satisfy $(I, J) \leq_p U_P(I, J)$). On a reliable approximation (I, J), the ultimate approximation acts as a revision operator which improves the bounds on fixpoints $K \in [I, J]$ by providing a larger conservative estimate $U(I, J)_1$ and a smaller liberal estimate $U(I, J)_2$. The pair $(\bot, \top)$ is U_P-reliable and certainly approximates all fixpoints of T_P. By iterating U_P starting at $(\bot, \top)$, we obtain the ultimate Kripke-Kleene model of P as the least fixpoint of U_P. This model approximates all fixpoints of U_P and, in particular, all exact fixpoints of U_P, that is, fixpoints of T_P (Proposition 3.1). It follows that the ultimate Kripke-Kleene model of P approximates all supported models of P.

Often, however, the Kripke-Kleene model is too weak because we are not interested in all supported models of P. Let us consider for example the program $P_0 = \{p \leftarrow p, \quad q \leftarrow \neg p\}$. Its operator T_{P_0} coincides with the operator O that we discussed in Section 3. It is easy to see that supported models of P_0, that is, the fixpoints of T_{P_0}, are $\{p\}$ and $\{q\}$. In a supported model, each true atom is *supported*, that is, each true atom occurs in the head of a rule with a true body. We note that both models are minimal models. In the model $\{p\}$, p

is *self-supported* in the sense that there is no constructive argument for the truth of atom p: p is derived using the rule $p \leftarrow p$. On the other hand, in the model $\{q\}$, p is false by absence of support and q is true because p is false. The goal underlying the well-founded and stable semantics is to accept as true only those atoms which have such a constructive argument.

The key concept in approximation theory that we use to formalize this intuition is the stable revision operator of U_P. The stable revision operator revises every U_P-reliable approximation (I, J) by $(J^{U_P\downarrow}, I^{U_P\uparrow})$. First, we will explain that, under the assumption that J is an upper bound, that is, all atoms false in J are definitely false, all atoms true in $J^{U_P\downarrow}$ have a constructive argument and, so, are definitely true. Second, we will explain why, under the assumption that I is a lower bound and all atoms in I are definitely true, atoms that are false in $I^{U_P\uparrow}$ can have no constructive argument and hence, must be false.

Let (I, J) be a U_P-reliable pair. We use J to construct a new lower bound consisting of atoms that have a constructive argument. Our only basic assumption is that J is an upper bound and atoms false in J are definitely false. In other words, J specifies atoms that are possibly true. So, initially we do not preclude any interpretation contained in J. If some atom p can be derived by applying the operator T_P to *every* element of $[\bot, J]$ then, arguably, p should be accepted as definitely true. The set of all these atoms is exactly $\bigcap T_P([\bot, J])$. So, this set can be taken as a safe new lower bound, giving a smaller interval $[I_1, J]$ of possible interpretations. We now repeat the same process and obtain a new lower bound, say I_2, consisting of those atoms that can be derived from every interpretation in $[I_1, J]$. It is given by $I_2 = \bigcap T_P([I_1, J])$. Clearly, I_2 improves on I_1. We iterate this process until a fixpoint $J^{U_P\downarrow}$ is reached. This fixpoint consists of all these atoms for which we have a *constructive* argument that they are true, given that all atoms not in J are false. Thus, under this assumption, $J^{U_P\downarrow}$ provides a safe lower bound for the set of atoms the program should specify as true.

We recall from Proposition 3.4 that each fixpoint $K \subseteq J$ of T_P is larger than $J^{U_P\downarrow}$. Let us note that one cannot guarantee that $J^{U_P\downarrow}$ improves the lower bound I. However, if (I, J) is prudent, then we know that $I \subseteq J^{U_P\downarrow}$. In the context of the example program P_0, one can verify easily that $\{p\}^{\downarrow} = \bot$ (that is, (p, p) is not U_P-prudent). This shows that if we only assume that q is false, there is no constructive argument for the truth of p. On the other hand, we have $\{q\}^{\downarrow} = \{q\}$ (that is, (q, q) is U_P-prudent), so assuming that p is false, we find a constructive argument that q is true.

The reasoning for revising the upper bound is different. The goal now is to discover all the atoms that, assuming that all the atoms in I are true, cannot possess a constructive argument. The idea is to identify atoms that can be reached from I in the derivation process. Any atom that cannot be reached

must be false. Certainly, we have that all atoms in I are possibly true (in fact, they are just true) and since all atoms in $T_P(I)$ can be derived from I, they are possibly true as well. We define $J_1 = T_P(I) = U_P(I, I)_2$. Since (I, J) is U_P-reliable, it follows that $I \subseteq J_1$. To find more possibly true atoms, we apply T_P on all interpretations in $[I, J_1]$. Each atom p that can be derived from at least one interpretation $J \in [I, J_1]$ could be possibly true. The set J_2 of all atoms that can be computed this way is exactly $U_P(I, J_1)_2$ and we have that $J_1 \subseteq J_2$. We can now iterate this process until the fixpoint $I^{U_P\uparrow}$ is reached. Since we have exhausted all possible ways of deriving atoms (starting from I), each atom false in $I^{U_P\uparrow}$ cannot have a constructive argument and hence is definitely false. Thus, $I^{U_P\uparrow}$ yields a safe upper bound. By an argument in Proposition 3.7, the new upper bound can be guaranteed to be included in J.

Let us apply this method in the example P_0. We start without assuming any atoms to be true, that is, we start from $\bot = \emptyset$. Applying T_{P_0} on $\emptyset$ yields $J_1 = \{q\}$. Next we apply T_{P_0} on $[\bot, \{q\}]$ and take the union; this yields $J_2 = \{q\}$. So we reached a fixpoint already and discovered that p cannot be reached from $\bot$ and hence is definitely false.

What do these intuitions imply for the different types of fixpoints of the stable revision operator? The ultimate well-founded model is computed starting from $(\bot, \top)$, without making any assumptions about what atoms are definitely true or definitely false. All atoms true in that model have a constructive argument. All atoms false in it have no constructive argument and may be taken as false. The arguments behind the truth and falsity of atoms in partial and exact stable models are weaker as they are based on additional assumptions. In a partial stable model (I, J), atoms in I have a constructive argument, under the assumption that the atoms in J are the only possible atoms, and atoms not in J cannot possibly have a constructive argument, given that atoms in I are true. An ultimate stable model I has the property that it is precisely the set of atoms with a constructive argument, under the assumption that all its false atoms are definitely false. This discussion demonstrates that abstract algebraic concepts of ultimate approximations can be given a sound intuitive account.

The T_P operator is a lattice operator and determines the family of its approximating operators. One of these operators, the operator $\mathcal{T}_P$, underlies much of the standard theory of normal logic programs. The operator $\mathcal{T}_P$ is not the most precise approximation of T_P. In other words, in general it is different than the ultimate approximation of T_P, the operator U_P. The operator U_P generates the most precise notions of Kripke-Kleene, well-founded and stable models of P. Exploiting the theory of fixpoints of approximating operators presented above, we will now study properties of the ultimate semantics (that is, those determined by U_P) and their relationship to the standard ones (that is, those determined by $\mathcal{T}_P$).

Theorem 6.4 *Let P, P' be two programs such that $T_P = T_{P'}$. Then, the ultimate well-founded models and ultimate stable models of P and P' coincide.*

Proof: Theorem 5.6 implies that $U_P = U_{P'}$. But then all fixpoints of U_P and $U_{P'}$ coincide. Thus, the result follows. □

This assertion does not hold for the (standard) well-founded and stable models. For instance, let $P_1 = \{p \leftarrow p, \quad p \leftarrow \neg p\}$ and $P_2 = \{p \leftarrow\}$. Clearly, $T_{P_1} = T_{P_2}$. However, P_2 has a stable model, $\{p\}$, while P_1 has no stable models. Furthermore, p is true in the well-founded model of P_2 and unknown in the well-founded model of P_1.

As a consequence to Theorem 6.4 we obtain the following corollary. It asserts that applying equivalence preserving transformations to the bodies of rules preserves all the ultimate versions of the semantics.

Corollary 6.5 *Let P, P' be two programs. If for every atom p, the formula $B_P(p) \leftrightarrow B_{P'}(p)$ is a tautology of propositional logic, then the ultimate Kripke-Kleene and well-founded models and the ultimate stable models of P and P' coincide.*

This property is not satisfied by the standard versions of the semantics. Let us consider programs P_1 and P_2 that we discussed above and observe that $B_{P_1}(p) \leftrightarrow B_{P_2}(p)$ is a tautology. At the same time, as we observed earlier, p is unknown in the well-founded model of P_1 and is true in the well-founded model of P_2.

The behavior displayed by P_1 and P_2 is a special case of a more general pattern. Let $r = p \leftarrow B$ be a logic program rule. The *complement splitting* of r with respect to an atom q is the set of two rules: $p \leftarrow B, q$ and $p \leftarrow B, \neg q$. The *complement splitting* of programs is an operation consisting of a sequence of *complement splittings* of rules. If P' is the result of complement splitting applied to a program P then for every atom p, $B_P(p)$ is logically equivalent to $B_{P'}(p)$.

One can show that if P' can be obtained from P by complement splitting, then $\mathcal{T}_{P'}$ is equal to or less precise than $\mathcal{T}_P$. This implies that P' has the same or weaker Kripke-Kleene and well-founded models as P, and every stable model of P' is a stable model of P. Hence, complement splitting yields programs that are "weaker" than the original ones (under the semantics induced by $\mathcal{T}_P$). Coming back to our example, one can see that P_1 is the result of applying complement splitting to P_2.

Although standard and ultimate semantics differ, they are always consistent with each other. The following corollary follows directly from Theorem 5.2.

Corollary 6.6 *Let P be a logic program.*

(1) The Kripke-Kleene model of P is the same as or less precise than the ultimate Kripke-Kleene model of P
(2) The well-founded model of P is the same as or less precise than the ultimate well-founded model of P
(3) Every stable model of P is an ultimate stable model of P.

While in general the standard and ultimate semantics differ, in many cases they coincide. One consequence of Corollary 5.11 is that if the well-founded model of a program is two-valued, then it coincides with the ultimate well-founded model. Thus, we have the following result on the classes of Horn and weakly stratified programs (the class of programs introduced and studied in [Prz90]):

Corollary 6.7 *If a logic program P is a Horn program or a (weakly) stratified program, then its ultimate well-founded semantics coincides with the standard well-founded semantics.*

Another condition implying equality of the standard and ultimate semantics is the monotonicity of the T_P operator. The following result is a consequence of Proposition 5.10.

Corollary 6.8 *Let P be a program such that T_P is a monotone operator. The least Herbrand model of P is the ultimate well-founded model and the unique ultimate stable model of P.*

Again this property is not satisfied by the standard well-founded semantics, as witnessed by the program $P_1 = \{p \leftarrow p,\ \ p \leftarrow \neg p\}$. The atom p is unknown in the standard well-founded semantics of P_1 but true in the least Herbrand model of this program (which exists since T_{P_1} is monotone).

We will now study computational aspects of ultimate semantics for logic programs. First, we recall that ultimate supported models of a program P are precisely complete supported models of P (Proposition 3.1). Consequently, problems concerning the existence of ultimate supported models have the same complexity as their counterparts concerning complete supported models. In particular, it follows from the results of [MT91] that the problem of the existence of an ultimate supported model of a finite propositional program is NP-complete.

The situation is different for other semantics. We will show that basic computational problems associated with the exact ultimate stable models, and the ultimate Kripke-Kleene and well-founded models are more complex. Thus, in general, attractive properties of ultimate semantics come at a price.

We start the discussion of complexity results with several simple observations and lemmas. In the remainder of this section, without loss of generality we assume that programs are given in their normal form.

Let I and J be two interpretations such that $I \subseteq J$ and let φ be a DNF formula. By $[\varphi]_{I,J}$ we denote the formula obtained from φ by substituting every atom x such that $x \notin J$ by $\mathbf{f}$, and every atom x such that $x \in I$ by $\mathbf{t}$.

Let P be a logic program in the normal form. We define the *reduct* of P with respect to interpretations I and J such that $I \subseteq J$, denoted $P_{I,J}$, as follows:

$$P_{I,J} = \{p \leftarrow [B_P(p)]_{I,J} \colon p \in At(P)\}.$$

We note that all atoms appearing in formulas $[B_P(p)]_{I,J}$ are elements of $J \setminus I$ (if $I = J$, the only symbols appearing in $[B_P(p)]_{I,J}$ are $\mathbf{f}$ and $\mathbf{t}$). We have the following lemma.

Lemma 6.9 *Let P be a logic program in the normal form and let I, J be two interpretations such that $I \subseteq J$.*

(1) An atom p of P belongs to $U_P(I,J)_1$ if and only if the formula $[B_P(p)]_{I,J}$ is a tautology.
(2) An atom p of P belongs to $U_P(I,J)_2$ if and only if the formula $[B_P(p)]_{I,J}$ is satisfiable.

Proof: We recall that

$$U_P(I,J)_1 = \bigcap T_P([I,J]) = \bigcap_{I \subseteq K \subseteq J} T_P(K).$$

Thus, an atom p belongs to $U_P(I,J)_1$ if and only if for every interpretation $M \in [\emptyset, J \setminus I]$ (according to our notation, $[\emptyset, J \setminus I]$ is the collection of all subsets of $J \setminus I$), the formula $[B_P(p)]_{I,J}$ is true in M or, equivalently, if and only if the formula $[B_P(p)]_{I,J}$ is a tautology. We also have that

$$U_P(I,J)_2 = \bigcup T_P([I,J]) = \bigcup_{I \subseteq K \subseteq J} T_P(K).$$

It follows that an atom p belongs to $U_P(I,J)_2$ if and only if for some interpretation $M \in [\emptyset, J \setminus I]$, the formula $[B_P(p)]_{I,J}$ is true in M or, equivalently, if and only if the formula $[B_P(p)]_{I,J}$ is satisfiable. Thus, the assertion follows. □

Lemma 6.9 implies that the complexity of algorithms to compute $U_P(I,J)$ depends on the complexity of the problems to decide whether a DNF formula is a tautology and whether it is satisfiable. The first of these problems is co-NP-complete. The second one is in P. Thus, we get the following corollary.

Corollary 6.10 *Let P be a finite propositional logic program and let I, J be*

two interpretations such that $I \subseteq J$. Then, computing $U_P(I,J)_1$ can be accomplished by means of polynomial number of calls to an NP-oracle, with all other tasks taking polynomial time, and computing $U_P(I,J)_2$ can be accomplished in polynomial time.

From Lemma 6.9, we also obtain the following lower bound on the complexity of the problem to determine the truth value of an atom in the ultimate Kripke-Kleene and well-founded semantics, denoted by $k^U(P)$ and $w^U(P)$, respectively.

Corollary 6.11 *The problems: given a logic program P and an atom q, determine whether $q \in k^U(P)_1$ and, similarly, determine whether $q \in w^U(P)_1$, are* co-NP-*hard.*

Proof: Let φ be a theory in the conjunctive normal form. Let ψ be the formula obtained from $\neg\varphi$ by applying the de Morgan Laws. Clearly, ψ is a DNF formula. We define a program P as follows. First, we include in P the rule

$$q \leftarrow \psi$$

where q is a new propositional variable that does not appear in φ. Since ψ is in the disjunctive normal form, the expression $q \leftarrow \psi$ is indeed a program clause. Next, for every propositional variable p in φ we introduce a new propositional variable p' and include in P the pair of rules of the form

$$p \leftarrow \neg p'$$
$$p' \leftarrow \neg p.$$

By Lemma 6.9, it is easy to see that $q \in k(P)_1$ (and to $w(P)_1$) if and only if ψ is a tautology or, equivalently, if and only if φ is not satisfiable. Thus, the assertion follows. □

We will now prove our first complexity result concerning the problem of the existence of exact ultimate stable models.

Theorem 6.12 *The problem "given a finite propositional logic program P, decide whether P has an exact ultimate stable model" is Σ_2^P-complete.*

Proof: By Corollary 5.8, an interpretation J is an ultimate stable model of a program P if and only if $J = lfp(U_P(\cdot, J)_1)$. One can compute $lfp(U_P(\cdot, J)_1)$ by iterating the operator $U_P(\cdot, J)_1$ starting with the empty set. At most n iterations, where n is the number of atoms in P, are needed. Since the problem of computing $U_P(I,J)_1$ can be accomplished in polynomial time using polynomially many references to an NP-oracle (Corollary 6.10), it follows that the problem to decide whether P has an exact ultimate stable model is in the class Σ_2^P.

To prove the "hardness" part, we proceed as follows. We start by recalling that the following problem, denoted QBF_2, is Σ_2^P-complete: given a DNF formula φ with $m+n$ variables $x_1, \ldots, x_m$, $y_1, \ldots, y_n$, decide whether there is a truth assignment $I \subseteq \{x_1, \ldots, x_m\}$ such that φ_I is a tautology, where φ_I is the formula obtained by replacing in φ all occurrences of atoms from I with $\mathbf{t}$, and by replacing all occurrences of atoms from $\{x_1, \ldots, x_m\} \setminus I$ with $\mathbf{f}$. In particular, φ_I is a formula containing variables from the set $\{y_1, \ldots, y_n\}$ only.

We will reduce the problem QBF_2 to our problem. For each x_i, $i = 1, \ldots, m$, in φ, we introduce a new propositional variable x'_i. We also introduce two new atoms p and q. By φ' we denote the formula obtained from φ by replacing literals $\neg x_i$ in the disjuncts of φ with new atoms x'_i. We define a program P to consist of the following clauses:

(1) $x_i \leftarrow \neg x'_i$ and $x'_i \leftarrow \neg x_i$, for every $i = 1, \ldots, m$
(2) $y_i \leftarrow \varphi'$, for every $i = 1, \ldots, n$
(3) $p \leftarrow \varphi'$
(4) $q \leftarrow \neg p, \neg q$.

We will show that there is an interpretation $I \subseteq \{x_1, \ldots, x_m\}$ such that φ_I is a tautology if and only if P has an ultimate exact stable model.

For a subset $I \subseteq \{x_1, \ldots, x_m\}$, let us define $I' = I \cup \{x'_i : x_i \notin I\}$. Let us also define $M_I = I' \cup \{p, y_1, \ldots, y_n\}$. Clearly, $I \subseteq M_I$ and the formula $[\varphi']_{I', M_I}$ (we introduced this notation just before Lemma 6.9) is well defined. We observe that

$$\varphi_I = [\varphi']_{I', M_I}.$$

By Lemma 6.9, and since $B_P(p) = \varphi'$, the formula $[\varphi']_{I', M_I}$ (or φ_I) is a tautology if and only if $p \in U_P(I', M_I)_1$. Therefore, to complete the proof, we need to show that there is an interpretation $I \subseteq \{x_1, \ldots, x_m\}$ such that $p \in U_P(I', M_I)_1$ if and only if P has an ultimate exact stable model.

It is easy to verify that for every $I \subseteq \{x_1, \ldots, x_m\}$ and for every $J \subseteq M_I$, $U_P(J, M_I)_1$ satisfies the following properties:

(1) $U_P(J, M_I)_1 \cap \{x_1, \ldots, x_n, x'_1, \ldots, x'_n\} = I'$
(2) $U_P(J, M_I)_1 \cap \{y_1, \ldots, y_n, p, q\}$ is either $\emptyset$ or $\{y_1, \ldots, y_n, p\}$.

The property (2) follows from the fact that the bodies of rules of $y_1, \ldots, y_n, p$ are identical.

Consequently, $U_P(J, M_I)_1$ is either I' or M_I. It follows that the monotone operator $U_P(\cdot, M_I)_1$ is an operator in $[\emptyset, M_I]$ and hence has a least fixpoint, which is either I' or M_I.

Let us assume that there is an interpretation $I \subseteq \{x_1, \ldots, x_m\}$ such that $p \in$

$U_P(I', M_I)_1$. This means that I' is not a fixpoint of the operator $U_P(\cdot, M_I)_1$, hence M_I is the least fixpoint. Since U_P approximates T_P, M_I is then also a fixpoint of T_P. By Proposition 3.13, M_I is an ultimate exact stable model of P.

Conversely, let us assume that M is an ultimate exact stable model of P. Then M is a fixpoint of the operator T_P and we have the following four properties.

(1) q is false in M (if q is true in M, T_P does not derive q)
(2) p is true in M (otherwise T_P derives q)
(3) $y_1, \ldots, y_n$ are true in M (since the rules defining them have the same bodies as p)
(4) for each x_i, either x_i or x_i' is true in M.

It follows from these properties that there is $I \subseteq \{x_1, \ldots, x_m\}$ such that $M = M_I$. This last identity and our assumption that M is an ultimate exact stable model of P imply that M_I is the least fixpoint of $U_P(\cdot, M_I)_1$. By the properties of the operator $U_P(\cdot, M_I)_1$ that we stated above, $p \in U_P(I', M_I)_1$ (if $p \notin U_P(I', M_I)_1$, then $U_P(I', M_I)_1 = I'$, a contradiction). □

Next, we establish the complexity of computing ultimate Kripke-Kleene and well-founded models.

Theorem 6.13 *Given a finite propositional logic program, one can compute the ultimate well-founded fixpoint of P as well as the ultimate Kripke-Kleene fixpoint of P using polynomially many calls to an NP-oracle with all other tasks taking polynomial time. In other words, the associated decision problems are in the class Δ_2^P.*

Proof: If P is a finite propositional program, then it follows directly from the definition of the ultimate Kripke-Kleene fixpoint of T_P (that is, the ultimate Kripke-Kleene model of P) that it can be computed by means of polynomially many (in the size of P) evaluations of the value $U_P(I, J)$, where $I \subseteq J$ are interpretations, with all other computational tasks taking only polynomial amount of time. Thus, the result follows from Corollary 6.10.

To compute the well-founded model we need a polynomial number of iterations of the stable operator for U_P. Each such computation requires a polynomial number of computations of the form $U_P(I, J)$, where $I \subseteq J$. It follows that to compute the well-founded semantics, the polynomial number of calls to a procedure computing values of U_P suffices. □

These results show that appealing semantic properties of the ultimate versions of semantics of logic programs come at a price. In practice, this price is often low. For wide classes of programs the complexity of computing well-founded or stable semantics does not grow at all. For instance, it is so for any class

of programs for which the (standard) well-founded model is 2-valued, standard and ultimate versions of the stable and well-founded semantics coincide. Consequently, the complexity of computing ultimate semantics is the same as computing the standard semantics. As mentioned above, this holds for the classes of Horn programs and weakly stratified programs.

Likewise, for any class of programs for which the ultimate approximation U_P and the standard operator $\mathcal{T}_P$ are identical, all standard and ultimate versions of programs coincide. Below, we will describe one such a class.

We will also present a broad class of programs where ultimate well-founded and exact ultimate stable models do not necessarily coincide with their standard counterparts; yet, the complexity of computation of ultimate well-founded and exact ultimate stable models does not increase.

The next proposition identifies a class of programs for which the standard and ultimate approximations are identical.

Proposition 6.14 *Let P be a program such that for every pair of atoms p and q, either each occurrence of q in $B_P(p)$ is positive or each occurrence of q in $B_P(p)$ is negative. Then U_P and $\mathcal{T}_P$ are identical.*

Proof: For each atom p, the formula $B_P(p)$ is a DNF formula satisfying the property that no atom q occurs both positively and negatively in it. By Proposition 6.3, to prove the assertion it is enough to show that for each DNF formula φ satisfying this property, $v_{(I,J)}(\varphi) = v^{sv}_{(I,J)}(\varphi)$.

By Proposition 6.2, $v_{(I,J)}(\varphi) \leq_p v^{sv}_{(I,J)}(\varphi)$. Thus, all we need to show is that if $v_{(I,J)}(\varphi) = \mathbf{u}$ then $v^{sv}_{(I,J)}(\varphi) = \mathbf{u}$ or equivalently, that there exist $K, K' \in [I, J]$ such that $K \models \varphi$ and $K' \not\models \varphi$.

We define $K = I \cup \{r : r \in J$ and r occurs positively in $\varphi\}$ and $K' = I \cup \{r : r \in J$ and r occurs negatively in $\varphi\}$. Clearly, both K and K' belong to $[I, J]$.

If $v_{(I,J)}(\varphi) = \mathbf{u}$ then there is a disjunct B of φ such that $v_{(I,J)}(B) = \mathbf{u}$. Let us assume that $B = a_1 \wedge \ldots \wedge a_m \wedge \neg b_1 \wedge \ldots \wedge \neg b_n$. Since $v_{(I,J)}(B) = \mathbf{u}$, all $a_1, \ldots, a_m$ are in J. Moreover, since all $a_1, \ldots, a_m$ occur in B positively, $a_1, \ldots, a_m$ are all in K. Hence, $K \models a_i$, for every i, $1 \leq i \leq m$. Since, $v_{(I,J)}(B) = \mathbf{u}$, for every i, $1 \leq i \leq n$, $b_i \notin I$. Moreover, $b_1, \ldots, b_n$ occur in B negatively. Thus, none of their occurrences in φ is positive and, so, $b_1, \ldots, b_n$ are not in K. Hence, $K \models \neg b_i$, for every i, $1 \leq i \leq n$. It follows that $K \models B$ and, consequently, $K \models \varphi$.

We will now consider the interpretation K'. Since $v_{(I,J)}(\varphi) = \mathbf{u}$, for every disjunct B of φ, $v_{(I,J)}(B) = \mathbf{f}$ or $\mathbf{u}$. Let us observe that $(I, J) \subseteq_p (K', K')$. If $v_{(I,J)}(B) = \mathbf{f}$, then by Proposition 6.1, $v_{(K',K')}(B) = \mathbf{f}$ and, consequently,

$K' \models \neg B$. Let us, therefore, assume that $v_{(I,J)}(B) = \mathbf{u}$. As before let us assume that $B = a_1 \wedge \ldots \wedge a_m \wedge \neg b_1 \wedge \ldots \wedge \neg b_n$. There are two possible cases. The first case is that for some i, $1 \leq i \leq m$, $a_i \notin I$. Since $v_{(I,J)}(B) = \mathbf{u}$, $v_{(I,J)}(a_i) = \mathbf{u}$. Thus, $a_i \in J$. But a_i occurs in B positively and, consequently, has no negative occurrences in φ. Therefore, by the definition of K', $a_i \notin K'$. It follows that $K' \models \neg B$. The other case is that for every i, $1 \leq i \leq m$, $a_i \in I$. Since $v_{(I,J)}(B) = \mathbf{u}$, there is j, $1 \leq j \leq n$, such that $v_{(I,J)}(\neg b_j) = \mathbf{u}$. Thus, $b_j \in J \setminus I$. Moreover, b_j occurs in B negatively. By the definition, $b_j \in K'$. Hence $K' \models b_j$ and, so, $K' \models \neg B$.

Thus, for every disjunct B of φ, $K' \models \neg B$. It follows that $K' \models \neg\varphi$. Since we already proved that $K \models \varphi$, we obtain that $v^{sv}_{(I,J)}(\varphi) = \mathbf{u}$. □

We will now present a broad class of programs for which the complexity bounds of Theorems 6.12 and 6.13 can be improved. Let k be a fixed integer. We define the class $\mathcal{E}_k$ to consist of all logic programs P such that for every atom $p \in At(P)$ at least one of the following conditions holds:

(1) P contains at most k clauses with p as the head;
(2) the body of each clause with the head p consists of at most two elements;
(3) the body of each clause with the head p contains at most one positive literal;
(4) the body of each clause with the head p contains at most one negative literal.

The program P_1 belongs to $\mathcal{E}_2$ and is an example of a program where standard well-founded and stable semantics do not coincide with ultimate versions of these semantics.

Let us recall that the decision whether an atom $p \in At(P)$ belongs to $U_P(I,J)_1$ reduces to the decision whether the formula $B_{P_{I,J}}(p)$ is a tautology. If P is in the class $\mathcal{E}_k$, this question can be resolved in polynomial time. Indeed each formula $B_P(p)$ is a disjunction of a fixed and pre-specified number (k, if $P \in \mathcal{E}_k$) of conjunctions of literals, a 2-DNF theory, the negation of a Horn theory, or the negation of a dual Horn theory [5]. It is well-known that testing whether a formula in any of the three latter forms is a tautology can be accomplished in polynomial time. To see that the same holds for formulas that are disjunctions of k conjunctions of literals (where k is fixed), it is enough to observe that such a formula can be rewritten into an equivalent CNF formula in time n^k (where n is the number of atoms in P). Since testing whether a CNF formula is tautology is a polynomial task, our claim follows.

Theorem 6.15 *The problem "given a finite propositional logic program from class $\mathcal{E}_k$, decide whether P has an exact ultimate stable model" is* NP*-complete.*

[5] A dual Horn theory is a set of clauses with at most one negative literal per clause.

Proof: Since for every program $P \in \mathcal{E}_k$, and for every interpretations I and J such that $I \subseteq J$, $U_P(I, J)$ can be computed in polynomial time, it takes polynomial time to verify whether $J = \mathit{lfp}(U_P(\cdot, J)_1)$. Thus, the problem in question is in the class NP. To prove completeness, we observe that the ultimate stable and the standard stable operator of purely negative programs coincide. Consequently,

(1) there is no difference between exact stable models and exact ultimate stable models
(2) purely negative programs are in $\mathcal{E}_k$
(3) the problem of existence of exact stable models for purely negative programs is NP-complete [MT91].

Thus, the assertion follows. □

Theorem 6.16 *The problem "given a finite propositional logic program P from class $\mathcal{E}_k$, compute the ultimate well-founded fixpoint of P" is in* P.

Proof: For every program $P \in \mathcal{E}_k$, and for every interpretations I and J such that $I \subseteq J$, $U_P(I, J)$ can be computed in polynomial time. Thus, the assertion follows by Lemma 6.9. □

7 Conclusions and discussion

In this paper, we extended our algebraic framework [DMT00a,DMT00b] for studying semantics of nonmonotonic reasoning systems. We started by developing a theory of *consistent* approximating operators that are defined on the set of *consistent* elements of the product bilattice. The advantage of the present approach is that it refers exclusively to objects that are well motivated by commonsense intuitions underlying the concepts of approximation, precision of an approximation and revision of an approximation. In the same time, this new approach turns out to be essentially as powerful as the earlier one.

The main contribution of this paper is the notion of an *ultimate approximation*. In earlier approaches, to study fixpoints of an operator O one had to *select* an appropriate approximating operator. There has been, however, no principled algebraic way to do so. In the present paper, we found a *distinguished* element in the space of all *consistent* approximations and showed that this particular approximation, the ultimate approximation, is an effective tool in studying fixpoints of O. In fact, we argued that the Kripke-Kleene, well-founded and stable fixpoints of the ultimate approximation of an operator O can be regarded as the Kripke-Kleene, well-founded and stable fixpoints of the operator O *itself*.

Our framework can be applied to any formalism whose semantics are defined as fixpoints of some lattice operator O. In this paper, we applied our approach to logic programming and obtained a family of new semantics generated by the immediate consequence operator: the ultimate Kripke-Kleene, the ultimate well-founded and the ultimate stable-model semantics. These semantics are well motivated and have attractive properties. First, they are preserved when we modify the program, as long as the 2-valued immediate consequence operator stays the same (a property that does not hold in general for standard semantics). Second, the ultimate Kripke-Kleene and the ultimate well-founded semantics are stronger, in general, than their standard counterparts, yet approximate the collection of all fixpoints of O and the collection of all stable fixpoints of O, respectively. The disadvantage is that the complexity of ultimate semantics is, in general, higher. Fortunately, as we demonstrated in Section 6, for large classes of programs that are likely to arise in practical applications, the complexity of computing ultimate semantics remains the same as that of their standard counterparts.

The approach developed in this paper can be applied in other settings as well. In [DPBn01], it was used to define well-founded and stable semantics for an extension of logic programs with aggregates. An aggregate in this language is an arbitrary second order relation taking set expressions and lambda expressions as arguments. The framework can also be applied to default and autoepistemic logics where it results in new (ultimate) semantics with appealing semantic features.

We end this discussion with a comment on the broader role played by approximation theory. Tarski's fixpoint theory can be considered as a general method for modeling monotone constructions and positive inductive definitions. We contend that approximation theory provides a generalized algebraic account of *non-monotone* constructions and *non-monotone* forms of induction in mathematics such as iterated induction and induction in well-ordered sets. Some arguments in support of our claim can be found in [Den98,DBM01]. Those papers present comparative studies of different forms of non-monotone induction and argue that logic programming with (some form of) the well-founded semantics formalizes this broad class of constructive techniques to specify (define) concepts.

Acknowledgments

This paper is based upon work supported by the National Science Foundation under Grants No. 9874764, 0097278 and 0325063. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors gratefully acknowledge helpful comments by anonymous

referees and by Nikolai Pelov.

References

[Acz77] Aczel, P. An Introduction to Inductive Definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 739–782, North-Holland Publishing Company, 1977.

[AvE82] K.R. Apt and M.H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM*, 29(3):841–862, 1982.

[BS91] C. Baral and V.S. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning In A. Nerode, W. Marek, and V.S. Subrahmanian, editors, *Logic programming and non-monotonic reasoning*, pages 69–86, MIT Press 1991.

[Cla78] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and data bases*, pages 293–322. Plenum Press, 1978.

[Den98] M. Denecker. The well-founded semantics is the principle of inductive definition. In J. Dix, L. Fariñas del Cerro, and U. Furbach, editors, *Logics in Artificial Intelligence*, Lecture Notes in Computer Science, volume 1489, pages 1–16, Springer-Verlag, 1998.

[DMT99] M. Denecker, V. Marek, and M. Truszczyński. Fixpoint 3-valued semantics for autoepistemic logic. In H.J. Levesque, F. Pirri editors, *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*, pages 113 – 136, Springer-Verlag, 1999.

[DMT00a] M. Denecker, V. Marek, and M. Truszczyński. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 127–144. Kluwer Academic Publishers, 2000.

[DMT00b] M. Denecker, V. Marek, and M. Truszczyński. Unified semantic treatment of default and autoepistemic logics. In A.G. Cohn, F. Giunchiglia, B. Selman, editors, *Principles of Knowledge Representation and Reasoning, Proceedings of the Seventh International Conference (KR2000)*, pages 74–84. Morgan Kaufmann Publishers, 2000.

[DBM01] M. Denecker, M. Bruynooghe, and V. Marek. Logic programming revisited: logic programs as inductive definitions. *ACM Transactions on Computational Logic*, 2(4):623–654, 2001.

[DMT02] M. Denecker, V. Marek, and M. Truszczyński. Ultimate approximations in nonmonotonic knowledge representation systems. In D. Fensel, F. Giunchiglia, D. McGuiness, M-A. Williams, editors, In *Principles of Knowledge Representation and Reasoning, Proceedings of the Eighth*

International Conference (KR2002), pages 177–188. Morgan Kaufmann Publishers, 2002.

[DMT03] M. Denecker, V. Marek, and M. Truszczyński. Uniform semantic treatment of default and autoepistemic logics. *Artificial Intelligence Journal*, 143:79–122, 2003.

[DPBn01] M. Denecker, N. Pelov, and M. Bruynooghe. Well-founded and stable semantics for logic programs with aggregates. In Ph. Codognet, editor, *Logic programming, Proceedings of the 2001 International Conference on Logic Programming*, Lecture Notes in Computer Science volume 2237, pages 212–226, Springer-Verlag, 2001.

[Fit85] M. C. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[Fit91] M. C. Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming* , 11:91–116, 1991.

[Fit94] M. C. Fitting. Tableaux for logic programming. *Journal of Automated Reasoning*, 13:175–188, 1994.

[Fit02] M. C. Fitting. Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science*, 278:25–51, 2002.

[Gin88] M.L. Ginsberg. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[GL88] M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In R. Kowalski and K. Bowen, editors, *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[Kon88] K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence Journal*, 35(3):343–382, 1988.

[Kun87] K. Kunen. Negation in logic programming. *Journal of Logic Programming*, 4(4):289–308, 1987.

[Lif90] V. Lifschitz. On open defaults. In J. Lloyd, editor, *Proceedings of the Symposium on Computational Logic*, pages 80–95. Springer-Verlag, 1990.

[Mar76] G. Markowsky. Chain-complete posets and directed sets with applications. *Algebra Universalis*, 6(1):53–68, 1976.

[Moo84] R.C. Moore. Possible-world semantics for autoepistemic logic. In *Proceedings of the Workshop on Non-Monotonic Reasoning*, pages 344–354, 1984. Reprinted in: M. Ginsberg, editor, *Readings on Nonmonotonic Reasoning*, pages 137–142, Morgan Kaufmann Publishers, 1990.

[MT91] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3):588–619, 1991.

[Prz90] T.C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, 13(4):445–464, 1990.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence Journal*, 13(1-2):81–132, 1980.

[Tar55] A. Tarski. Lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[Van93] A. Van Gelder. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, 47(1):185–221, 1993.

[vEK76] M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[vFr66] B. van Fraassen. Singular terms, truth-value gaps, and free logic. *The Journal of Philosophy*, 63(17):481-495, 1966.

[VRS91] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.