ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

B Novikov, University of St Petersburg, Russia and J W Schmidt, University of Hamburg, Germany. (Eds)

Advances in Databases and Information Systems, Moscow 1996

Proceedings of the International Workshop on Advances in Databases and Information Systems (ADBIS '96). Moscow, 10-13 September 1996

Disjunctive Rules and Null Values: Logic Programs with Incomplete Information

Jürgen Kalinski

Published in collaboration with the
British Computer Society

©Copyright in this paper belongs to the author(s)

Disjunctive Rules and Null Values: Logic Programs with Incomplete Information

Jürgen Kalinski

Institute of Computer Science III, University of Bonn
Römerstr. 164, 53117 Bonn, Germany

Abstract

The introduction of negation in rule bodies of logic programs may lead to semantic ambiguities: Normal programs can have more than one stable model and the three-valued well-founded model may leave the truth status of some ground atoms undetermined. We show that this kind of ambiguity can be naturally combined with ambiguities caused by incomplete information. We define generalized notions of stable and well-founded models which allow to combine recursion, negation and incomplete information within a uniform framework. The well-founded approximation is composed of two not necessarily identical anti-monotonic operators. A general characterization of the best well-founded approximation is given.

Programs with disjunctions, maybe tuples and null values are studied as examples of logic programs with incomplete information. What concerns null values, our approach can also be seen as a generalization of Biskup's proposal for relational databases.

1 Introduction

Definite logic programs without function symbols (Datalog) have proved to provide a powerful and adequate representation formalism for many applications. The equivalence of the smallest model of a definite program with the least fixpoint of the corresponding immediate consequence operator has been established two decades ago [21]. Semantic issues become more complicated when negations are allowed in rule bodies. One of the most elegant approaches for normal logic programs is stable model semantics as proposed by Gelfond and Lifschitz [5]. It generalizes the perfect model semantics [16] and is closely related with Autoepistemic Logic [15] as a major formalization of nonmonotonic reasoning.

Unfortunately, stable models do not necessarily exist, and even when they exist they are hard to compute (cf. [13]). In contrast, every normal program has a three-valued well-founded model which in the propositional case can be computed in quadratic time (see [22]). Van Gelder's alternating fixpoint definition [23] as well as Baral and Subrahmanian's studies (see [1] and [2]) have shown that the well-founded model can be understood as an efficient approximation of stable models: It can be denoted by a pair of interpretations $\langle I_1, I_2 \rangle$, such that I_1 is a (sound) subset and I_2 a (complete) superset of every stable model. An implementation is described in [20].

The well-founded model can be expressed as

$$\langle I_1, I_2 \rangle = \langle \text{lf}_P(S_P^2), \text{gfp}(S_P^2) \rangle$$

where S_P is an anti-monotonic operator. Furthermore, S_P oscillates between I_1 and I_2 such that $S_P(I_1) = I_2$ and $S_P(I_2) = I_1$. Whereas stable model semantics can quite directly be extended to disjunctive programs [18], the generalization is not obvious for the well-founded semantics and numerous different semantics have been proposed (see [11]).

The major issues of this paper are the following:

- Stable models are redefined in a generalized setting. They are based on the concept of assumption functions which assign a set of intended models to every program, after negative literals have been removed by a hypothetical pre-interpretation.

- In the same general setting well-founded approximations are defined by a pair of anti-monotonic operators. One could say that stable models are the semantic and well-founded approximations the operational side of our approach.
- Assumption functions admit of different pre-interpretations of negative literals in logic programs. Stable model semantic says which of these pre-interpretations are “reasonable”. Analogously, incomplete data can be completed on a hypothetical basis, such that our uniform framework can be applied again.
- Stable models and well-founded approximations can thus be defined for logic programs with maybe tuples and null values. We believe that in order to make non-monotonic reasoning fit for applications it is imperative to further enhance expressiveness of logic programs and to combine the key notions of non-monotonic reasoning with representation features that have been proposed for or are in use with databases.

Section 2 introduces the basic concepts and notation. Section 3 first defines assumption functions and a generalized notion of stable models. Then alternating operator pairs are introduced and some results are established which highlight the relationship between operator pairs and stable models. The remaining sections present instances of this general scheme. In Section 5 normal and disjunctive logic programs are discussed. In Section 7 relations with maybe tuples are considered. On this basis it will in Section 8 be possible to handle relations with null values.

2 Preliminaries

A *rule* is of the form $A_1 \vee \dots \vee A_m \leftarrow L_1, \dots, L_n$ where every A_i is an atom and every L_i an atom or an atom negated by ‘ $\neg$ ’. Function symbols are not allowed. The intuitive meaning of a rule is that the truth of L_1 and $\dots$ and L_n imply the truth of A_1 or $\dots$ or A_m . Both disjunctions and conjunctions are regarded as sets of atoms and literals, respectively.

Rules can be *disjunctive* ($m \geq 1$) or *normal* ($m = 1$). A rule without any occurrence of ‘ $\neg$ ’ is called *positive*. A (logic) *program* P is a set of rules. If all the rules of P possess one of the above mentioned properties, this property is also ascribed to P itself. Throughout this paper it is assumed that the Herbrand universe $\mathcal{U}$ is finite, the same holds for the Herbrand base HB. Programs are tacitly identified with all ground instantiations of rules in $\mathcal{U}$.

Stable model semantics is one of the most elegant approaches for specifying the meaning of logic programs.

Definition 2.1 (Gelfond/Lifschitz [5]) *Let P be a disjunctive program and I a Herbrand interpretation. The Gelfond-Lifschitz transformation P/I of P modulo I is defined by the following procedure:*

- Remove every rule whose body contains some $\neg A$ with $A \in I$.
- Delete every expression $\neg A$ in the remaining rules.

Intuitively spoken, the Gelfond-Lifschitz transformation realizes a pre-interpretation of negative literals with respect to I .

Definition 2.2 (Gelfond/Lifschitz [6], Przymusiński [18]) *Let P be a disjunctive program. A Herbrand interpretation I is called a stable model of P , if it is a minimal model of P/I .*

Example 2.1 *Consider the following programs:*

$$\begin{array}{ll} P_1 : & P_2 : \\ p \vee q & q \leftarrow \neg p \\ p \leftarrow \neg q & p \leftarrow \neg q \end{array}$$

The graphs in Figure 1 contain an arc $I \rightarrow J$, iff J is a minimal model of P/I . (This illustration is after [1].) When for example the empty set is taken as a hypothetical pre-interpretation of negative literals in P_1 , the arc $\emptyset \rightarrow \{p\}$ indicates that $\{p\}$ is a minimal model of $P_1/\emptyset$.

A stable model corresponds to a loop in the graph representation. So, P_1 as well as P_2 both have stable models $\{p\}$ and $\{q\}$.

 Figure 1: Minimal models of P_1/I (left) and P_2/I (right)

The *stable model semantics* ascribes a logic program its set of stable models. Unfortunately, stable models are hard to compute. For normal programs the three-valued *well-founded semantics* is computationally more attractive [22]. The reader is assumed to be familiar with van Emden and Kowalski's *immediate consequence operator* T_P (see [21]). Then for normal program P

$$S_P(I) := \text{lfp}(T_{P/I})$$

determines which atoms can be inferred from P/I .

Proposition 2.1 (Van Gelder [23], Baral/Subrahmanian [1]) *Let P be a normal program.*

1. *The operator S_P is anti-monotonic.*
2. *The operator S_P^2 is monotonic.*
3. *$\text{lfp}(S_P^2) \subseteq I \subseteq \text{gfp}(S_P^2)$ for every stable model I of P .*
4. *$\text{lfp}(S_P^2) = S_P(\text{gfp}(S_P^2))$ and $\text{gfp}(S_P^2) = S_P(\text{lfp}(S_P^2))$*

The pair $\langle \text{lfp}(S_P^2), \text{gfp}(S_P^2) \rangle$ corresponds to the well-founded model of P , where every atom of $\text{lfp}(S_P^2)$ is true and every atom not in $\text{gfp}(S_P^2)$ is false.

Example 2.2 *The reader may verify that for P_2 we have $\text{lfp}(S_{P_2}^2) = \emptyset$ and $\text{gfp}(S_{P_2}^2) = \{p, q\}$.*

For positive disjunctive programs an immediate consequence operator on disjunctions of atoms has been proposed by Minker and Rajasekar (cf. [14]):

$$T_P^\vee(I) = \{ \delta \mid \delta_0 \leftarrow A_1, \dots, A_n \in P \text{ and } A_i \vee \delta_i \in I \text{ for } 1 \leq i \leq n \text{ and } \delta = \delta_0 \vee \delta_1 \vee \dots \vee \delta_n \}$$

Proposition 2.2 (Minker/Rajasekar [14]) *Let P be a positive disjunctive program and δ a disjunction of atoms. Then the following are equivalent:*

1. *δ is a logical consequence of P .*
2. *$\delta' \in \text{lfp}(T_P^\vee)$ for some disjunction $\delta' \subseteq \delta$.*

In a straightforward adoption of the approach which turned out to be so successful for normal programs we might define

$$S_P^\vee(I) := \text{lfp}(T_{P/I}^\vee) \cap \text{HB}$$

But, interestingly, in spite of its anti-monotonicity, this operator fails to “catch” stable models.

Example 2.3 *For P_1 we get $\text{lfp}(S_{P_1}^{\vee 2}) = \text{gfp}(S_{P_1}^{\vee 2}) = \{p\}$, such that the greatest fixpoint does not subsume the stable model $\{q\}$.*

3 Stable Models and Their Operator Pairs

3.1 Notions of Stability

Gelfond-Lifschitz transformation uses a single Herbrand interpretation as a hypothesis for the evaluation of negative literals. A stable model I stands out by the fact that it is a hypothesis which ‘justifies itself’: I is a minimal model of P/I . But it is not strictly necessary to confine the basic concepts to single Herbrand interpretations. In fact, Baral and Subrahmanian introduced the concept of stable classes for *normal* programs (cf. [1] and [2]). They start from a whole set $\mathcal{I}$ of hypothetical assumptions yielding a set of program transformations and, correspondingly, a smallest model for each of them. Once again, $\mathcal{I}$ is called ‘stable’, if it justifies itself, i.e. if it is a set which is identical with the collection of smallest models of P/I for every $I \in \mathcal{I}$:

$$\mathcal{I} = \{ J \mid J \subseteq \text{HB and there is } I \in \mathcal{I} \text{ such that } J \text{ is the smallest model of } P/I \}$$

The general scenario we are aiming at now consists of the following constituents:

- We are given a complete lattice $(\mathcal{M}, \sqsubseteq)$ of interpretations, where $\sqsubseteq$ is to be thought as an information order. Every positive program is assigned a (possibly empty) set $\mathcal{M}(P)$ of *intended models*. Usually, we will prefer models that are $\sqsubseteq$ -minimal.
- Programs with negation require a hypothetical pre-interpretation of negative literals. The *intended models* of P under hypothesis I will be denoted by $\mathcal{A}_P(I)$. In most cases $\mathcal{A}_P$ will be based on a Gelfond-Lifschitz-like transformation P/I such that $\mathcal{A}_P(I) = \mathcal{M}(P/I)$.

Definition 3.1 Let $(\mathcal{M}, \sqsubseteq)$ be a complete lattice of interpretations and P a program. A mapping $\mathcal{A}_P : \mathcal{M} \rightarrow \wp(\mathcal{M})$ is called an *assumption function* for P .

Assumption functions can be generalized to sets of interpretations by

$$\mathcal{A}_P(\mathcal{I}) := \bigcup_{I \in \mathcal{I}} \mathcal{A}_P(I)$$

The superset relation in $\wp(\mathcal{M})$ can be understood as an information order in the following sense: Every set $\mathcal{I}$ of interpretations represents a certain amount of true statements — namely those which are true in every element of $\mathcal{I}$ — and a number of false statements — namely those which are false in every element of $\mathcal{I}$. In this respect $\mathcal{M}$ is the smallest (i.e. least informative) set of interpretations. We note that $\mathcal{A}_P$ is monotonic in the complete lattice $(\wp(\mathcal{M}), \supseteq)$. Different notions of stability can now be defined within these general terms.

Definition 3.2 Let $(\mathcal{M}, \sqsubseteq)$ be a complete lattice of interpretations, P a program and $\mathcal{A}_P$ an assumption function.

1. An interpretation $I \in \mathcal{M}$ is called a *stable model* of P , iff $I \in \mathcal{A}_P(I)$.
2. A pair of interpretations $\langle I_1, I_2 \rangle$ is called a *stable pair* of P , iff $I_1 \sqsubseteq I_2$, $I_1 \in \mathcal{A}_P(I_2)$ and $I_2 \in \mathcal{A}_P(I_1)$.
3. A set $\mathcal{I} \subseteq \mathcal{M}$ is called a *stable class* of P , iff $\mathcal{I} \subseteq \mathcal{A}_P(\mathcal{I})$.

Obviously, if I is a stable model, then $\langle I, I \rangle$ is a stable pair; and if $\langle I_1, I_2 \rangle$ is a stable pair, then $\{I_1, I_2\}$ is a stable class. Furthermore, the union of stable classes forms a stable class again.

Example 3.1 As has been mentioned in Example 2.1, stable models correspond to loops in the graph representation.

A stable pair is depicted by two nodes $I_1 \sqsubseteq I_2$ such that the graph contains both arcs $I_1 \rightarrow I_2$ and $I_2 \rightarrow I_1$. The stable models of P_1 constitute its only stable pairs $\langle \{p\}, \{p\} \rangle$ and $\langle \{q\}, \{q\} \rangle$, whereas P_2 has a stable pair $\langle \emptyset, \{p, q\} \rangle$.

A stable class is a set $\mathcal{I}$ of nodes such that every node in $\mathcal{I}$ has an incoming arc from some other member of $\mathcal{I}$. So, $\{\{p\}, \{q\}\}$ is a stable class of both P_1 and P_2 .

Proposition 3.1 *Every assumption function has a least informative stable class $\text{lfp}(\mathcal{A}_P)$.¹ Furthermore: If $\mathcal{I}$ is a stable class of P , then $\mathcal{I} \subseteq \text{lfp}(\mathcal{A}_P)$.*

Proof As $\mathcal{A}_P$ is monotonic in the lattice $(\wp(\mathcal{M}), \supseteq)$, the least fixpoint of $\mathcal{A}_P$ always exists and:

$$\text{lfp}(\mathcal{A}_P) = \text{glb}(\{\mathcal{I} \mid \mathcal{I} \subseteq \mathcal{M} \text{ and } \mathcal{A}_P(\mathcal{I}) \supseteq \mathcal{I}\})$$

If $\mathcal{I}$ is a stable class, then $\mathcal{I} \subseteq \mathcal{A}_P(\mathcal{I})$ by definition. Hence: $\text{lfp}(\mathcal{A}_P) \supseteq \mathcal{I}$. ◇

The proposition states that $\text{lfp}(\mathcal{A}_P)$ is sound with respect to stable model semantics: Whatever is true in (every member of) the least fixpoint, will be true in every stable model, and whatever is false in $\text{lfp}(\mathcal{A}_P)$, must be false in every stable model.

3.2 Alternating Operator Pairs

3.2.1 Introducing Alternating Operator Pairs

We will now define a generalized notion of well-founded semantics.

Definition 3.3 *Let $\text{Op}_{P,1} : \mathcal{M} \rightarrow \mathcal{M}$ and $\text{Op}_{P,2} : \mathcal{M} \rightarrow \mathcal{M}$ be anti-monotonic functions such that*

$$\text{Op}_{P,1}(I) \subseteq J \subseteq \text{Op}_{P,2}(I)$$

for every $J \in \mathcal{A}_P(I)$. Then the pair $\langle \text{Op}_{P,1}, \text{Op}_{P,2} \rangle$ is called an alternating operator pair.

Proposition 3.2 *Let $\mathcal{A}_P$ be an assumption function and $\langle \text{Op}_{P,1}, \text{Op}_{P,2} \rangle$ an alternating operator pair. Then:*

$$\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2}) \subseteq \text{glb}(\text{lfp}(\mathcal{A}_P)) \subseteq \text{lub}(\text{lfp}(\mathcal{A}_P)) \subseteq \text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1})$$

Furthermore:

$$\begin{aligned} \text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2}) &= \text{Op}_{P,1}(\text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1})) \\ \text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1}) &= \text{Op}_{P,2}(\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2})) \end{aligned}$$

Proof Monotonicity of $(\text{Op}_{P,1} \circ \text{Op}_{P,2})$ and $(\text{Op}_{P,2} \circ \text{Op}_{P,1})$ is obvious.

Suppose that $(\text{Op}_{P,1} \circ \text{Op}_{P,2})^n(\emptyset) \subseteq \text{glb}(\mathcal{A}_P^{2n}(\mathcal{M}))$ has already been shown, i.e. $(\text{Op}_{P,1} \circ \text{Op}_{P,2})^n(\emptyset) \subseteq I$ for all $I \in \mathcal{A}_P^{2n}(\mathcal{M})$. As $\text{Op}_{P,2}$ is anti-monotonic we obtain:

$$\text{Op}_{P,2}((\text{Op}_{P,1} \circ \text{Op}_{P,2})^n(\emptyset)) \supseteq \text{Op}_{P,2}(I) \quad \forall I \in \mathcal{A}_P^{2n}(\mathcal{M})$$

By definition $\text{Op}_{P,2}(I) \supseteq J$ for every $J \in \mathcal{A}_P(I)$. So:

$$\begin{aligned} \text{Op}_{P,2}((\text{Op}_{P,1} \circ \text{Op}_{P,2})^n(\emptyset)) &\supseteq J \quad \forall I \in \mathcal{A}_P^{2n}(\mathcal{M}) \quad \forall J \in \mathcal{A}_P(I) \\ \text{Op}_{P,2}((\text{Op}_{P,1} \circ \text{Op}_{P,2})^n(\emptyset)) &\supseteq J \quad \forall J \in \mathcal{A}_P^{2n+1}(\mathcal{M}) \end{aligned}$$

Hence:

$$\text{Op}_{P,2}((\text{Op}_{P,1} \circ \text{Op}_{P,2})^n(\emptyset)) \supseteq \text{lub}(\mathcal{A}_P^{2n+1}(\mathcal{M}))$$

One more application of this argument yields

$$(\text{Op}_{P,1} \circ \text{Op}_{P,2})^{n+1}(\emptyset) \subseteq \text{glb}(\mathcal{A}_P^{2n+2}(\mathcal{M}))$$

The proof for the second claim is analogous.

¹ Remember that we consider the information order, where $\mathcal{M}$ is the smallest element.

What concerns the remaining claims first note:

$$\begin{aligned}
 & (\text{Op}_{P,2} \circ \text{Op}_{P,1})(\text{Op}_{P,2}(\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2}))) \\
 = & \text{Op}_{P,2}((\text{Op}_{P,1} \circ \text{Op}_{P,2})(\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2}))) \\
 = & \text{Op}_{P,2}(\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2}))
 \end{aligned}$$

and

$$\begin{aligned}
 & (\text{Op}_{P,1} \circ \text{Op}_{P,2})(\text{Op}_{P,1}(\text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1}))) \\
 = & \text{Op}_{P,1}((\text{Op}_{P,2} \circ \text{Op}_{P,1})(\text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1}))) \\
 = & \text{Op}_{P,1}(\text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1}))
 \end{aligned}$$

So, $\text{Op}_{P,2}(\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2}))$ is a fixpoint of $(\text{Op}_{P,2} \circ \text{Op}_{P,1})$ and

$$\text{Op}_{P,2}(\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2})) \sqsubseteq \text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1})$$

Analogously: $\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2}) \sqsubseteq \text{Op}_{P,1}(\text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1}))$. One more application of $\text{Op}_{P,1}$ and $\text{Op}_{P,2}$ respectively yields:

$$\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2}) = (\text{Op}_{P,1} \circ \text{Op}_{P,2})(\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2})) \sqsubseteq \text{Op}_{P,1}(\text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1}))$$

and

$$\text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1}) = (\text{Op}_{P,2} \circ \text{Op}_{P,1})(\text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1})) \sqsubseteq \text{Op}_{P,2}(\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2}))$$

which completes the proof. $\diamond$

As a consequence of Propositions 3.1 and 3.2 every stable model is “caught” in between $\text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2})$ and $\text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1})$.

3.2.2 Comparing Alternating Operator Pairs

Often an assumption function satisfies the additional constraint that $\mathcal{A}_P$ yields anti-chains, i.e. that for every $I \in \mathcal{M}$:

$$(\mathbf{AC}) \quad \forall J_1 \in \mathcal{A}_P(I) \quad \forall J_2 \in \mathcal{A}_P(I) : \quad J_1 \sqsubseteq J_2 \Rightarrow J_1 = J_2$$

Furthermore, it might be the case that for every $I_1, I_2 \in \mathcal{M}$ with $I_1 \sqsubseteq I_2$:

$$(\mathbf{AMS}) \quad \forall J_1 \in \mathcal{A}_P(I_1) \quad \exists J_2 \in \mathcal{A}_P(I_2) : \quad J_2 \sqsubseteq J_1$$

$$(\mathbf{AMH}) \quad \forall J_2 \in \mathcal{A}_P(I_2) \quad \exists J_1 \in \mathcal{A}_P(I_1) : \quad J_2 \sqsubseteq J_1$$

Property **(AMS)** requires anti-monotonicity in the Smyth-ordering: The intuition is that for $I_1 \sqsubseteq I_2$ every model of P/I_1 should also be a model of P/I_2 . Hence, an intended model of P/I_1 is also an intended model of P/I_2 or the latter has an even smaller one. And **(AMH)** is anti-monotonicity in the Hoare-ordering: Here every intended model of P/I_2 is an improvement of one of P/I_1 .

Proposition 3.3 *If $\mathcal{A}_P$ is an assumption function that satisfies (AC) and one of (AMS) or (AMH), then the stable models of P form an anti-chain.*

Proof Assume that $I_1 \sqsubseteq I_2$ are stable models.

By definition of stable models $I_1 \in \mathcal{A}_P(I_1)$. By **(AMS)** there is some $J_2 \in \mathcal{A}_P(I_2)$ with $J_2 \sqsubseteq I_1$. Hence: $J_2 \sqsubseteq I_2$. But I_2 is also a stable model, i.e. $I_2 \in \mathcal{A}_P(I_2)$. By **(AC)** we obtain $J_2 = I_2$. And this now implies $I_1 = I_2$.

If **(AMH)** is assumed, then $I_2 \in \mathcal{A}_P(I_2)$ implies $\exists J_1 \in \mathcal{A}_P(I_1) : I_2 \sqsubseteq J_1$. Now **(AC)** leads to $I_1 = I_2$ again. $\diamond$

We are going to compare different approximations. We say that $\langle \text{Op}_{P,1}, \text{Op}_{P,2} \rangle$ is *better* than $\langle \text{Op}'_{P,1}, \text{Op}'_{P,2} \rangle$, if $\text{Op}'_{P,1}(I) \sqsubseteq \text{Op}_{P,1}(I)$ and $\text{Op}_{P,2}(I) \sqsubseteq \text{Op}'_{P,2}(I)$ for every interpretation I .

Lemma 3.4 *Let $(\mathcal{M}, \sqsubseteq)$ be a complete lattice and $f : \mathcal{M} \rightarrow \mathcal{M}$ and $g : \mathcal{M} \rightarrow \mathcal{M}$ monotonic functions such that $f(x) \sqsubseteq g(x)$ for every $x \in \mathcal{M}$. Then $\text{lfp}(f) \sqsubseteq \text{lfp}(g)$ and $\text{gfp}(f) \sqsubseteq \text{gfp}(g)$.*

Proof We have:

$$\begin{aligned} \text{lfp}(g) &= \text{glb}(\{x \mid g(x) \sqsubseteq x\}) \\ \text{gfp}(f) &= \text{lub}(\{x \mid x \sqsubseteq f(x)\}) \end{aligned}$$

As $f(x) \sqsubseteq g(x)$ for every $x \in \mathcal{M}$:

$$\{x \mid g(x) \sqsubseteq x\} \subseteq \{x \mid f(x) \sqsubseteq x\} \quad \text{and} \quad \{x \mid x \sqsubseteq f(x)\} \subseteq \{x \mid x \sqsubseteq g(x)\}$$

Hence:

$$\text{glb}\{x \mid f(x) \sqsubseteq x\} \subseteq \text{glb}\{x \mid g(x) \sqsubseteq x\} \quad \text{and} \quad \text{lub}\{x \mid x \sqsubseteq f(x)\} \subseteq \text{lub}\{x \mid x \sqsubseteq g(x)\}$$

which means that $\text{lfp}(f) \sqsubseteq \text{lfp}(g)$ and $\text{gfp}(f) \sqsubseteq \text{gfp}(g)$. $\diamond$

Corollary 3.5 *If alternating operator pair $\langle \text{Op}_{P,1}, \text{Op}_{P,2} \rangle$ is better than $\langle \text{Op}'_{P,1}, \text{Op}'_{P,2} \rangle$, then*

$$\text{lfp}(\text{Op}'_{P,1} \circ \text{Op}'_{P,2}) \sqsubseteq \text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2}) \quad \text{and} \quad \text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1}) \sqsubseteq \text{gfp}(\text{Op}'_{P,2} \circ \text{Op}'_{P,1})$$

Proof By definition $\text{Op}'_{P,1}(I) \sqsubseteq \text{Op}_{P,1}(I)$ for every interpretation I . By anti-monotonicity of $\text{Op}'_{P,2}$ we obtain

$$(\text{Op}'_{P,2} \circ \text{Op}'_{P,1})(I) \sqsupseteq (\text{Op}_{P,2} \circ \text{Op}_{P,1})(I)$$

And as $\text{Op}'_{P,2}(I) \sqsupseteq \text{Op}_{P,2}(I)$:

$$(\text{Op}'_{P,2} \circ \text{Op}'_{P,1})(I) \sqsupseteq (\text{Op}_{P,2} \circ \text{Op}_{P,1})(I)$$

Lemma 3.4 now implies $\text{gfp}(\text{Op}_{P,2} \circ \text{Op}_{P,1}) \sqsubseteq \text{gfp}(\text{Op}'_{P,2} \circ \text{Op}'_{P,1})$.

The proof for smallest fixpoints is analogous: $\text{Op}_{P,2}(I) \sqsubseteq \text{Op}'_{P,2}(I)$ and anti-monotonicity imply

$$(\text{Op}_{P,1} \circ \text{Op}_{P,2})(I) \sqsupseteq (\text{Op}_{P,1} \circ \text{Op}'_{P,2})(I) \sqsupseteq (\text{Op}'_{P,1} \circ \text{Op}'_{P,2})(I)$$

such that Lemma 3.4 yields $\text{lfp}(\text{Op}'_{P,1} \circ \text{Op}'_{P,2}) \sqsubseteq \text{lfp}(\text{Op}_{P,1} \circ \text{Op}_{P,2})$. $\diamond$

So, the better an alternating operator pair, the closer it approximates stable models. It will now be shown that there is a best alternating operator pair and how it can be characterized. Let us first define:

$$\begin{aligned} A_P^{\sqcap\sqcup}(I) &:= \text{glb}(\bigcup \{\mathcal{A}_P(J) \mid J \sqsubseteq I\}) & A_P^{\sqcap}(I) &:= \text{glb}(\mathcal{A}_P(I)) \\ A_P^{\sqcup\sqcup}(I) &:= \text{lub}(\bigcup \{\mathcal{A}_P(J) \mid I \sqsubseteq J\}) & A_P^{\sqcup}(I) &:= \text{lub}(\mathcal{A}_P(I)) \end{aligned}$$

Proposition 3.6 *Let $\mathcal{A}_P$ be an assumption function. Then $\langle A_P^{\sqcap\sqcup}, A_P^{\sqcup\sqcup} \rangle$ is the best alternating operator pair. If $\mathcal{A}_P$ satisfies (AMS), then $A_P^{\sqcap\sqcup} = A_P^{\sqcap}$. If $\mathcal{A}_P$ satisfies (AMH), then $A_P^{\sqcup\sqcup} = A_P^{\sqcup}$.*

Proof Let us first show that $A_P^{\sqcap\sqcup}(I) \sqsubseteq J \sqsubseteq A_P^{\sqcup\sqcup}(I)$ for every $J \in \mathcal{A}_P(I)$:

$$\begin{aligned} A_P^{\sqcap\sqcup}(I) &= \text{glb}(\bigcup \{\mathcal{A}_P(J) \mid J \sqsubseteq I\}) \sqsubseteq \text{glb}(\mathcal{A}_P(I)) \\ A_P^{\sqcup\sqcup}(I) &= \text{lub}(\bigcup \{\mathcal{A}_P(J) \mid I \sqsubseteq J\}) \sqsupseteq \text{lub}(\mathcal{A}_P(I)) \end{aligned}$$

Furthermore, both $A_P^{\sqcap\sqcup}$ and $A_P^{\sqcup\sqcup}$ are anti-monotonic. For $I_1 \sqsubseteq I_2$:

$$\begin{aligned} A_P^{\sqcap\sqcup}(I_2) &= \text{glb}\left(\bigcup \{\mathcal{A}_P(J) \mid J \sqsubseteq I_2\}\right) \\ &\sqsubseteq \text{glb}\left(\bigcup \{\mathcal{A}_P(J) \mid J \sqsubseteq I_1\}\right) \\ &= A_P^{\sqcap\sqcup}(I_1) \\ A_P^{\sqcup\sqcup}(I_2) &= \text{lub}\left(\bigcup \{\mathcal{A}_P(J) \mid I_2 \sqsubseteq J\}\right) \\ &\sqsubseteq \text{lub}\left(\bigcup \{\mathcal{A}_P(J) \mid I_1 \sqsubseteq J\}\right) \\ &= A_P^{\sqcup\sqcup}(I_1) \end{aligned}$$

Hence, $\langle A_P^{\sqcap\sqcup}, A_P^{\sqcup\sqcup} \rangle$ is an alternating operator pair.

Let $\langle \text{Op}_{P,1}, \text{Op}_{P,2} \rangle$ be any alternating operator pair. By definition $\text{Op}_{P,1}(J) \sqsubseteq \text{glb}(\mathcal{A}_P(J))$. For $J \sqsubseteq I$ anti-monotonicity yields:

$$\text{Op}_{P,1}(I) \sqsubseteq \text{Op}_{P,1}(J) \sqsubseteq \text{glb}(\mathcal{A}_P(J))$$

As this relation holds for *every* $J \sqsubseteq I$, we get:

$$\text{Op}_{P,1}(I) \sqsubseteq \text{glb}\left(\bigcup \{\mathcal{A}_P(J) \mid J \sqsubseteq I\}\right) = A_P^{\sqcap\sqcup}(I)$$

And

$$\text{Op}_{P,2}(I) \sqsupseteq \text{lub}\left(\bigcup \{\mathcal{A}_P(J) \mid I \sqsubseteq J\}\right) = A_P^{\sqcup\sqcup}(I)$$

is derived analogously. ◇

If $\mathcal{A}_P$ yields singletons (i.e. $|\mathcal{A}_P(I)| = 1$ for all $I \in \mathcal{M}$), let A_P be defined by:

$$A_P(I) = J \text{ iff } \mathcal{A}_P(I) = \{J\}$$

Then both **(AMS)** and **(AMH)** coincide and state that A_P is anti-monotonic.

Corollary 3.7 *If A_P is anti-monotonic, the stable models of P form an anti-chain and $\langle A_P, A_P \rangle$ is the best alternating operator pair.*

4 Two-Valued and Three-Valued Models I

To start with familiar a concept, let $\mathcal{M}$ be the set of two-valued Herbrand interpretations with

$$\mathcal{A}_P^\forall(I) := \{I \mid I \text{ is minimal two-valued model of } P/I\}$$

We consider the following disjunctive program:

$$\begin{aligned} P_0 : \\ p \vee q \\ p \leftarrow \neg p \end{aligned}$$

It has the unique stable model $\{p\}$ and the unique stable pair $\langle \{p\}, \{p\} \rangle$. Under both concepts p is true and q false. Initially, every interpretation of $\mathcal{M}$ is a possible world. But then $\mathcal{A}_{P_0}^\forall(\mathcal{M})$ selects those worlds which remain possible under *every* pre-interpretation of negative literals. The smallest fixpoint of $\mathcal{A}_{P_0}^\forall$ is $\{\{p\}, \{q\}\}$ which requires that exactly one of p or q must be true.

Different results are obtained, when $\mathcal{M}$ becomes the set of three-valued Herbrand interpretations with $\sqsubseteq$ as defined by Przymusiński [17], namely $I_1 \sqsubseteq I_2$ iff the true atoms of I_1 are true in I_2 and the false atoms of I_2 are false in I_1 . A three-valued interpretation is subsequently denoted by its truth value assignments for p and q : **uf**, for example, states

 Figure 2: Minimal two-valued models of P_0/I

that p is undefined and q false. Finally, the assumption function assigns every three-valued interpretation the set of all minimal three-valued models of P/I .² For P_0 we obtain:

ff	$\mapsto$	{ tf }	fu	$\mapsto$	{ tf }	uf	$\mapsto$	{ tf, ut }
ft	$\mapsto$	{ tf }	uu	$\mapsto$	{ tf, ut }	tf	$\mapsto$	{ ft, tf }
tu	$\mapsto$	{ ft, tf }	ut	$\mapsto$	{ tf, ut }	tt	$\mapsto$	{ ft, tf }

We recognize two three-valued stable models, namely **tf** and **ut**. The first corresponds to the two-valued stable model $\{p\}$. The second, however, states that p may be undefined and q true.

Program P_0 furthermore illustrates that two-valued stable pairs are different from three-valued stable models.

Finally, our framework allows to introduce the concept of three-valued stable pairs and classes. In the current example **{ ft, tf }** and **{ ft, tf, ut }** are three-valued stable classes of P_0 . The smallest fixpoint of the assumption function is **{ ft, tf, ut }**.

5 Normal and Disjunctive Programs

In this section the previously introduced concepts will be illustrated by means of two-valued Herbrand interpretations. In the context of disjunctive logic programs (AC) is satisfied, because $\mathcal{A}_P^\vee$ always yields *minimal* models. For $I_1 \subseteq I_2$ we have $P/I_1 \supseteq P/I_2$. So, every model of P/I_1 is also a model of P/I_2 . As a consequence, (AMS) will hold. But (AMH) need not be true, as P_1 demonstrates:

$$\mathcal{A}_{P_1}^\vee(\emptyset) = \{\{p\}\} \text{ but } \mathcal{A}_{P_1}^\vee(\{q\}) = \{\{p\}, \{q\}\}$$

For normal programs, however, (AMH) also holds, because then every P/I has a unique smallest model.

Corollary 5.1 *If P is a normal program, then $\langle S_P, S_P \rangle$ (well-founded semantics) is the best alternating operator pair.*

Example 5.1 *It can now be seen why $\langle S_P^\vee, S_P^\vee \rangle$ fails to be an approximation of stable models in Example 2.3: In general, $S_P^\vee(I)$ will not be a superset of J for $J \in \mathcal{A}_P^\vee(I)$. For P_1 we have $\mathcal{A}_{P_1}^\vee(\{q\}) = \{\{p\}, \{q\}\}$ and $S_{P_1}^\vee(\{q\}) = \emptyset$.*

Corollary 5.2 *If P is a disjunctive program, $\langle S_P^\vee, S_P^{\min} \rangle$ is the best alternating operator pair, where*

$$S_P^{\min}(I) = \bigcup_{I \subseteq J} \{A \mid A \text{ is in some minimal model of } P/J\}$$

²Przymusiński also defines a suitable Gelfond-Lifschitz transformation for the three-valued case.

Example 5.2 For P_1 we have

$$S_{P_1}^{\min}(\emptyset) = \bigcup \{ \{p\}, \{q\} \} = \{p, q\}$$

In accordance with Proposition 3.2 the alternating operator pair “embraces” both of the stable models:

$$\text{lfp}(S_{P_1}^{\vee} \circ S_{P_1}^{\min}) = \emptyset \quad \text{and} \quad \text{gfp}(S_{P_1}^{\min} \circ S_{P_1}^{\vee}) = \{p, q\}$$

Example 5.3 The following program

$$\begin{aligned} P_3 : \\ p \vee q \\ r \leftarrow \neg p \\ r \leftarrow \neg q \end{aligned}$$

has stable models $\{p, r\}$ and $\{q, r\}$ while

$$\text{lfp}(S_{P_3}^{\vee} \circ S_{P_3}^{\min}) = \emptyset \quad \text{and} \quad \text{gfp}(S_{P_3}^{\min} \circ S_{P_3}^{\vee}) = \{p, q, r\}$$

As the above two stable models are also perfect (for the definition of perfect models see [16]), this example demonstrates that no well-founded semantics in the sense of Definition 3.3, i.e. based on anti-monotonic operators, is able to coincide with perfect model semantics.

Definition 5.1 Every normal rule $A_i \leftarrow L_1, \dots, L_n$ ($1 \leq i \leq m$) is a normalization of the disjunctive rule

$$A_1 \vee \dots \vee A_m \leftarrow L_1, \dots, L_n$$

Let P be a disjunctive program. Then P^{N+} denotes the strong normalization of P which is obtained by replacing every disjunctive rule by the set of all of its normalizations.³

Corollary 5.3 Let P be a disjunctive program and P_{N+} its strong normalization. Then $\langle S_P^{\vee}, S_{P_{N+}} \rangle$ is an alternating operator pair.

Example 5.4 The strong normalization of P_1 contains both facts p and q . We obtain:

$$\text{lfp}(S_{P_1}^{\vee} \circ S_{P_{N+}}) = \emptyset \quad \text{and} \quad \text{gfp}(S_{P_{N+}} \circ S_{P_1}^{\vee}) = \{p, q\}$$

In analogy to the strong normalization P_{N+} which specifies an upper bound for what can be derived from P , a lower bound P_{N-} can also be given in terms of normal programs, essentially by removing all non-normal rules from P . It has been shown in [8] that the application of so-called knowledge compilation techniques then allows to improve $\langle S_{P_{N-}}, S_{P_{N+}} \rangle$.

6 Two-Valued and Three-Valued Models II

Let us return to P_0 in the context of two-valued interpretations. The fixpoints

$$\text{lfp}(S_{P_0}^{\vee} \circ S_{P_0^{N+}}) = \emptyset \quad \text{and} \quad \text{gfp}(S_{P_0^{N+}} \circ S_{P_0}^{\vee}) = \{p, q\}$$

state that *every world between* $\emptyset$ and $\{p, q\}$ is a possible one. Note that this is quite more liberal than the smallest fixpoint of $\mathcal{A}_{P_0}^{\vee}$.

In the three-valued context the smallest and greatest model are **ff** and **tt**. Interestingly, after the three-valued Gelfond-Lifschitz transformation we end up with a set of rules where the **u**-symbol does not occur. The lower and upper bound computed by $\langle S_P^{\vee}, S_{P_{N+}} \rangle$ will be classical models again and will remain classical through all iterations.

In other words: Alternating operator pairs yield the same result as they did in the two-valued case with the additional effect that

$$\text{lfp}(S_{P_0}^{\vee} \circ S_{P_0^{N+}}) \sqsubseteq I \sqsubseteq \text{gfp}(S_{P_0^{N+}} \circ S_{P_0}^{\vee})$$

for every three-valued stable model I .

³The strong normalization determines the so-called *Weak Generalized Closed World Assumption* of positive programs [11].

Project	ProjectName	Location	Employee	Name	Salary
	HumGen	Boston		Jim	4,000
	XenoTrans	London		John	4,500
	GenTec	York		George	3,500

EP	Employee	Project
	Jim	XenoTrans
	John	HumGen
	George	XenoTrans
	George	GenTec

Figure 3: Relations with Maybe Tuples

7 Relations with Maybe Tuples

Assumption functions express that there are many ways to pre-interpret negative literals in rules of logic programs. Every hypothetical pre-interpretation comes with a set of intended models. It is by the concept of stable models or by iterative applications of an alternating operator pair that we get to know which hypotheses are “reasonable”. We will show that this framework can be naturally combined with representations of incomplete information which are different from disjunctions.

In deductive databases one often distinguishes extensional relations, which are explicitly given by a collection of tuples, from intensional relations, which are implicitly defined by rules. In our notation: We have a set of ground atoms $p(c_1, \dots, c_n)$ for every extensional relation symbol p , and for every intensional relation symbol q there are some range-restricted rules with q in the rule head. Once again, for semantic studies it is convenient to identify rules with all their ground instantiations.

Let us now assume that our knowledge about extensional relation p is incomplete in the following sense: For some tuples we know that they are certainly true or certainly false, but there are some tuples whose truth status we do not know.

Example 7.1 *In the EP relation in Figure 3 tuples below the horizontal line are maybe tuples. So, John is assigned to the HumGen project, Jim is definitely not involved in HumGen, and George may be working for XenoTrans or GenTec or both.*

A logic program with maybe tuples P consists of two disjoint sets of ground atoms denoted by $\text{SURE}(P)$ (those which are certainly true) and $\text{MAYBE}(P)$ (those which are possibly true) and a set of normal rules.

Definition 7.1 *Let P be a program with maybe tuples. Program P^c is called a completion of P , if it consists of the rules of P together with a set of ground atoms $F \subseteq \text{HB}$ such that $\text{SURE}(P) \subseteq F \subseteq \text{SURE}(P) \cup \text{MAYBE}(P)$.*

As in Section 5 an interpretation is a subset of the Herbrand base. Intended interpretations can be obtained by selecting some of the possibly true ground atoms. The intended models of P pre-interpreted with respect to I are the smallest models of P^c/I for every completion P^c . The assumption function is defined as follows:

$$\mathcal{A}_P^{\text{maybe}}(I) = \{ J \mid \text{there is a completion } P^c \text{ of } P \text{ such that } J \text{ is the smallest model of } P^c/I \}$$

An interpretation I is a *stable model* of P , if it is the smallest model of P^c/I for some completion P^c . It is not difficult to see that the assumption function does not satisfy (AC), but (AMS) and (AMH). Obviously, every program has a smallest and a largest completion P^{c-} and P^{c+} .

Corollary 7.1 $\langle S_{P^{c-}}, S_{P^{c+}} \rangle$ is the best alternating operator pair.

Example 7.2 *A query for all projects which have no employee assigned to them can be expressed as*

$$q1(\text{PName}, \text{Loc}) \leftarrow \text{project}(\text{PName}, \text{Loc}), \text{employee}(\text{Emp}, \text{Sal}), \neg \text{ep}(\text{Emp}, \text{PName})$$

EP	Employee	Project
	Jim	XenoTrans
	John	HumGen
	-	XenoTrans
	George	-

Figure 4: Relation with Null Values

There are stable models which contain $q1(\text{GenTec}, \text{York})$, because George's involvement in GenTec is not for sure. But none has $q1(\text{HumGen}, \text{Boston})$, because John is certainly working for HumGen.

The $q1$ -extension in $\text{lfp}(S_{Pc-}^V \circ S_{Pc+})$ is empty and in $\text{gfp}(S_{Pc+} \circ S_{Pc-}^V)$ it consists of $q1(\text{GenTec}, \text{York})$.

Example 7.3 Consider the disjunctive program:

P_4 :
 $p(a) \vee p(b)$
 $p(a) \leftarrow \neg p(c)$

It has the unique stable model $\{p(a)\}$. Furthermore:

$$\begin{aligned} \text{lfp}(S_{P_4}^V \circ S_{P_4}^{\min}) &= \{p(a)\} \quad \text{and} \quad \text{gfp}(S_{P_4}^{\min} \circ S_{P_4}^V) = \{p(a)\} \\ \text{lfp}(S_{P_4}^V \circ S_{P_4}^{N+}) &= \{p(a)\} \quad \text{and} \quad \text{gfp}(S_{P_4}^{N+} \circ S_{P_4}^V) = \{p(a), p(b)\} \end{aligned}$$

Rule $p(a) \leftarrow \neg p(c)$ together with maybe tuples $p(a)$ and $p(b)$ has two stable models $\{p(a)\}$ and $\{p(a), p(b)\}$. Furthermore:

$$\text{lfp}(S_{P_4c-}^V \circ S_{P_4c+}) = \{p(a)\} \quad \text{and} \quad \text{gfp}(S_{P_4c+} \circ S_{P_4c-}^V) = \{p(a), p(b)\}$$

8 Relations with Null Values

8.1 Motivation

Null values are a popular means of representing missing information in databases. Zaniolo [24] reports that 14 different types of nulls have been distinguished in an ANSI/SPARC interim report. In what follows we assume that a null is to represent an existing but unknown value in a database tuple.

Example 8.1 Consider the EP relation in Figure 4. The query for projects without employees as expressed in SQL reads:

```
SELECT s.ProjectName, s.Location
FROM   Project s
WHERE  NOT EXISTS (
        SELECT t.Employee
        FROM   EP t
        WHERE  t.Project=s.ProjectName
    )
```

It will return the answer "GenTec". The reason is that the comparison $t.Project=s.ProjectName$ fails, whenever $t.Project$ is null. So there is no join partner for GenTec Project.

But the answer is a rather optimistic one in the light of the actual database instance: It may well be that George is involved in GenTec.

A normal program with null values P consists of a set F of ground atoms together with a set of normal rules. In F a term may be a constant or the null value symbol.

EP	Employee	Project	EP	Employee	Project
	Jim	XenoTrans		Jim	XenoTrans
	John	HumGen		John	HumGen
	George	XenoTrans		George	XenoTrans
				George	GenTec

Figure 5: Completions of EP

Example 8.2 Relations Project and Employee as in Figure 3 together with relation EP of Figure 4 and the rule

$$q1(PName, Loc) \leftarrow project(PName, Loc), employee(Emp, Sal), \neg ep(Emp, PName)$$

form a normal program with null values.

Intuitively spoken, null values are placeholders for constants and possible worlds are obtained by replacing them.

Definition 8.1 Let P be a normal program with null values. Program P^c is called a completion of P , if it can be obtained by replacing every null value of P by a constant of $\mathcal{U}$.

Example 8.3 Figure 5 depicts two of the many possible completions of relation EP. In the first completion null values have been substituted by Jim and XenoTrans, in the second by George and GenTec.

8.2 Interpretations

Of course, in the process of query evaluation we cannot generate every completion of P . While completions will be the formal basis for stable models, alternating operator pairs will not be defined in terms of null value replacements. They will treat nulls as distinguished constants.

Example 8.4 Assume we are interested in project-employee associations:

$$q2(PName, Loc, EmpName) \leftarrow project(PName, Loc), ep(EmpName, PName)$$

Atoms $q2(XenoTrans, London, Jim)$ as well as $q2(HumGen, Boston, John)$ are implied by every completion. If we intend to compute a set of sound answers on the basis of P instead of P^c , the null value must not be matched with any other constant, thus resembling existential quantification.

Figure 5 illustrates that atoms $q2(XenoTrans, London, George)$ and $q2(GenTec, York, George)$ are implied by some but not by all completions. When aiming at a complete superset of answers, it seems that null values should be unifiable with any constant (universal quantification).

The example shows that null values have to be treated in different ways in the computation of lower and upper bounds of stable models. We will refer to two new symbols for this purpose: ‘ $\perp$ ’ which is less than any constant c and cannot be matched with c , and ‘ $\top$ ’ which stands for and can be unified with arbitrary constants.

The flat Herbrand universe $\mathcal{U}$ is augmented accordingly: Let $\mathcal{U}^\oplus := \mathcal{U} \cup \{\perp, \top\}$ with $\perp \sqsubseteq c \sqsubseteq \top$ for all $c \in \mathcal{U}$. The augmented Herbrand base is denoted by $HB^\oplus$. Then

$$p(c_1, \dots, c_n) \sqsubseteq p(c'_1, \dots, c'_n)$$

if $c_i \sqsubseteq c'_i$ for every $1 \leq i \leq n$. An interpretation is a subset of $HB^\oplus$ with the restriction that for all $A, B \in I$ the following holds:

$$A \sqsubseteq B \Rightarrow A = B$$

Every $X \subseteq HB^\oplus$ can be reduced to an interpretation by

$$reduce(X) := \{ A \mid A \in X \text{ and there is no } B \in X \text{ with } A \sqsubset B \}$$

q2	Project	Location	Employee	q2	Project	Location	Employee
	XenoTrans	London	Jim		XenoTrans	London	Jim
	HumGen	Boston	John		HumGen	Boston	John
	XenoTrans	London	George		XenoTrans	London	George
					GenTec	York	George

Figure 6: Stable Models of q2

The collection of all interpretations is a complete lattice with

$$\begin{aligned}
 I_1 \sqsubseteq I_2 & \text{ iff } \forall A \in I_1 \exists B \in I_2 : A \sqsubseteq B \\
 I_1 \sqcap I_2 & = \text{reduce}(\{ A \mid \text{there are } A_1 \in I_1 \text{ and } A_2 \in I_2 \text{ with } A = A_1 \sqcap A_2 \}) \\
 I_1 \sqcup I_2 & = \text{reduce}(I_1 \cup I_2)
 \end{aligned}$$

The smallest lattice element is the empty set and the greatest element consists of $p(\top, \dots, \top)$ for every relation symbol p .

Interpretation I satisfies $A \in \text{HB}^\oplus$ (in symbols: $I \models A$), if $B \in I$ for some $B \sqsupseteq A$. Let $A_0 \leftarrow A_1, \dots, A_n$ be a positive rule that is ground in $\mathcal{U}^\oplus$. Then I is a model of this rule, if $I \models A_0$, whenever $I \models A_i$ for every $1 \leq i \leq n$.

8.3 Assumption Functions and Operator Pairs

The next definition adapts the Gelfond-Lifschitz transformation for the augmented Herbrand base $\text{HB}^\oplus$.

Definition 8.2 Let P be an instance of a program with null values and I an interpretation. The transformation P/I of P modulo I is defined by the following procedure:

- Remove every rule whose body contains some $\neg A$ such that $I \models A$.
- Delete every expression $\neg A$ in the remaining rules.

Assumption functions are formally the same as for relations with maybe tuples:

$$\mathcal{A}_P^{\text{null}}(I) = \{ J \mid \text{there is a completion } P^c \text{ of } P \text{ such that } J \text{ is the smallest model of } P^c/I \}$$

Every interpretation in $\mathcal{A}_P^{\text{null}}(I)$ is a subset of the non-augmented HB. This reflects our intention that intended models should correspond with possible worlds, and there are no $\perp$'s and $\top$'s in a possible world.

Example 8.5 Figure 6 depicts the q2-extension in two of the stable models.

Let us turn to alternating operator pairs now. The idea behind $\perp$ and $\top$ was that the former allows approximations from below and the latter from above. In the computation of the lower bound a null value should not be unified with a constant or another null. Therefore $\perp$ will not be allowed to substitute a variable that has already been replaced by a constant or a null within the same rule. In the computation of the upper bound nulls should be unifiable with arbitrary constants and nulls. So the same variable may be replaced by $\top$ as well as constants.

Definition 8.3 Let P be a normal program with null values.

The lower bound P^- is the collection of all rules of P , after null values have been replaced by $\perp$ and all variables been instantiated in $\mathcal{U}^\oplus$ in accordance with the following restrictions:

- In positive literals variables may be substituted by constants as well as $\perp$.
- In negative literals variables may be substituted by constants.
- The same variable may be replaced by $\perp$ only once within the body and once within the head.

- The same variable must not be replaced by different symbols.
- If a variable is replaced by a constant somewhere, it must also be replaced by that constant in some positive body literal.

The upper bound P^+ is obtained from P by replacing null values by $\top$ and instantiating variables in $\mathcal{U}^\oplus$. The following restrictions have to be obeyed:

- In positive literals variables may be substituted by constants as well as $\top$.
- In negative literals variables may be substituted by constants as well as $\top$.
- The same variable must not be replaced by different constants.
- If a variable is replaced by a constant somewhere, it must also be replaced by that constant in some positive body literal.
- If a variable is replaced by a constant somewhere, it must not be replaced by $\top$ in the head or in a negative literal.

It is now possible to implement query evaluation via S_P . It just has to be ensured that the result is a valid interpretation:

$$S_P^{\text{null}}(I) := \text{reduce}(\text{lfp}(T_{P/I}))$$

Proposition 8.1 *Let P be a normal program with null values. Then $\langle S_{P^-}^{\text{null}}, S_{P^+}^{\text{null}} \rangle$ is an alternating operator pair.*

Lemma 8.2 *Let P^c be a completion of P/I and J its smallest model. Then $S_{P^-}^{\text{null}}(I) \sqsubseteq J$.*

Proof We show: If $A \in \text{lfp}(T_{P^-/I})$, then there is some $A^c \in \text{lfp}(T_{P^c/I})$ with $A^c \sqsupseteq A$ and $A^c \in \text{HB}$.

The proof is by induction on the number of $T_{P^-/I}$ -applications. Consider $A \in T_{P^-/I}^{n+1}(\emptyset)$. There is a rule in P^- :

$$A \leftarrow B_1, \dots, B_k, \neg B_{k+1}, \dots, \neg B_l$$

with B_i in $T_{P^-/I}^n(\emptyset)$ for $1 \leq i \leq k$ and no $C_j \sqsupseteq B_j$ in I for $k+1 \leq j \leq l$. This rule is an instantiation of a P -rule in accordance with definition 8.3. In P^c there is a corresponding rule which in some places has constants where the P^- -rule has $\perp$. If $B_i \in \text{HB}$ for $1 \leq i \leq k$, then $B_i \in T_{P^c/I}^n(\emptyset)$ by induction hypothesis. If $\perp$ occurs in B_i , then by definition of P^- it substitutes a variable which occurs exactly once within the body. By induction hypothesis there is $B_i^c \in T_{P^c/I}^n(\emptyset)$ with $B_i^c \sqsupseteq B_i$. Consider the rule instance where B_i^c is placed instead of B_i . Note that $B_j \in \text{HB}$ for $k+1 \leq j \leq l$ such that $B_j^c = B_j$. Then $T_{P^c/I}^{n+1}(\emptyset)$ contains $A^c \sqsupseteq A$. $\diamond$

Lemma 8.3 *Let P^c be a completion of P/I and J its smallest model. Then $J \sqsubseteq S_{P^+}^{\text{null}}(I)$.*

Proof We show: If $A^c \in \text{lfp}T_{P^c/I}$, then there is some $A \in \text{lfp}(T_{P^+/I})$ with $A \sqsupseteq A^c$.

The proof is by induction on the number of $T_{P^c/I}$ -applications. Consider $A^c \in T_{P^c/I}^{n+1}(\emptyset)$. There is a rule in P^c :

$$A^c \leftarrow B_1^c, \dots, B_k^c, \neg B_{k+1}^c, \dots, \neg B_l^c$$

with B_i^c in $T_{P^c/I}^n(\emptyset)$ for $1 \leq i \leq k$ and no $C_j \sqsupseteq B_j^c$ in I for $k+1 \leq j \leq l$. This rule is an instantiation of a P -rule. In P^+ it has an instance where constants may be replaced by $\top$. By induction hypothesis $T_{P^+/I}^n(\emptyset)$ contains $B_i \sqsupseteq B_i^c$ for $1 \leq i \leq k$. So consider that rule instance in P^+ , where B_i occurs in place of B_i^c . Moreover, in B_j some constants may be replaced by $\top$, i.e. $B_j \sqsupseteq B_j^c$. But as I contains no $C_j \sqsupseteq B_j^c$, it will not contain $C_j \sqsupseteq B_j$ either. Hence: $T_{P^+/I}^{n+1}(\emptyset)$ contains $A \sqsupseteq A^c$. $\diamond$

Example 8.6 Consider the program P :

$$\begin{aligned} & s(a, b) \\ & p(-) \\ & q(-) \\ & r(X) \leftarrow p(X), \neg q(X) \end{aligned}$$

Then $\{s(a, b), p(a), q(a)\}$ and $\{s(a, b), p(a), q(a), r(a)\}$ are some stable models. With $\mathcal{U} = \{a, b\}$, we get:

$P^- :$ $s(a, b)$ $p(\perp)$ $q(\perp)$ $r(a) \leftarrow p(a), \neg q(a)$ $r(b) \leftarrow p(b), \neg q(b)$	$P^+ :$ $s(a, b)$ $p(\top)$ $q(\top)$ $r(a) \leftarrow p(a), \neg q(a)$ $r(b) \leftarrow p(b), \neg q(b)$ $r(\top) \leftarrow p(\top), \neg q(\top)$
--	--

Therefore, the extension of r is empty in $\text{lfp}(S_{P^-}^{\text{null}} \circ S_{P^+}^{\text{null}})$, whereas $\text{gfp}(S_{P^-}^{\text{null}} \circ S_{P^+}^{\text{null}})$ contains $r(\top)$.

Null values are not exchanged between S_{P^-} and S_{P^+} . In other words: Null values can be combined with negation in recursive rules, but the nulls are only effective within every single computation of S_{P^-} and S_{P^+} .

Example 8.7 For Example 8.2 we now obtain the correct answer, namely the empty $q1$ -extension in $\text{lfp}(S_{P^-}^{\text{null}} \circ S_{P^+}^{\text{null}})$ and $q1(\text{GenTec}, \text{York})$ in $\text{gfp}(S_{P^-}^{\text{null}} \circ S_{P^+}^{\text{null}})$.

9 Related Work

Several semantics have been proposed for disjunctive logic programs (see [11]). In [19] Przymusiński proposes the novel STATIC semantics for disjunctive programs. It is formulated in terms of a propositional modal language with belief operator $\mathbf{B}$. Negative program literals are translated as $\mathbf{B} \neg A$. The approach is based on two entailment relations:

- If T is a belief theory, then $T \models_{\min} \alpha$ states that α is true in every minimal model of T .
- The logical consequences of T augmented with axiom $\mathbf{B} \alpha \rightarrow \neg \mathbf{B} \neg \alpha$ and $\mathbf{B}(\alpha \wedge \beta) \equiv \mathbf{B} \alpha \wedge \mathbf{B} \beta$ are denoted by $\text{Con}^*(T)$.

A static expansions of T is a belief theory $\hat{T}$ such that

$$\hat{T} = \text{Con}^*(T \cup \{\mathbf{B} \alpha \mid \hat{T} \models_{\min} \alpha\})$$

If P be a disjunctive program and $T(P)$ its translation, the smallest static expansion can be obtained by the following iterative procedure [19]:

$$\begin{aligned} T^0 &= \text{Con}^*(T(P)) \\ T^{n+1} &= \text{Con}^*(T(P) \cup \{\mathbf{B} \alpha \mid T^n \models_{\min} \alpha\}) \end{aligned}$$

There is a close relationship between $\mathcal{A}_P^Y$ and STATIC, and $\langle S_P^Y, S_{P^+} \rangle$ is sound with respect to STATIC, but this theme cannot be discussed here.

Brass and Dix [3] propose to characterize logic program semantics by abstract properties. D-WFS is the weakest semantics which satisfies natural and elementary program transformations like unfolding. They also develop a bottom-up evaluation algorithm and conjecture that D-WFS equals STATIC.

In [9] we presented some more assumption functions and operator pairs for disjunctive programs and also discussed the case where $\mathcal{M}$ stands for *states* (closed sets of disjunctions) instead of models. The different notions of stability as self-justifiable models or pairs of models or sets of models, however, generalize the concepts presented in [9]. We also believe that the current approach is more intuitive.

In [12] Maier presents a survey of several approaches where the meaning of relations with null values is given in terms of so-called *possibility functions* POSS. A possibility function could, for example, assign a relation with null values the set of its completions. The goal then is to define a modified algebra operation γ' on relations r'_i with null values r' for every operation γ on relations r_i without null values such that POSS is *adequate*:

$$\text{POSS}(r'_1 \gamma' r'_2) \supseteq \text{POSS}(r'_1) \gamma \text{POSS}(r'_2)$$

where

$$\text{POSS}(r'_1) \gamma \text{POSS}(r'_2) := \{ r_1 \gamma r_2 \mid r_1 \in \text{POSS}(r'_1) \text{ and } r_2 \in \text{POSS}(r'_2) \}$$

Amongst others Maier discusses Biskup's approach where relations consist of sure and of maybe tuples. It seems that except from negation S_{P-}^{null} coincides with the computation of sure tuples and S_{P+}^{null} coincides with that of maybe tuples in Biskup's approach.

Buneman et al. [4] study a generalization of partial tuples for complex objects and Libkin [10] gives a general account of approximations in databases. As mentioned in the introduction, we regard the enhancement of logic programs with more sophisticated representation mechanisms as highly important. The combination of ideas presented in this paper with complex objects seems to be promising.

10 Conclusions

Stable models have been defined in a generalized setting, where every hypothetical pre-interpretation of a negative literals is assigned a set of intended models. Well-founded approximations have then been defined by a pair of anti-monotonic operators. We have been able to characterize the best alternating operator pair and to simplify its definition in the presence of anti-monotonicity restrictions.

The framework has first been applied in the context of disjunctive logic programs based on two-valued as well as three-valued logic. It has then been illustrated that programs with incomplete information can be given hypothetical pre-interpretations in an analogous way. In the context of maybe tuples the hypothesis describes which maybe tuples should be included in the database. In the context of null values it specifies substitutions of nulls by constants. Our approach then yields a uniform framework where on the semantic side stable models can be defined and on the operational side sound query evaluation can be based on standard operators.

References

- [1] Chitta Baral and V. S. Subrahmanian. Dualities between alternative semantics for logic programming and non-monotonic reasoning. In *Proc. of the 1st Int. Workshop on Logic Programming and Non-monotonic Reasoning*, pages 69–86. MIT Press, 1991.
- [2] Chitta R. Baral and V. S. Subrahmanian. Stable and extension class theory for logic programs and default logics. *Journal of Automated Reasoning*, 8:345–366, 1992.
- [3] Stefan Brass and Jürgen Dix. Disjunctive semantics based upon partial and bottom-up evaluation. In *Proc. of the 12th Int. Conf. on Logic Programming*, pages 199–213. MIT Press, 1995.
- [4] Peter Buneman, Susan Davidson, and Aaron Waters. A semantics for complex objects and approximate queries. In *Proc. of the 7th Symp. on Principles of Database Systems*, pages 305–314, 1988.
- [5] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In ICLP88 [7], pages 1070–1080.
- [6] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
- [7] *Proc. of the 5th Int. Conf. and Symp. on Logic Programming*. MIT Press, 1988.

- [8] Jürgen Kalinski. From above and from below: Approximating stable models. In Bernd Wolfinger, editor, *Innovationen bei Rechen- und Kommunikationssystemen, GI-Fachgespräch: Disjunctive logic programming and disjunctive databases*, pages 61–68. Springer, 1994.
- [9] Jürgen Kalinski. Stable classes and operator pairs for disjunctive programs. In V.W. Marek, A. Nerode, and M. Truszczyński, editors, *Proc. of the 3rd Int. Conference on Logic Programming and Nonmonotonic Reasoning*, pages 358–371. Springer, Lecture Notes in Artificial Intelligence 928, 1995.
- [10] Leonid Libkin. Approximation in databases. In Georg Gottlob and Moshe Y. Vardi, editors, *Proc. of the 5th Int. Conf. on Database Theory*, pages 411–424. Springer, Lecture Notes in Computer Science 893, 1995.
- [11] Jorge Lobo, Jack Minker, and Arcot Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, 1992.
- [12] David Maier. *The Theory of Relational Databases*. Pitman, 1983.
- [13] Wiktor Marek and Mirosław Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3):588–619, 1991.
- [14] Jack Minker and Arcot Rajasekar. A fixpoint semantics for disjunctive logic programs. *Journal of Logic Programming*, 9:45–74, 1990.
- [15] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(4):75–94, 1985.
- [16] Teodor Przymusiński. Perfect model semantics. In ICLP88 [7], pages 1081–1096.
- [17] Teodor C. Przymusiński. Extended stable semantics for normal and disjunctive programs. In *Proc. of the 7th Int. Conf. on Logic Programming*, pages 459–477. MIT Press, 1990.
- [18] Teodor C. Przymusiński. Stable semantics for disjunctive programs. *New Generation Computing*, 9:401–424, 1991.
- [19] Teodor C. Przymusiński. Static semantics for normal and disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, Special Issue on Disjunctive Programs (to appear), 1995.
- [20] Konstantinos Sagonas, Terrence Swift, and David S. Warren. An abstract machine for computing the well-founded semantics. In *Proc. of the 1996 Joint Int. Conf. and Symp. on Logic Programming*, pages 274–288. MIT Press, 1996.
- [21] 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.
- [22] 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.
- [23] Allen Van Gelder. The alternating fixpoint of logic programs with negation. In *Proc. of the 8th Symp. on Principles of Database Systems*, pages 1–10, 1989.
- [24] Carlo Zaniolo. Database relations with null values. *Journal of Computer and System Sciences*, 28:142–166, 1984.