A Comparison of the Static and the Disjunctive Well-founded Semantics and its Implementation

Stefan Brass
Computer Science
University of Hildesheim
Marienburger Platz 22
31141 Hildesheim, Germany
brass@uni-hildesheim.de

Jürgen Dix
Computer Science
University of Koblenz
Rheinau 1
56075 Koblenz, Germany
dix@uni-koblenz.de

Ilkka Niemelä
Computer Science
Helsinki University of Tech.
P.O.Box 1100
02015 HUT, Finland
Ilkka.Niemela@hut.fi

Teodor C. Przymusinski*
Computer Science
University of California
Riverside
CA 92521, USA
teodor@cs.ucr.edu

Abstract

In recent years, much work was devoted to the study of theoretical foundations of *Disjunctive Logic Programs* and *Disjunctive Deductive Databases*. While the semantics of non-disjunctive programs is fairly well understood the declarative and computational foundations of disjunctive programming proved to be much more elusive and difficult. Recently, two new and very promising semantics have been proposed for the class of disjunctive logic programs. Both of them extend the wellfounded semantics of normal programs. The first one is the *static semantics* proposed by Przymusinski and the other is the *disjunctive wellfounded semantics* proposed by Brass and Dix. Although the two semantics are based on very different ideas, we show in this paper that they turn out to be very closely related. In fact, we show that it is possible to restrict the underlying language of STATIC to get D-WFS. We also show how to use this characterization for an implementation based a circumscriptive theorem prover.

1 Introduction

Recently, considerable interest and research effort has been devoted to disjunctive logic programming, disjunctive deductive databases and to various extensions of non-monotonic formalisms ensuring a proper treatment of disjunctive information.

There are good reasons justifying this extensive research effort. In natural discourse as well as in various programming applications we often use *disjunctive statements*. One particular example of such a situation is *reasoning by cases*. Other obvious examples include:

- *Approximate information:* for instance, an age "around 30" can be 28, 29, 30, 31, or 32;
- *Legal rules:* the judge always has some freedom for his decision, otherwise he/she would not be needed; so laws cannot have unique models;
- *Diagnosis:* only at the end of a fault diagnosis we may know exactly which part of some machine was faulty but as long as we are searching, different possibilities exist;
- *Biological inheritance:* if the parents have blood groups A and 0, the child must also have one of these two blood groups (example from [Lip79]);
- *Natural language understanding:* here there are many possibilities for ambiguity and they are represented most naturally by multiple intended models;
- *Reasoning about concurrent processes:* since we do not know the exact sequence in which certain operations are performed, again multiple models come into play;
- *Conflicts in multiple inheritance:* if we want to keep as much information as possible, we should assume disjunction of the inherited values [BL93].

Formalisms promoting disjunctive reasoning are more *expressive* and *natural to use* since they permit direct translation of disjunctive statements from natural language and from informal specifications. The additional expressive power of disjunctive logic programs [EG93, EGM94, EG96] significantly simplifies the problem of *translation* of non-monotonic formalisms into logic programs, and, consequently, facilitates using logic programming as an *inference engine*

Partially supported by the National Science Foundation grant #IRI-9313061.

for non-monotonic reasoning. Moreover, extensive recent work devoted to theoretic and algorithmic foundations of disjunctive programming suggests that there are good prospects for *extending the logic programming paradigm* to disjunctive programs.

However, the issue of finding a suitable semantics for disjunctive programs and databases proved to be far more complex than it was in the case of normal, non-disjunctive programs[1]. Quite recently, however, two new and very promising semantics have been proposed for the class of disjunctive logic programs. Both of them extend the wellfounded semantics of normal programs. The first one is the *static semantics* proposed by Przymusinski in [Prz95] and the other is the *D-WFS semantics* proposed by Brass and Dix in [BD94]. The two semantics are based on very different intuitions and are constructed by completely different means. While *D-WFS* is defined as the least semantics that is invariant under certain natural program transformations, the static semantics is obtained by first translating a logic program into a belief theory in the *Autoepistemic Logic of Beliefs (AEB)* and then constructing its *least static expansion* (defined as the least fixed point of a natural monotonic operator).

This paper is devoted to an in-depth study of the relationship between the two semantics. In spite of the fact that the two semantics attempt to provide a meaning for arbitrary disjunctive programs based on very different ideas, they turn out to be closely related. It is fairly easy to show that the static semantics satisfies the abstract properties on which the definition of D-WFS is based and therefore it is always at least as strong as D-WFS. However, because of the fact that the two semantics are defined using quite *different languages*, with a much richer language allowed by the static semantics, it is easy to give examples showing that, in general, the two semantics are different.

However, it turns out that when they are restricted to a common language, which we call *data base logic*, the two semantics become entirely *equivalent*. This fundamental result uses a nice and powerful characterization of STATIC$_{DBL}$, the static semantics restricted to the language $\mathcal{L}_{DBL}$. We consider this result to be quite significant. The *D-WFS* semantics [BD94] is based on the idea of generating the smallest "well-behaved" disjunctive semantics, i.e., the smallest semantics which satisfies some intuitive and natural structural properties. On the other hand, the static semantics [Prz95] is based on viewing disjunctive logic programs as special epistemic theories in which (introspective) beliefs are based on minimal model entailment (circumscription). The original papers [BD94, Prz95] (see also [Prz98, BD98a, BD98b]) contain a thorough discussion of the two semantics, compare them to other semantics and argue that they both represent natural and intuitive semantics for disjunctive programs. Our main result shows that as long as both semantics are restricted to the common language they become identical and therefore combine the benefits of both semantics. Consequently, this discovery constitutes yet one more powerful argument in favor of the two semantics.

Our main result not only clarifies the declarative meaning of the two semantics. It turns out that it also allows us to further clarify their *procedural semantics* by establishing new computational results. Namely, recently one of the authors [Nie96b, Nie96a] introduced a novel method for implementing circumscription (with fixed and varying predicates) which has polynomial space complexity and is able to handle circumscriptive reasoning also in the case when the set of premises has a large number of minimal models. Our second main result shows how this method can be nicely adapted for the computation of the D-WFS and static semantics.

Our results are obtained by using the characterization of least static expansions from [BDP96] and a characterization of D-WFS contained in [BD98a].

The paper is organized as follows. In the next Section 2 we recall the definition and basic properties of disjunctive logic programs and semantics as well as the modal framework of the *autoepistemic logic of beliefs* (AEB). In Section 3 we briefly recall the semantics D-WFS and STATIC. We introduce in Section 4 the restricted language $\mathcal{L}_{DBL}$ with non-nested beliefs, *database logic*, and characterize the least static expansion in this language — this defines the semantics STATIC$_{DBL}$. We then show, using our characterization, that STATIC$_{DBL}$ coincides with D-WFS: STATIC$_{DBL}(P)$ = D-WFS(P). In Section 5 we consider how to use our characterizations for an efficient implementation based on methods introduced recently by Niemelä in [Nie96b, Nie96a]. Related approaches are discussed in Section 6. We end with the conclusions in Section 7.

2 Preliminaries

We introduce in this section the necessary notions used throughout the paper. In Subsection 2.1 we consider disjunctive programs and semantics, while in Subsection 2.2 we introduce the modal framework of the au-

[1] The book by Minker et. al. [LMR92] provides a detailed and well-organized account of the extensive research effort in this area. See also [Dix95, Min96].

toepistemic logic of beliefs.

2.1 Disjunctive Logic Programs and Their Semantics

We consider instantiated programs over some fixed finite language $\mathcal{L}$ containing the binary connectives $\vee, \wedge, \leftarrow$, the unary connective *not* and the *falsum* $\bot$. We write $At_{\mathcal{L}}$ for the set of all atoms different from $\bot$.

Definition 2.1 (Logic Program P, heads(P)) *A logic program P is a finite set of rules of the form*

$$A_1 \vee \cdots \vee A_k \leftarrow B_1 \wedge \cdots \wedge B_m \wedge not\, C_1 \wedge \cdots \wedge not\, C_n,$$

where the $A_i/B_i/C_i$ are $\mathcal{L}$-atoms different from $\bot$, $k \geq 1$, $m \geq 0$, $n \geq 0$. We allow empty conjunctions: they are considered as abbreviations for not $\bot$ (the verum*).*

We identify such a rule with the pair consisting of the following set of atoms $\mathcal{A} := \{A_1, \ldots, A_k\}$, and the set of atoms and not-atoms $\mathcal{B}ody := \{B_1, \ldots, B_m, not\, C_1, \ldots, not\, C_n\}$, and write it as

$$\mathcal{A} \leftarrow \mathcal{B} \wedge not\, \mathcal{C}.$$

We write **heads**(P) *for the set of all atoms occuring in rule heads in P.*

By pure *disjunctions we mean disjunctions consisting solely of positive or of negative literals.*

The set **heads**(P) captures with a simple syntactical condition atoms that are *possibly true*. This condition is used later to derive literals *not* A, namely if A is *not* contained in **heads**(P'), where P' is a transformed program obtained from P.

The general problem now is to define a *semantics* for the class of all disjunctive logic programs. A semantics usually is given by a set of intended models. An even more general notion is given by

Definition 2.2 (Operator $\mid\!\sim$, Semantics $\mathcal{S}_{\mid\sim}$) *By a semantic operator $\mid\!\sim$ we mean a binary relation between logic programs and pure disjunctions (i. e. disjunctions of the form $A_1 \vee \ldots \vee A_n$ or $\neg A_1 \vee \ldots \vee \neg A_n$) which satisfies the following three arguably obvious conditions:*

1. *If $P \mid\!\sim \psi$ and $\psi \subseteq \psi'$,then $P \mid\!\sim \psi'$* (Right Weakening).
2. *If $\mathcal{A} \leftarrow true \in P$ for a disjunction $\mathcal{A}$, then $P \mid\!\sim \mathcal{A}$* (Necessarily True).
3. *If $A \notin$* **heads**(P) *for any $\mathcal{L}$-atom A, then $P \mid\!\sim \neg A$* (Necessarily False).

Given such an operator $\mid\!\sim$ and a logic program P, by the semantics $\mathcal{S}_{\mid\sim}(P)$ of P determined by $\mid\!\sim$ we mean the set of all pure disjunctions derivable by $\mid\!\sim$ from P, i.e., $\mathcal{S}_{\mid\sim}(P) := \{\psi \mid P \mid\!\sim \psi\}$.

It will become clear later why we represent negative disjunctions by $\neg A_1 \vee \ldots \vee \neg A_n$ and not by $not\, A_1 \vee \ldots \vee not\, A_n$. Note that any of the known semantics in logic programming induces such a formal notion of semantics.

2.2 The Logic of Autoepistemic Beliefs

The language of AEB is a propositional modal language $\mathcal{L}_{AEB}$ obtained by augmenting the classical propositional language $\mathcal{L}$ with a modal operator $\mathbf{B}$, called the *belief* operator. The atomic formulae of the form $\mathbf{B}F$, where F is an arbitrary formula of $\mathcal{L}_{AEB}$, are called *belief atoms*. The formulae of $\mathcal{L}$ are called *objective*. Any theory T in the language $\mathcal{L}_{AEB}$ is called a *belief theory*. Observe that arbitrarily deep levels of *nested beliefs* are allowed in belief theories.

We assume the following two simple axiom schemata and one inference rule describing the arguably obvious properties of belief atoms:

(D) Consistency: $\neg\mathbf{B}\bot$

(K) Normality: $\mathbf{B}(F \rightarrow G) \rightarrow (\mathbf{B}F \rightarrow \mathbf{B}G)$ for any formulae F,G

(N) Necessitation Rule: $\frac{F}{\mathbf{B}F}$ for any formula F.

The first axiom states that tautologically false formulae are *not* believed. The second axiom states that if we believe that a formula F implies a formula G and if we believe that F is true then we believe that G is true as well. The necessitation inference rule states that if a formula F has been proven to be true then F is believed to be true.

Definition 2.3 (Derivability from $\mathcal{L}_{AEB}$) *For any belief theory T, we denote by $Cn_{AEB}(T)$ the smallest set of formulae of the language $\mathcal{L}_{AEB}$ which contains the theory T, all the (substitution instances of) the axioms (K) and (D) and is closed under both standard propositional consequence and the necessitation rule (N). We say that a formula F is* derivable *from theory T in the logic AEB if F belongs to $Cn_{AEB}(T)$. T is* consistent *if the theory $Cn_{AEB}(T)$ is consistent.*

In the presence of axiom (K), the axiom (D) is equivalent to the axiom

(D′) Restricted Consistency: $\mathbf{B}F \rightarrow \neg\mathbf{B}\neg F$ for any formula F.

stating that if we believe in a formula F then we do *not* believe in $\neg F$. Moreover, it is easy to see that the axioms imply that the belief operator $\mathbf{B}$ obeys the *distributive law for conjunctions* (see [Prz95]).

(C) Distributivity wrt Conjunctions: For any formulae F and G:

$$\mathbf{B}(F \wedge G) \equiv \mathbf{B}F \wedge \mathbf{B}G \quad (1)$$

The most important notion in the sequel is that of a *minimal* model:

Definition 2.4 (Min. Models in $\mathcal{L}_{AEB}$, [Prz95]) *By a* minimal model *of a belief theory T we mean a model M of T with the property that there is* no *smaller model N of T which coincides with M on belief atoms $\mathbf{B}F$. If a formula F is true in all minimal models of T then we write: $T \models_{\min} F$ and say that F is* minimally entailed[2] *by T.*

The intended meaning of belief atoms $\mathbf{B}F$ is based on *predicate minimization*:

$$\begin{array}{lcl} \mathbf{B}F & \equiv & F \text{ is minimally entailed} \\ & \equiv & F \text{ is true in all minimal models.} \end{array}$$

Accordingly, beliefs in AEB can be called *minimal beliefs* (see [Min82, GPP88] and [McC80]). As in Moore's Autoepistemic Logic, also in the Autoepistemic Logic of Beliefs the intended meaning of belief atoms is implemented by defining plausible sets of beliefs that an ideally rational and introspective agent may hold, given a set of premises T.

Definition 2.5 (Static AE Expansion, [Prz95]) *A belief theory $T^\diamond$ is called a* static autoepistemic expansion *of a belief theory T if it satisfies the following fixed-point equation:*

$$T^\diamond = Cn_{AEB}(T \cup \{\mathbf{B}F : T^\diamond \models_{\min} F\}),$$

where F ranges over all formulae of $\mathcal{L}_{AEB}$.

It turns out that every belief theory T in AEB has the *least* (in the sense of set-theoretic inclusion) static expansion $T^\diamond$ which has an *iterative* definition as the *least fixed point* of the monotonic[3] belief closure operator:

$$\Psi_T(S) = Cn_{AEB}(T \cup \{\mathbf{B}F : S \models_{\min} F\}),$$

where S is an arbitrary belief theory and the F's range over all formulae of $\mathcal{L}_{AEB}$.

3 The semantics D-WFS and STATIC

In this section we review briefly the semantics D-WFS and STATIC to make the paper self-contained.

3.1 D-WFS

The semantics D-WFS was originally defined ([BD94, BD98b]) as the *weakest* semantics $\mid\!\sim$ which is invariant under certain natural transformations $Trans : P \mapsto Trans(P)$ mapping logic programs into logic programs.

Theorem 3.1 (D-WFS, [BD98b]) *There exists the weakest semantics $\mathcal{S}_{\mid\sim}$ which satisfies the properties of Generalized Principle of Partial Evaluation (GPPE), Elimination of Tautologies (TAUT), Non-Minimal Rules (NMIN), Positive and Negative Reduction (RED^+, RED^-). Moreover, this semantics is consistent (i.e. it does not derive a literal A and its negation $\neg A$) and closed under logical consequences (as a set consisting of pure disjunctions). We call it the Disjunctive Wellfounded Semantics, or, briefly, D-WFS. It coincides with the wellfounded semantics WFS for normal and with GCWA for positive disjunctive programs.*

It remains to illustrate the properties mentioned.

TAUT: Delete a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,\mathcal{C}$ with $\mathcal{A} \cap \mathcal{B} \neq \emptyset$.

NMIN: Delete a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,\mathcal{C}$ if there is another rule $A' \leftarrow \mathcal{B}' \wedge not\,\mathcal{C}'$ with $\mathcal{A}' \subseteq \mathcal{A}$, $\mathcal{B}' \subseteq \mathcal{B}$, and $\mathcal{C}' \subseteq \mathcal{C}$.

$\mathbf{RED}^+$: Replace a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,\mathcal{C}$ by $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,(\mathcal{C} \cap heads(\Phi))$.

$\mathbf{RED}^-$: Delete a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,\mathcal{C}$, if there is a rule $(\mathcal{A}' \leftarrow)$ with $\mathcal{A}' \subseteq \mathcal{C}$.

GPPE: Replace a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge not\,\mathcal{C}$ where $\mathcal{B}$ contains a distinguished atom B by the k rules

$$\mathcal{A} \cup (A_i \setminus \{B\}) \leftarrow (\mathcal{B} \setminus \{B\}) \cup \mathcal{B}_i \wedge not\,(\mathcal{C} \cup \mathcal{C}_i)$$

[2] For readers familiar with *circumscription*, this means that we are considering predicate circumscription $CIRC(T; \mathcal{L})$ of the theory T in which atoms from the objective language are minimized while the belief atoms $\mathbf{B}F$ are fixed, i.e., $T \models_{\min} F \equiv CIRC(T; \mathcal{L}) \models F$. In other words, minimal models are obtained by first assigning *arbitrary* truth values to the belief atoms and then *minimizing* objective atoms. See also [YY93].

[3] Strictly speaking the operator is only monotone on the lattice of all theories of the form $T \cup M_{bel}$ where T is fixed and M_{bel} ranges over all sets of belief formulae.

where $\mathcal{A}_i \leftarrow \mathcal{B}_i \wedge not\, \mathcal{C}_i$ $(i = 1, \ldots, k)$ are *all* the rules with $B \in \mathcal{A}_i$.

The last rule has been introduced for disjunctive programs independently by Sakama/Seki and Brass/Dix in [SS94, BD94]. It allows us to get rid of positive body atoms. Given a program rule, we are free to select a specific positive occurrence of an atom B and then resolve this rule with all other rules that contain B in their heads (this set of rules constitutes the definition of B). Thus we replace the original program rule with a set of new rules. GPPE also covers the degenerate case when the atom B to be replaced does not appear in any head: then the whole rule is simply deleted.

In fact, we discovered later ([BD98a]) that our set of transformations is also *confluent* and terminating. This is of special importance, because given a program P, we can just apply our transformations in an arbitrary order and eventually reach a normal form res(P). From this normal form res(P), we can immediately read off the semantics as follows

$$P \mid\sim_{\text{D-WFS}} \psi \text{ iff } \begin{cases} \exists\, \mathcal{A} \subseteq \psi \text{ with } (\mathcal{A} \leftarrow) \in \text{res}(P) \text{ or} \\ \exists \neg A \in \psi \text{ and } A \notin \textbf{heads}(\text{res}(P)). \end{cases}$$

This is also the best method to really compute D-WFS for small examples and we will use it in the sequel. However in order to link D-WFS with a theorem prover and also with STATIC, we need a completely different characterization. This is a very technical characterization and had been stated in [BD98a]. It is similar to the Gelfond/Lifschitz translation and uses the notion of *reducing a program P with respect to a set of disjunctions Dis*:

Definition 3.2 (P/Dis)
Let P be a disjunctive logic program and let Dis be a set of pure disjunctions. Let P/Dis be the program obtained from P by doing the following reductions for all $not\, C$ and $C_1 \vee \ldots \vee C_k$

- *if $\neg C \in Dis$, then remove all occurrences of $not\, C$,*
- *if $C_1 \vee \ldots \vee C_k \in Dis$ then remove all rules that contain $\{not\, C_1, \ldots, not\, C_k\}$ in their bodies.*

P/Dis is obviously a slight generalization of the Gelfond-Lifschitz transformation. While the latter is defined relative to a set $N \subseteq At_{\mathcal{L}}$ in such a way that P/N is always positive*, our P/Dis still is a disjunctive program containing possibly the not-atoms. In fact, the GL-transform can be obtained from our transform by setting $P/N = P/Dis_N$ where $Dis_N := N \cup \{\neg X : X \in At_{\mathcal{L}} \setminus N\}$.*

Definition 3.3 (SEM_0)
Let P be a disjunctive logic program over $\mathcal{L}$. We define a set $Dis(P)$ of pure disjunctions as follows

$Dis(P) := Dis^+(P) \cup Dis^-(P)$ *where*

$Dis^+(P)$ *is the following set of disjunctions*

$$\{\, A_1 \vee \ldots \vee A_k \,:\, P/At_{\mathcal{L}} \models A_1 \vee \ldots \vee A_k \,\}$$

and $Dis^-(P)$ is the following set of negative literals

$$\{\, \neg A \,:\, P/Dis_N \models_{\min} \neg A \text{ for all } N \subseteq At_{\mathcal{L}} \text{ s. t. } N \models Dis^+(P) \}$$

Here, the series D_i defined by $D_{n+1} := Dis(P/D_n)$ grows monotonically and eventually gets constant. We define $SEM_0(P)$ to be the limit of this series.

The underlying idea in the last definition of a semantics SEM_0 is to use $P/At_{\mathcal{L}}$ (i.e. we just delete all rules containing negative literals) for deriving positive disjunctions and to use P/Dis_N (i.e. we set all negative literals outside N to true) for deriving negative literals. $\models_{\min}$ stands for the usual notion of *"truth in all minimal models"* where *minimal* is used classically. Note that since P/Dis_N is just a positive disjunctive program, this notion coincides with that introduced in Definition 2.4 because no belief-atoms occur.

Let us briefly comment on the occurrence of the set N in the $Dis^-(P)$ Definition 3.3. Suppose we have the program "$A \vee B \leftarrow$, $C \leftarrow not\, A \wedge not\, B$". Obviously of all our transformations only RED$^-$ can be applied and leads to the program "$A \vee B \leftarrow$" which constitutes the normal form. Therefore $not\, C$ is derivable in D-WFS, because it does not appear in $\textbf{heads}(\{A \vee B \leftarrow\})$.

Now if we remove the proviso "$N \models Dis^+(P)$" in the definition of $Dis^-(P)$, then we also have to consider $N_0 := \emptyset$, which implies that $\neg C$ would not be included. This proviso therefore guarantees that only those $N \subseteq At_{\mathcal{L}}$ that are compatible with $Dis^+(P)$ are considered.

Theorem 3.4 (D-WFS = SEM_0, ([BD98a]))
D-WFS is complete with respect to SEM_0:

$$\textit{D-WFS}(P) = SEM_0(P) \text{ for all } P.$$

3.2 STATIC

Here we show how the general framework introduced in Subsection 2.2 can be naturally used to define a semantics for disjunctive logic programs.

Of course, any disjunctive logic program in the sense of Definition 2.1 can be seen as a theory in $\mathcal{L}_{AEB}$ by

replacing *not* simply by $\neg$. Unfortunately, this would not give the right interpretation of the $\leftarrow$-arrow used in logic programming (there, usually, contrapositions are not allowed) and it would also not correspond to the various meanings of *negation-as-failure*.

But it turns out that a precise meaning of negation by default can be captured by appropriate translations of a disjunctive logic program into a belief theory. In our approach this meaning will be defined by

translating $not\, F$ into the formula $\mathbf{B}\neg F$ in AEB

thus providing $not\, F$ with the intended meaning of *"$\neg F$ is minimally entailed"* or *"F is false in all minimal models"*.

Definition 3.5 (Belief-Programs)
A belief-program is a (finite or infinite) set of clauses of the form:

$$A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \mathbf{B}\neg C_1 \wedge \ldots \wedge \mathbf{B}\neg C_n,$$

where A_i's, B_i's, and C_i's are $\mathcal{L}$-atoms and $k > 0$.

Therefore any disjunctive logic program P in the sense of Definition 2.1 can be translated into a special belief theory in $\mathcal{L}_{AEB}$ – the associated belief program. The meaning of *not* is given by $not\, F \equiv$ F is false in all minimal models $\equiv$ $\neg F$ is minimally entailed.

Thus STATIC can be seen as a semantics for disjunctive logic programs in the sense of Definition 2.2:

Definition 3.6 (STATIC)
Given a logic program P, we translate it into a belief program, compute the least static expansion and consider all formulae of the form $A_1 \vee \ldots \vee A_n$ or $\mathbf{B}\neg A_1 \vee \ldots \vee \mathbf{B}\neg A_m$ true in it. Their re-translations $A_1 \vee \ldots \vee A_n$ resp. $\neg A_1 \vee \ldots \vee \neg A_m$ constitute the set of derivable pure disjunctions: we call this set STATIC(P).

While arbitrary belief theories do not necessarily have consistent least static expansions, belief programs do. In fact, they can be characterized nicely as follows ([BDP96]):

Theorem 3.7 (Least Static Expansion in $\mathcal{L}_{AEB}$)
Every belief program P in AEB has the least *static expansion, which is consistent. It is obtained as the least fixed point $P^\diamond$ of the following monotonic belief closure operator Ψ_P, which assigns to a set S the closure*

$$Cn(P \cup \{\mathbf{B}(\neg p_1) \wedge \cdots \wedge \mathbf{B}(\neg p_n) \rightarrow \mathbf{B}(\neg q_1 \vee \ldots \vee \neg q_m) : S \models_{min} \neg p_1 \wedge \cdots \wedge \neg p_n \rightarrow (\neg q_1 \vee \ldots \vee \neg q_m)\}),$$

The sequence $\{P^\alpha\}$, defined by $P^0 := \emptyset$, $P^{\alpha+1} := \Psi_P(P^\alpha)$, $P^\gamma := \bigcup_{\beta \leq \gamma} \Psi_P(P^\beta)$, is monotonically increasing and has a unique fixed point $P^\diamond = P^\lambda = \Psi_P(P^\lambda)$, for some ordinal λ. For finite theories P the fixed point $P^\diamond$ is reached after finitely many steps.

4 A Restriction of STATIC and its equivalence to D-WFS

We start by considering an example illustrating a setting where STATIC is stronger than D-WFS.

Example 4.1 [STATIC is stronger than D-WFS] Let

$$\begin{array}{lll} P_1: & a \vee c & \leftarrow \; not\, d \\ & c & \leftarrow \; not\, c \\ & d & \leftarrow \; c \end{array}$$

Now *STATIC* concludes $\neg a$ but D-WFS does not. Let us explain why.

The normal form $\mathrm{res}(P_1)$ is obtained by applying GPPE to the third clause (replacing c). Thus $\mathrm{res}(P_1)$ consists of the first two clauses and the following two clauses $d \leftarrow not\, c$ and $d \vee a \leftarrow not\, d$. Consequently, by our characterization of D-WFS in terms of the normal form, all atoms are undefined.

For *STATIC* we have to consider the corresponding belief theory. Due to the (K)–axiom we have $\mathbf{B}\neg c \leftarrow \mathbf{B}\neg d$ in that belief theory. Together with the second clause we have $c \leftarrow \mathbf{B}\neg d$. Therefore, if we look at the first clause, a has no chance to be true in minimal models: $\mathbf{B}\neg d$ does not hold (because then both $\mathbf{B}c$ and $\mathbf{B}\neg c$ would hold). Therefore $\mathbf{B}\neg d$ holds, but then c must also hold and a need not to hold.

The last example shows that the (K)–Axiom in STATIC is powerful enough to use contrapositives and thus it is stronger than D-WFS. The idea to restrict STATIC is to replace (K) by a weaker axiom and also to work in a language where there are no nested beliefs.

4.1 STATIC in a restricted language $\mathcal{L}_{DBL}$

As for $\mathcal{L}_{AEB}$ we start with an objective language $\mathcal{L}$ and augment it by belief atoms of the form $\mathbf{B}F$ where F is a *positive* or *negative* formula from $\mathcal{L}$: this gives us the language $\mathcal{L}_{DBL}$. Thus, $\mathcal{L}_{DBL}$-formulae do not contain nested beliefs.

Definition 4.2 (Formulae Derivable in $\mathcal{L}_{DBL}$)
For any belief theory T, we denote by $Cn_{DBL}(T)$ the smallest set of formulae of the language $\mathcal{L}_{DBL}$ which contains the theory T, all the (substitution instances of) the axioms (C) and (D′) and is closed under both

standard propositional consequence and the necessitation rule (N). We say that a formula F is derivable *from theory T in the logic DBL if F belongs to $Cn_{DBL}(T)$. A belief theory T is* consistent *if the theory $Cn_{DBL}(T)$ is consistent.*

As before, we call a theory $T^\diamond$ a static autoepistemic expansion, if it satisfies

$$T^\diamond = Cn_{DBL}(T \cup \{\mathbf{B}F : T^\diamond \models_{\min} F\}).$$

Our next theorem shows that the least static expansion of a belief program in $\mathcal{L}_{DBL}$ is consistent and can be computed as follows:

Theorem 4.3 (Least Static Expansion in $\mathcal{L}_{DBL}$) *Every belief program P in DBL has the* least *static expansion, which is consistent. In addition, the set of pure disjunctions contained in this least expansion is already contained in the least fixed point $P^\diamond$ of the following monotonic belief closure operator Ψ_P, which assigns to a set S the closure*

$$Cn(P \cup \{\neg\mathbf{B}\neg q_1 \vee \ldots \vee \neg\mathbf{B}\neg q_k : S \models q_1 \vee \ldots \vee q_k\} \cup \{\mathbf{B}(\neg p_1 \vee \ldots \vee \neg p_n) : S \models_{\min} \neg p_1 \vee \ldots \vee \neg p_n\}),$$

where Cn denotes the consequences of propositional logic and the p_i, q_i are objective atoms. The sequence $\{P^\alpha\}$, defined by $P^0 := \emptyset$, $P^{\alpha+1} := \Psi_P(P^\alpha)$, $P^\gamma := \bigcup_{\beta \leq \gamma} \Psi_P(P^\beta)$, is monotonically increasing and has a unique fixed point $P^\diamond = P^\lambda = \Psi_P(P^\lambda)$, for some ordinal λ. For finite theories P the fixed point $P^\diamond$ is reached after finitely many steps.

Proof. (Sketch): It is easy to show that the operator is monotonic and thus has a least fixed-point $P^\diamond$. However, it is also immediate that this fixpoint is not the least static expansion, simply because not all necessary belief atoms are contained. Note that in the construction only the most simplified belief atoms were added: instead of adding $\mathbf{B}(q_1 \vee \ldots \vee q_k)$ the disjunction $\neg\mathbf{B}\neg q_1 \vee \ldots \vee \neg\mathbf{B}\neg q_k$ was added thereby applying axiom (C), (D′) and (N). It is important to note that the belief program P used in the construction just contains belief atoms of the most simple form: $\mathbf{B}(\neg q_1)$. This implies that the real least static expansion is obtained by "closing" $P^\diamond$ under the axioms (C), (D′) and (N). This only introduces new beliefs of the form $\mathbf{B}(\phi)$ where ϕ is a (positive or negative) formula such that $\mathbf{B}(\phi')$ (where ϕ' is a *subformula* of ϕ) is already present. Thus it has no effect on the set of pure disjunctions. □

The last theorem gives rise to a semantics STATIC_{DBL} in the same way as STATIC was introduced in Definition 3.6. This means that STATIC_{DBL}(P) is a certain set of pure disjunctions. In fact we have the following corollary which will be used later to establish the equivalence of D-WFS and STATIC_{DBL}:

Corollary 4.4 (STATIC_{DBL}) *The set of pure disjunctions contained in the least static expansion in $\mathcal{L}_{DBL}$ is already obtained as the least fixpoint of the following simplified closure operator $\Psi'_P(S)$*

$$Cn(P \cup \{\neg\mathbf{B}\neg q_1 \vee \ldots \vee \neg\mathbf{B}\neg q_k : S \models q_1 \vee \ldots \vee q_k\} \cup \{\mathbf{B}(\neg p_1) : S \models_{\min} \neg p_1\}),$$

Corollary 4.4 follows from Theorem 4.3 because the corresponding belief theory does not contain any axioms that relate belief atoms $\mathbf{B}(p \vee q)$ with $\mathbf{B}(q)$ and $\mathbf{B}(p)$! If this were the case, then the last characterization (which only uses belief atoms of the form $\mathbf{B}(\phi)$ where ϕ is a negated atom) would certainly be different from the one in Theorem 4.3, because there much more belief atoms (those where ϕ is a disjunction of negated atoms) are contained. Therefore the choice of the underlying axioms for $\mathcal{L}_{DBL}$ is very important.

4.2 STATIC_{DBL} coincides with D-WFS

We aim to reformulate the characterization of D-WFS given in Theorem 3.4 in order to bring it closer to minimal entailment $\models_{\min}$ of Definition 2.4 which is used in Lemma 4.4.

As we are considering disjunctive logic programs where we do not have belief atoms but $not\,(\phi)$ atoms, minimal entailment means that $not\,(\phi)$ atoms are fixed and all objective atoms are minimized. This is just to facilitate notation: fixing $not\,(\phi)$ atoms is completely equivalent to fixing $\mathbf{B}(\neg\phi)$ atoms.

To capture the notion of minimal entailment used in Definition 3.3 we strengthen the notion of models for a disjunctive logic program P by requiring that a model I (a set of atoms and formulae $not\,(\phi)$) of P has to be *NAF-consistent* for P, i.e., it must satisfy the following condition:

$$\text{If } P \models a_1 \vee \cdots \vee a_n, \text{ then } \exists i \in \{1, \ldots, n\}, not\,(a_i) \notin I.$$

For example, $\{A, C, not\,A, not\,B\}$ is not a NAF-consistent model of $P = \{A \vee B \leftarrow, C \leftarrow not\,A \wedge not\,B\}$ but $\{A, C, not\,A\}$ is. However, the latter is not a minimal model of P.

If the operator Dis employed in the characterization of Theorem 3.4 is modified in the following way, D-WFS is captured using normal minimal entailment but with

respect to NAF-consistent models. We denote this entailment relation by $\models_{\min(N)}$.

$$Dis_{\min}(P) = \begin{array}{ll} \{A_1 \vee \cdots \vee A_k & | P \models A_1 \vee \cdots \vee A_k\} \cup \\ \{\neg A & | P \models_{\min(N)} \neg A\} \end{array}$$

We denote by $\mathrm{SEM_c}(P)$ the semantics which is obtained by iterating this operator and propagating the results as in $\mathrm{SEM_0}$ in Theorem 3.4. This gives us exactly D-WFS.

Theorem 4.5 *$SEM_0(P) = \mathrm{SEM_c}(P)$.*

The theorem can be proved by establishing the following two propositions.

Proposition 4.6 *For any set of atoms $\{A_1, \cdots, A_k\}$,*

$$P/At_{\mathcal{L}} \models A_1 \vee \cdots \vee A_k \text{ iff } P \models A_1 \vee \cdots \vee A_k.$$

Proposition 4.7 *For any disjunctive logic program P and propositional formula ϕ,*

$$P/Dis_N \models_{\min} \phi \text{ for all } N \subseteq At_{\mathcal{L}} \text{ with } N \models Dis^+(P)$$
$$\text{iff}$$
$$P \models_{\min(N)} \phi.$$

Theorem 4.8 (STATIC_{DBL} = D-WFS) *$STATIC_{DBL}$ coincides with D-WFS.*

Proof. (Sketch): We have to compare the construction of D-WFS based on $\models_{\min(N)}$ and the construction of STATIC_{DBL} based on Corollary 4.4.

While in the first construction every step is split into deriving certain disjunctions and doing certain program transformations (to prepare the next step), the latter is given by applying an operator and getting larger and larger theories.

It is easy to see that the program transformations of the first construction are simulated in the latter method by the introduction of the belief atoms: new belief atoms restrict in the next round the set of minimal models used to derive negative disjunctions. To be more precise, the first transformation in Definition 3.2 is captured by adding $\mathbf{B}(\neg C)$ in the definition of the operator Ψ' in Corollary 4.4. The second transformation in Definition 3.2 is simulated by adding $\neg\mathbf{B}\neg C_1 \vee \ldots \vee \neg\mathbf{B}\neg C_k$. NAF-consistent models are nothing else than arbitrary minimal models (as used in the construction of Corollary 4.4) satisfying the additional constraints formalized by the belief atoms.

In fact, in the limit we get the same results. □

5 Towards an Implementation

The results established in the previous sections show interesting connections between minimal entailment and the D-WFS and STATIC semantics. They imply that D-WFS (and thus STATIC_{DBL}) can be implemented using iterative minimal model reasoning with a fairly standard notion of minimal entailment $\models_{\min(N)}$ (which is standard circumscription but over NAF-consistent models). In this section we demonstrate how this can be done by exploiting directly the new characterization of D-WFS in Theorem 4.5. Then we discuss how the required notion of minimal entailment $\models_{\min(N)}$ can be implemented using a new tableau method developed for circumscription [Nie96b, Nie96a, BFN96]. An interesting feature of the method is that it enables an implementation technique for D-WFS which works in *polynomial space*. This is important when dealing with larger programs.

The idea is that D-WFS is implemented as an iterative reduction on disjunctive logic programs. The reduction starts with the original program P and leads to a *reduced* program P^* with the property that every query can be answered from this program with one theorem prover call. This can be seen as a compilation (or a partial evaluation) of the program leading to a smaller program from which all queries can be answered.

The reduced program is obtained as a "fixed point" of a reduction operator $\mathrm{R_D}(\cdot)$ which is derived from the operator used in $\mathrm{SEM_c}$.

$$\begin{array}{rcl} \mathrm{R_D}(P) & = & \{\mathcal{A} \leftarrow \mathcal{B} \wedge not\, \mathcal{C}' \mid \mathcal{A} \leftarrow \mathcal{B} \wedge not\, \mathcal{C} \in P, \\ & & P/At_{\mathcal{L}} \not\models C_1 \vee \cdots \vee C_n \text{ where} \\ & & \mathcal{C} = \{C_1, \ldots, C_n\}, \\ & & \mathcal{C}' = \{C \in \mathcal{C} \mid P \not\models_{\min(N)} \neg C\}\} \end{array}$$

Now let P^* be the limit of the (monotonically decreasing) series of programs $P_0, P_1, \ldots$ where $P_0 = P$ and $P_{i+1} = \mathrm{R_D}(P_i)$. Hence, the reduced program P^* is obtained by

- removing each rule such that $not\, C_1, \ldots, not\, C_n$ are all the negative body literals and $C_1 \vee \cdots \vee C_n$ is classically entailed by those rules of the remaining program with no negative body literals and
- removing all $not\,(C)$ for which $\neg C$ is minimally entailed by the remaining program

until no reduction is possible. It can be established that $P_i = P/D_i$ for all $i = 0, 1, \ldots$ Then we have the following theorem.

Theorem 5.1 $\mathrm{SEM_c}(P) = Dis_{\min}(P^*)$.

The theorem combined with Proposition 4.6 implies that a positive query can be answered using a classical theorem prover and those rules of the reduced program not containing negative body literals, i.e., $A_1 \vee \cdots \vee A_k \in$ D-WFS(P) iff $P^*/At_{\mathcal{L}} \models A_1 \vee \cdots \vee A_k$ but for negative queries minimal model reasoning is needed, i.e., $\neg A \in$ D-WFS(P) iff $P^* \models_{\min(\mathrm{N})} \neg A$.

The series of reductions is straightforward to implement[4] if decision procedures for classical entailment and minimal entailment $\models_{\min(\mathrm{N})}$ are available.

decision procedure for classical entailment and minimal entailment $\models_{\min(\mathrm{N})}$ is available. The latter procedure can be implemented using a new tableau method for circumscription presented in [Nie96b, Nie96a]. Next we explain briefly the main ideas in the method.

Standard tableau methods deciding whether a query follows (classically) from a set of premises can be seen as systematic attempts to construct a counter-model, i.e., a model where the premises are true and the query false. Each open branch in the tableau presents a potential counter-model and a tableau proof is found if all branches can be closed indicating that there are no counter-models. In principle, it seems that tableau methods can be easily extended to handle minimal model reasoning by accepting as counter-models only minimal models of the premises. Technically this is more complicated because it is not straightforward to decide whether a model provided by an open branch is a minimal model of the premises without actually considering exponentially many other models. We illustrate the difficulties with the program P

$$\begin{array}{l} a_1 \vee b_1 \leftarrow, \ldots, a_n \vee b_n \leftarrow, \\ a_1 \leftarrow a_0, \ldots, a_n \leftarrow a_{n-1} \end{array} \qquad (2)$$

and the task of deciding whether $\neg a_0$ is minimally entailed by the program. A tableau method like that in [Nie96b, Nie96a] starts with the negation of the query and generates one counter-model $M = \{a_0, \ldots, a_n\}$. The problem is to decide whether this is a minimal model of P. A brute-force method would examine all $2^{n+1} - 1$ models that are smaller than M. In [Nie96b, Nie96a] an alternative approach is proposed by devising a novel characterization of minimal models which makes it possible to test whether a model is a minimal one independently of other models using one theorem prover call. The method can handle standard circumscription with fixed and varying predicates and for the notion of minimality in disjunctive programs it leads to the following characterization of minimal models. A model (a set of atoms and $not\,(\phi)$ atoms) of a program P is a minimal one iff

$$P \cup \mathrm{N}(M) \models A_1 \wedge \cdots \wedge A_n \qquad (3)$$

where $A_1, \ldots, A_n$ are the (ordinary) atoms in M and

$$\mathrm{N}(M) = \{\neg A \mid A \text{ is an (ordinary) atom}, A \notin M\} \cup \{not\, C \in M\}.$$

For example, given the program P (2) and the model $M = \{a_0, \ldots, a_n\}$, $\mathrm{N}(M) = \{\neg b_1, \ldots, \neg b_n\}$ and the minimality of M can be determined by verifying whether $P \cup \{\neg b_1, \ldots, \neg b_n\} \models a_0 \wedge \cdots \wedge a_n$ holds. It is a trivial task for an efficient classical theorem prover to decide that this is not the case implying that M is not a minimal model of P and that $\neg a_0$ is minimally entailed by P.

In order to extend the method from standard minimal model entailment to $\models_{\min(\mathrm{N})}$ the only change is to require that only NAF-consistent models are accepted. The test for NAF-consistency can be implemented using the following observation: a model is NAF-consistent for P iff

$$P \not\models C_1 \vee \cdots \vee C_m \qquad (4)$$

where $not\, C_1, \ldots, not\, C_m$ are all the $not\,(\phi)$ atoms in the model. Hence, a model of P is a NAF-consistent minimal model of P iff the conditions (3) and (4) hold for the model.

Example 5.2 Consider the program P

$$\begin{array}{lll} a \vee b & \leftarrow & \\ c & \leftarrow & not\, a \\ d & \leftarrow & not\, b \\ e & \leftarrow & c \wedge d \\ f & \leftarrow & not\, e \\ f & \leftarrow & not\, a \wedge not\, b \wedge not\, c \end{array}$$

and models $M_1 = \{a, not\, b, c, d, e\}$ and $M_2 = \{a, not\, a, not\, b, c, d, e\}$. For M_1 the condition (3) fails as $\mathrm{N}(M_1) = \{\neg b, \neg f, not\, b\}$ and, e.g., $P \cup \mathrm{N}(M_1) \not\models c$. For M_2, $\mathrm{N}(M_2) = \{\neg b, \neg f, not\, a, not\, b\}$ and the condition (3) holds but (4) does not as $P \models a \vee b$. In fact, there are no NAF-consistent minimal models of P where e is true and, hence, $P \models_{\min(\mathrm{N})} \neg e$. Thus when deriving the reduced program P^*, $not\, e$ is removed. Similarly, the last rule is removed since $a \vee b \vee c$ is classically entailed by the rules without negative literals.

As mentioned in the beginning of the section, the technique outlined above enables a *polynomial space* implementation of D-WFS . This is because both classical entailment and minimal model entailment $\models_{\min(\mathrm{N})}$ can

[4] See <URL:http://www.uni-koblenz.de/ag-ki/DLP/systems/dwfs-mm.html>.

be implemented in polynomial space. For the latter a polynomial space bound can be realized by developing the tableau one branch at a time and performing the minimality and NAF-consistency tests using a polynomial space classical theorem prover. When the two entailment relations can be decided in polynomial space, similar space bounds clearly hold for computing the reduced program and for query-evaluation, as well.

6 Related Work

Meanwhile some systems for disjunctive logic programming semantics exist or are in development. Besides systems for computing the classical stable semantics ([GL88, GL90]), D. Seipel ([Sei94]) implemented a system (<URL:http://www-info1.informatik.uni-wuerzburg.de/database/DisLog/introduction.html>) for computing various variants of the disjunctive semantics contained in [LMR92].

A group in Vienna (T. Eiter, G. Gottlob, N. Leone) is currently building a system called `dlv` [ELM+97]. This is a knowledge representation system, based on disjunctive logic programming, which offers front-ends to several advanced KR formalisms and implements the disjunctive stable semantics. The system runs in *polynomial space* and *single exponential time*, and is able to efficiently recognize and process syntactical subclasses of disjunctive logic programs which have lower computational complexity than the general case (like, e.g., programs with head-cycle free disjunction or stratified negation).

Implementation of D-WFS and its extensions is one of the main goals of the *DisLoP project*, undertaken by the Artificial Intelligence Research Group at the University of Koblenz (J. Dix and U. Furbach) ([ADN97a, ADN97b], see also <http://www.uni-koblenz.de/ag-ki/DLP/>). In contrast to the other approaches just mentioned, it aims at extending certain theorem-proving calculi (*restart model elimination* and *hyper tableau calculi*) for disjunctive logic programming.

Finally, we refer to [DFN97], which contains a collection of nonmonotonic system descriptions (including some handling disjunction) and to the following webpage which is actively maintained and contains information on various logic programming systems that concentrate on non-monotonic aspects (different kinds of negation, disjunction, abduction etc.): <URL:http://www.uni-koblenz.de/ag-ki/LP/>.

7 Conclusion

While the semantics of non-disjunctive programs is fairly well understood and there exist several implementations of the best known semantics (wellfounded semantics and stable semantics), the declarative and computational foundations of disjunctive programming proved to be much more elusive and difficult. However, the recent introduction of the two new, promising semantics for the class of disjunctive logic programs, namely, the *static semantics* proposed by Przymusinski [Prz95] and the *D-WFS semantics* proposed by Brass and Dix [BD94, BD98b], seems to significantly enhance our understanding of the theoretical and algorithmic aspects of disjunctive logic programs.

Both semantics extend the wellfounded semantics of normal programs and have very natural and regular features analogous to those shared by the wellfounded semantics. Although the two semantics are based on very different ideas, they turn out to be closely related as we have just shown. We note that also the disjunctive stable semantics can be captured in our framework (this has been shown in [BD97] for a specific set of transformation rules).

In this paper we conducted a detailed study of the relationship between the static and the D-WFS semantics in the class of *disjunctive logic programs*. We proved an equivalence result showing how to restrict STATIC to get D-WFS.

Subsequently, we demonstrated the applicability and importance of the above theorems by using them to devise an implementation technique building on a novel method for circumscriptive reasoning. An implementation based on this technique has been developed in the DisLoP-project[5] at the university of Koblenz.

Acknowledgments

We thank Alexander Bochman and several referees for useful comments on parts of the paper.

References

[ADN97a] Chandrabose Aravindan, Jürgen Dix, and Ilkka Niemelä. Dislop: A research project on disjunctive logic programming. *AI Communications*, 10:151–165, 1997.

[ADN97b] Chandrabose Aravindan, Jürgen Dix, and Ilkka Niemelä. DisLoP: Towards a Disjunctive Logic Programming System. In J. Dix,

[5] See <URL:http://www.uni-koblenz.de/ag-ki/DLP/>.

U. Furbach, and A. Nerode, editors, *Logic Programming and Non-Monotonic Reasoning, Proceedings of the Fourth International Conference*, LNAI 1265, pages 342–353, Berlin, June 1997. Springer.

[BD94] Stefan Brass and Jürgen Dix. A disjunctive semantics based on unfolding and bottom-up evaluation. In Bernd Wolfinger, editor, *Innovationen bei Rechen- und Kommunikationssystemen,* (IFIP '94-Congress, Workshop FG2: Disjunctive Logic Programming and Disjunctive Databases), pages 83–91, Berlin, 1994. Springer.

[BD97] Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. *Journal of Logic Programming*, 32(3):207–228, 1997. (Extended abstract appeared in: Characterizations of the Stable Semantics by Partial Evaluation *LPNMR, Proceedings of the Third International Conference, Kentucky*, pages 85–98, 1995. LNCS 928, Springer.).

[BD98a] Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Well-founded Semantics: Confluent Calculi and Iterated GCWA. *Journal of Automated Reasoning*, 20(1):143–165, 1998. (Extended abstract appeared in: Characterizing D-WFS: Confluence and Iterated GCWA. *Logics in Artificial Intelligence, JELIA '96*, pages 268–283, 1996. Springer, LNCS 1126.).

[BD98b] Stefan Brass and Jürgen Dix. Semantics of Disjunctive Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, accepted for publication, 1998. (Extended abstract appeared in: Disjunctive Semantics Based upon Partial and Bottom-Up Evaluation, *Proceedings of the 12-th International Logic Programming Conference, Tokyo*, pages 199–213, 1995. MIT Press.).

[BDP96] Stefan Brass, Jürgen Dix, and Teodor. C. Przymusinski. Super Logic Programs. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, pages 529–541. San Francisco, CA, Morgan Kaufmann, 1996.

[BFN96] Peter Baumgartner, Ulrich Furbach, and Ilkka Niemelä. Hyper tableaux. In L.M. Pereira J.J. Alferes and E. Orlowska, editors, *Logics in Artificial Intelligence (JELIA '96)*, LNCS 1126, pages 1–17. Springer, 1996.

[BL93] Stefan Brass and Udo W. Lipeck. Bottom-up query evaluation with partially ordered defaults. In Stefano Ceri, Katsumi Tanaka, and Shalom Tsur, editors, *Deductive and Object-Oriented Databases, Third Int. Conf., (DOOD'93)*, number 760 in LNCS, pages 253–266, Berlin, 1993. Springer.

[DFN97] J. Dix, U. Furbach, and A. Nerode, editors. *Logic Programming and Nonmonotonic Reasoning*, LNAI 1265, Berlin, 1997. Springer.

[Dix95] Jürgen Dix. Semantics of Logic Programs: Their Intuitions and Formal Properties. An Overview. In Andre Fuhrmann and Hans Rott, editors, *Logic, Action and Information – Essays on Logic in Philosophy and Artificial Intelligence*, pages 241–327. DeGruyter, 1995.

[EG93] Thomas Eiter and Georg Gottlob. Complexity aspects of various semantics for disjunctive databases. In *Proc. of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'93)*, pages 158–167, 1993.

[EG96] Thomas Eiter and Georg Gottlob. Mächtigkeit von Logikprogrammierung über Datenbanken. *KI*, 3:32–39, 1996.

[EGM94] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Adding disjunction to datalog. In *Proc. of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'94)*, pages 267–278, 1994.

[ELM+97] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A Deductive System for Non-Monotonic Reasoning. In J. Dix, U. Furbach, and A. Nerode, editors, *Logic Programming and Non-Monotonic Reasoning, Proceedings of the Fourth International Conference*, LNAI 1265, pages 363–374, Berlin, June 1997. Springer.

[GL88] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen,

editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[GL90] Michael Gelfond and Vladimir Lifschitz. Logic Program with Classical Negation. In David H.D. Warren and Peter Szeredi, editors, *Proceedings of the 7th Int. Conf. on Logic Programming*, pages 579–597. MIT, June 1990.

[GPP88] Michael Gelfond, Halina Przymusinska, and Teodor Przymusinski. The extended Closed World Assumption and its Relationship to parallel Circumscription. In *Proceedings 7th Symposion on Principles of Database Systems*, 1988.

[Lip79] W. Lipski, Jr. On semantic issues connected with incomplete information databases. *ACM Transactions on Database Systems*, 4:262–296, 1979.

[LMR92] Jorge Lobo, Jack Minker, and Arcot Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT-Press, 1992.

[McC80] John McCarthy. Circumscription: A Form of Nonmonotonic Reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[Min82] Jack Minker. On indefinite databases and the closed world assumption. In *Proceedings of the 6th Conference on Automated Deduction, New York*, pages 292–308, Berlin, 1982. Springer.

[Min96] Jack Minker. Logic and databases: A 20 year retrospective. In Dino Pedreschi and Carlo Zaniolo, editors, *Proceedings of the International Workshop on Logic in Databases (LID)*, LNCS 1154, pages 3–58. Springer, Berlin, 1996.

[Nie96a] Ilkka Niemelä. Implementing circumscription using a tableau method. In W. Wahlster, editor, *Proceedings of the European Conference on Artificial Intelligence*, pages 80–84, Budapest, Hungary, August 1996. John Wiley.

[Nie96b] Ilkka Niemelä. A tableau calculus for minimal model reasoning. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 278–294, Terrasini, Italy, May 1996. LNAI 1071, Springer-Verlag.

[Prz95] Teodor Przymusinski. Static Semantics For Normal and Disjunctive Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 14:323–357, 1995.

[Prz98] Teodor C. Przymusinski. Autoepistemic logic of knowledge and beliefs. *Artificial Intelligence*, 95:115–154, 1998.

[Sei94] Dietmar Seipel. An efficient computation of the extended generalized closed world assumption by support-for-negation sets. In *Proc. Int. Conf. on Logic Programming and Automated Reasoning (LPAR'94)*, number 822 in LNAI, pages 245–259, Berlin, 1994. Springer.

[SS94] Chiaki Sakama and Hirohisa Seki. Partial Deduction of Disjunctive Logic Programs: A Declarative Approach. In *Logic Program Synthesis and Transformation – Meta Programming in Logic*, LNCS 883, pages 170–182, Berlin, 1994. Springer.

[YY93] Jia-Huai You and Li-Yan Yuan. Autoepistemic Circumscription and Logic Programming. *Journal of Automated Reasoning*, 10:143–160, 1993.