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

Stefan Brass[1] and Jürgen Dix[2]
and Ilkka Niemelä[2] and Teodor C. Przymusinski[3]**

[1] Univ. Hannover, Inst. f. Informatik, Lange Laube 22, D-30159 Hannover, Germany
[2] Univ. Koblenz, Inst. f. Informatik, Rheinau 1, D-56075 Koblenz, Germany
[3] Dept. of Computer Science, University of California, Riverside, CA 92521, USA

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. Quite recently, two new and very promising semantics have been proposed for the class of disjunctive logic programs. Both of them extend the well-founded semantics of normal programs. The first one is the *static semantics* proposed by Przymusinski and the other is the *D-WFS 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 efficient implementation based on a prover for computing minimal models for positive disjunctive programs.

Keywords: Non-Monotonic Reasoning, Negation-as-Failure,
Disjunctive Logic Programming and Semantics.

1 Introduction

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*. That is why considerable interest and research effort has been recently 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 research effort. 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 significantly simplifies the problem of *translation* of non-monotonic formalisms into logic programs, and, consequently, facilitates using logic programming as an

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

inference engine for non-monotonic reasoning. Finally, 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 well-founded 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 (AELB)* 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 of the static semantics and of D-WFS. In spite of the fact that the two semantics attempt to provide meaning to disjunctive programs based on very different ideas, they turn out to be quite closely related. For example it is fairly easy to show that STATIC also satisfies the abstract properties on which D-WFS is based: it is therefore stronger than D-WFS. We give an example where STATIC is strictly stronger than D-WFS — the reason is the (K)–Axiom and the possibility of having arbitrarily nested beliefs. But it turns out that it is possible to restrict the underlying language of STATIC (so that only non-nested beliefs are present) and to define a new version of STATIC in this language: STATIC_{DBL}. Our main theorem gives us a nice characterization of STATIC_{DBL}. Using this result, we are able to prove our equivalence result, namely that this version of STATIC coincides with D-WFS:

$$\mathrm{STATIC}_{DBL}(P) = \text{D-WFS}(P).$$

Another important feature of a semantics is its computational behaviour. Recently one of the authors introduced a very efficient method to compute minimal models (circumscription with fixed predicates). This approach allows for the computation of hundreds of thousands of minimal models. In Section 5, as our second main result, we show how this method can be nicely adapted for the computation of D-WFS and STATIC.

Our results are obtained by using the characterization of least static expansions in [BDP96] and a characterization of D-WFS contained in [BD96b].

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* (AELB). In Section 3

[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,Min93].

we recapitulate shortly 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].

2 Preliminaries

We introduce in this section the neccessary 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 autoepistemic logic of beliefs.

2.1 Disjunctive Logic Programs and Semantics

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

Definition 1 (Logic Program P, Possibly True Facts $\mathcal{P}(P)$).
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 $\mathcal{A} \leftarrow \mathcal{B}ody$. We write $\mathcal{P}(P)$ for the set of all atoms occuring in rule heads in P: these are atoms that are *possibly true*. By *pure* disjunctions we mean disjunctions consisting solely of positive or of negative literals.

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 (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. *Right Weakening:* If $P \mid\!\sim \psi$ and $\psi \subseteq \psi'$[2], then $P \mid\!\sim \psi'$.

[2] I. e. ψ is a subdisjunction of ψ'.

2. *Necessarily True:* If $\mathcal{A} \leftarrow true \in P$ for a disjunction $\mathcal{A}$, then $P \mid\!\sim \mathcal{A}$.
3. *Necessarily False:* If $A \notin \mathcal{P}(P)$ for some $\mathcal{L}$-ground atom A, then $P \mid\!\sim \neg A$.

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 $AELB$ is a propositional modal language $\mathcal{L}_{AELB}$ obtained by augmenting the classical propositional language with a modal operator $\mathcal{B}_{el}$, called the *belief* operator. The atomic formulae of the form $\mathcal{B}_{el}F$, where F is an arbitrary formula of $\mathcal{L}_{AELB}$, are called *belief atoms*. The formulae of the original language are called *objective*. Any theory T in the language $\mathcal{L}_{AELB}$ is called a *belief theory*. Observe that arbitrarily deep level of *nested beliefs* is 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 Axiom: $\neg\mathcal{B}_{el}\bot$
(K) Normality Axiom: $\mathcal{B}_{el}(F \to G) \ \to\ (\mathcal{B}_{el}F \to \mathcal{B}_{el}G)$ for formulae F,G
(N) Necessitation Rule: $\frac{F}{\mathcal{B}_{el}F}$ 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 3 (Formulae Derivable from a Belief Theory $\mathcal{L}_{AELB}$).
For any belief theory T, we denote by $Cn_{AELB}(T)$ the smallest set of formulae of the language $\mathcal{L}_{AELB}$ 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 $AELB$ if F belongs to $Cn_{AELB}(T)$. T is *consistent* if the theory $Cn_{AELB}(T)$ is consistent.

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

(D') Restricted Consistency: $\mathcal{B}_{el}F \to \neg\mathcal{B}_{el}\neg F$ 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 $\mathcal{B}_{el}$ obeys the *distributive law for conjunctions* (see [Prz95]).

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

$$\mathcal{B}_{el}(F \wedge G) \;\equiv\; \mathcal{B}_{el}F \wedge \mathcal{B}_{el}G \tag{1}$$

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

Definition 4 (Minimal Models in $\mathcal{L}_{AELB}$, [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 $\mathcal{B}_{el}F$. 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*[3] by T.

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

$$\mathcal{B}_{el}F \;\equiv\; F \text{ is minimally entailed } \;\equiv\; F \quad \text{is true in all minimal models.}$$

Accordingly, beliefs in $AELB$ 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 5 (Static Autoepistemic 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:

$$T^{\diamond} = Cn_{AELB}(T \;\cup \{\mathcal{B}_{el}F : T^{\diamond} \models_{\min} F\}),$$

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

It turns out that every belief theory T in $AELB$ 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[4] belief closure operator:

$$\Psi_T(S) = Cn_{AELB}(T \cup \{\mathcal{B}_{el}F : S \;\models_{\min} F\}),$$

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

[3] For readers familiar with *circumscription*, this means that we are considering predicate circumscription $CIRC(T;\mathcal{K})$ of the theory T in which atoms from the objective language are minimized while the belief atoms $\mathcal{B}_{el}F$ 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].

[4] 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.

3 The semantics D-WFS and STATIC

In this section we review very shortly the semantics D-WFS and STATIC to make the paper selfcontained.

3.1 D-WFS

The semantics D-WFS was originally defined ([BD94,BD97]) as the *weakest* semantics $\mid\!\sim$ which is invariant under certain natural transformations $\mathit{Trans} : P \mapsto \mathit{Trans}(P)$ mapping logic programs into logic programs. In fact, we discovered later ([BD96b]) that our set of transformations is also *confluent*.

For our purposes here we do not need to repeat the original definition. We just cite an equivalent characterization of D-WFS from [BD96b]. This characterization 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 6 (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 ocurrences 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 *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 7 ($\mathbf{SEM}_0$).
Let P be a disjunctive logic program over $\mathcal{L}$. We define a set $Dis(P)$ of pure disjunctions as follows

$$Dis(P) := \quad Dis^+(P) \ \cup Dis^-(P) \text{ where}$$

$$\begin{array}{lll} Dis^+(P) := \{\ A_1 \vee \ldots \vee A_k : P/At_{\mathcal{L}} & \models & A_1 \vee \ldots \vee A_k \qquad\qquad \} \\ Dis^-(P) := \{\ \neg A \qquad\qquad\quad : P/Dis_N & \models_{min} & \neg A \text{ for all } N \subseteq At_{\mathcal{L}} \\ & & \qquad \text{s. t.}^5 N \models Dis^+(P)\ \} \end{array}$$

Here, the series D_i defined by $D_{n+1} := Dis(P/D_n)$ grows monotonically and eventually gets constant. We define $\mathrm{SEM}_0(P)$ to be the limit of this series.

[5] We need to consider only those $N \subseteq At_{\mathcal{L}}$ that are compatible with $Dis^+(P)$. For "$A \vee B,\ C \leftarrow not\, A, not\, B$" we do not want to consider $N_0 := \emptyset$ because then $\neg C$ would not be derivable.

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 4 because no belief-atoms occur.

Theorem 8 (D-WFS = $\mathbf{SEM}_0$, ([BD96b])).
D-WFS is complete with respect to SEM_0: D-WFS(P) = $SEM_0(P)$ 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 1 can be seen as a theory in $\mathcal{L}_{AELB}$ 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 its meaning will be defined by

$$\text{translating } \textit{not}\, F \text{ into the formula } \mathcal{B}_{el}\neg F \text{ in } AELB$$

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

Definition 9 (Belief-Programs).
A Belief-program is a (finite or infinite) set of clauses of the form:

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

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 1 can be translated into a special belief theory in $\mathcal{L}_{AELB}$ – 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.

Therefore STATIC can be seen as a semantics for disjunctive logic programs in the sense of Definition 2:

Definition 10 (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 $\mathcal{B}_{el}\neg A_1 \vee \ldots \vee \mathcal{B}_{el}\neg A_m$ 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 quite nicely as follows:

Theorem 11 (Least Static Expansion in $\mathcal{L}_{AELB}$, [BDP96]).
Every belief program P in AELB 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 Φ_P:*

$$\begin{aligned}\Phi_P(S) := Cn(P \cup \{&\mathcal{B}(\neg p_1) \wedge \cdots \wedge \mathcal{B}(\neg p_n) \rightarrow \mathcal{B}(\neg q_1 \vee \ldots \vee \neg q_m) \ : \\ &S \models_{min} \neg p_1 \wedge \cdots \wedge \neg p_n \rightarrow (\neg q_1 \vee \ldots \vee \neg q_m)\}),\end{aligned}$$

The sequence $\{P^\alpha\}$ 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 12 (STATIC is stronger than D-WFS).
Let

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

Now *STATIC* concludes $\neg a$ but D-WFS does not. The difference is the (K)–axiom which implies $not\, c \leftarrow not\, d$ in the corresponding belief theory.

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 $\mathcal{L}_{AELB}$ we start with an objective language $\mathcal{L}$ and augment it by belief atoms of the form $\mathcal{B}_{el}F$ 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 13 (Formulae Derivable from a Belief Theory 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

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

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

Theorem 14 (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 Φ_P:*

$$\begin{aligned}\Phi_P(S) := Cn(P \cup \{ &\neg\mathcal{B}_{el}\neg q_1 \vee \ldots \vee \neg\mathcal{B}_{el}\neg q_k : S \models \quad q_1 \vee \ldots \vee q_k \quad \} \\ \cup \{ &\mathcal{B}_{el}(\neg p_1 \vee \ldots \vee \neg p_n) \qquad : S \models_{min} \neg p_1 \vee \ldots \vee \neg p_n \}),\end{aligned}$$

where Cn denotes the consequences of propositional logic and the p_i, q_i are objective atoms. The sequence $\{P^\alpha\}$ is monotonically increasing and has a unique fixed point $P^\diamond = P^\lambda = \Phi_P(P^\lambda)$, 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 last static expansion, simply because not all neccessary belief atoms are contained. Note that in the construction only the most simplified belief atoms were added: instead of adding $\mathcal{B}_{el}(q_1 \vee \ldots \vee q_k)$ the disjunction $\neg\mathcal{B}_{el}\neg q_1 \vee \ldots \vee \neg\mathcal{B}_{el}\neg q_k$ 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 belef atoms of the most simple form: $\mathcal{B}_{el}(\neg q_1)$. 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 $\mathcal{B}_{el}(\phi)$ where ϕ is a (positive or negative) formula such that $\mathcal{B}_{el}(\phi')$ (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 10. This means that STATIC_{DBL}(P) is a certain set of pure disjunctions. It is very important that the original axiom (K) is *not* present — this fact implies that belief atoms $\mathcal{B}_{el}(p \vee q)$ are no more related to $\mathcal{B}_{el}(q)$ and $\mathcal{B}_{el}(p)$! In particular, we have the following corollary which will be used later to establish the equivalence of D-WFS and STATIC_{DBL}:

Corollary 15 (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

$$\begin{aligned}\Phi'_P(S) := Cn(P \cup \{ &\neg\mathcal{B}_{el}\neg q_1 \vee \ldots \vee \neg\mathcal{B}_{el}\neg q_k : S \models \quad q_1 \vee \ldots \vee q_k \} \\ \cup \{ &\mathcal{B}_{el}(\neg p_1) \qquad : S \models_{min} \neg p_1 \qquad \}),\end{aligned}$$

4.2 STATIC$_{DBL}$ coincides with D-WFS

We aim to reformulate the characterization of D-WFS given in Theorem 8 in order to bring it closer to minimal entailment $\models_{\min}$ introduced in Definition 4 which is used in Corollary 15.

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 $\mathcal{B}_{el}(\neg\phi)$ atoms.

In order to capture the notion of minimal entailment used in Definition 7 we must 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:

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

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 8 is modified in the way shown below, D-WFS is captured using normal minimal entailment but with respect to NAF-consistent models. We denote this entailment relation by $\models_{\min(\mathrm{N})}$.

We denote by $\mathrm{SEM_c}(P)$ the semantics which is obtained by iterating the operator

$$\begin{aligned} Dis_{\min}(P) := \{A_1 \vee \cdots \vee A_k : P &\models A_1 \vee \cdots \vee A_k\} \cup \\ \{\neg A : \quad P &\models_{\min(\mathrm{N})} \neg A\} \end{aligned}$$

and propagating the results as in SEM_0 in Definition 7, i.e., $\mathrm{SEM_c}(P)$ is the limit of the series, $D_0, D_1, \ldots$ where $D_0 = \emptyset$ and $D_{n+1} := Dis_{\min}(P/D_n)$. This gives us exactly D-WFS.

Theorem 16. *$SEM_0(P) = \mathrm{SEM_c}(P)$.*

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

Proposition 17. *For any set of atoms $\{A_1, \cdots, A_k\}$,*

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

Proof. ($\Rightarrow$) Assume $P \not\models A_1 \vee \cdots \vee A_k$. Then there is a model M such that $M \models P \cup \{\neg(A_1 \vee \cdots \vee A_k)\}$. Since $P/At_{\mathcal{L}} \subseteq P$, $P/At_{\mathcal{L}} \not\models A_1 \vee \cdots \vee A_k$ holds.

($\Leftarrow$) Assume $P/At_{\mathcal{L}} \not\models A_1 \vee \cdots \vee A_k$. Then there is a model M in which $P/At_{\mathcal{L}}$ is true and $A_1 \vee \cdots \vee A_k$ is false. Choose a minimal model M' of $P/At_{\mathcal{L}}$ for which $M' \subseteq M$. Now $M' \not\models A_1 \vee \cdots \vee A_k$. M' is a model of P because all belief atoms are false in M' and if $\mathcal{A} \leftarrow \mathcal{B} \in P$, then $\mathcal{A} \leftarrow \mathcal{B} \in P/At_{\mathcal{L}}$. Hence M' is an minimal model of P and $P \not\models_{\min} A_1 \vee \cdots \vee A_k$ holds. Hence, $P \not\models A_1 \vee \cdots \vee A_k$ holds. $\square$

Note that the proof shows that for any set of atoms $\{A_1, \cdots, A_k\}$,

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

holds as well.

Proposition 18. *For any disjunctive logic program P and propositional formula ϕ,*

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

Proof. ($\Rightarrow$) Assume $P \not\models_{\min(\mathrm{N})} \phi$. Then there is a NAF-consistent minimal model M of P such that $M \not\models \phi$. Let $N = \{C \in At_{\mathcal{L}} \mid M \not\models not\, C\}$ and let $M' = M \cap At_{\mathcal{L}}$. Then $M' \not\models \phi$ and it is easy to establish by Proposition 17 that NAF-consistency of M implies $N \models Dis^+(P)$. We show that (i) $M' \models P/Dis_N$ and (ii) M' is an minimal model of P/Dis_N. This implies that $P/Dis_N \not\models_{\min} \phi$ holds.

(i) If $\mathcal{A} \leftarrow \mathcal{B} \in P/Dis_N$, then $\mathcal{A} \leftarrow \mathcal{B} \wedge not\, \mathcal{C} \in P$ and $\mathcal{C} \cap N = \emptyset$. Then $M \models not\, \mathcal{C}$. If $M' \models \mathcal{B}$, then $M \models \mathcal{B}$, and hence $M \models \mathcal{A}$ because $M \models \mathcal{A} \leftarrow \mathcal{B} \wedge not\, \mathcal{C}$. But then $M' \models \mathcal{A}$. Hence, $M' \models P/Dis_N$.

(ii) Assume that there is a model M'' of P/Dis_N for which $M'' \subset M'$. Now it is straightforward to show that $M''' = M'' \cup \{not\, \psi \mid M \models not\, \psi\}$ is a model of P. If $\mathcal{A} \leftarrow \mathcal{B} \wedge not\, \mathcal{C} \in P$ and $M''' \models \mathcal{B} \wedge not\, \mathcal{C}$, then $M \models not\, \mathcal{C}$ and, hence, $\mathcal{A} \leftarrow \mathcal{B} \in P/Dis_N$. Since $M'' \models \mathcal{B}$, $M'' \models \mathcal{A}$ holds and thus $M''' \models \mathcal{A}$. Hence, $M''' \models P$. This contradicts the assumption that M is an minimal model of P because $M''' \cap At_{\mathcal{L}} = M'' \subset M' = M \cap At_{\mathcal{L}}$ holds.

($\Leftarrow$) Assume $P/Dis_N \not\models_{\min} \phi$ for some $N \subseteq At_{\mathcal{L}}$ such that $N \models Dis^+(P)$. Then there is a minimal model M of P/Dis_N such that $M \not\models \neg\phi$. Let $M' = M \cup \{not\, C \mid C \in At_{\mathcal{L}} - N\}$. Then $M' \not\models \phi$ and it is straightforward to establish by Proposition 17 that M' is NAF-consistent. We show that (i) $M' \models P$ and (ii) M' is a minimal model of P. This implies that $P \not\models_{\min(\mathrm{N})} \phi$ holds.

(i) If $\mathcal{A} \leftarrow \mathcal{B} \wedge not\, \mathcal{C} \in P$ and $M' \models \mathcal{B} \wedge not\, \mathcal{C}$, then $\mathcal{C} \cap N = \emptyset$ and $\mathcal{A} \leftarrow \mathcal{B} \in P/Dis_N$. Since $M \models \mathcal{B}$, $M \models \mathcal{A}$ because $M \models \mathcal{A} \leftarrow \mathcal{B}$. But then $M' \models \mathcal{A}$. Thus, $M' \models P$.

(ii) Assume that there is a model M'' of P for which $M'' \cap At_{\mathcal{L}} \subset M' \cap At_{\mathcal{L}}$ but $M'' \cap \mathrm{At}_{\mathrm{not}} = M' \cap \mathrm{At}_{\mathrm{not}}$. Now it is straightforward to show that $M''' = M'' \cap At_{\mathcal{L}}$ is a model of P/Dis_N. If $\mathcal{A} \leftarrow \mathcal{B} \in P/Dis_N$, then $\mathcal{C} \cap N = \emptyset$ and $M'' \models not\, \mathcal{C}$. If $M''' \models \mathcal{B}$, then $M'' \models \mathcal{B}$ and, hence, then $M'' \models \mathcal{A}$. But this implies $M''' \models \mathcal{A}$. Hence, $M''' \models P/Dis_N$. This contradicts the assumption that M is an minimal model of P/Dis_N because $M''' \cap At_{\mathcal{L}} = M'' \cap At_{\mathcal{L}} \subset M' \cap At_{\mathcal{L}} = M \cap At_{\mathcal{L}}$. □

Theorem 19 (STATIC$_{DBL}$ = D-WFS).
STATIC$_{DBL}$ coincides with D-WFS.

Proof. (Sketch): We have to compare the construction of D-WFS based on $\models_{\min(\mathrm{N})}$ and the construction of STATIC$_{DBL}$ based on Corollary 15.

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 6 is captured by adding $\mathcal{B}_{el}(\neg C)$ in the definition of the operator Φ' in Corollary 15. The second transformation in Definition 6 is simulated by adding $\neg\mathcal{B}_{el}\neg C_1 \vee \ldots \vee \neg\mathcal{B}_{el}\neg C_k$. NAF-consistent models are nothing else than arbitrary minimal models (as used in the construction of Corollary 15) satisfying the additional constraints formalized by the belief atoms.

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

5 Implementation with Polynomial Space Complexity

In the previous sections we have established a tight connection between minimal entailment and the D-WFS and STATIC semantics. In this section we show how this can be employed to develop a novel implementation method for D-WFS (and thus for STATIC$_{DBL}$) which contrary to most other implementations of disjunctive semantics works in *polynomial space*. For an implementation of the method, see `http://www.uni-koblenz.de/ag-ki/DLP`.

The idea is to exploit directly the new characterization of D-WFS in Theorem 16 and to implement D-WFS 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 call to a decision procedure of the minimal entailment $\models_{\min(\mathrm{N})}$. 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 $\mathrm{R_D}(\cdot)$ mirroring the operator in $\mathrm{SEM_c}$.

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

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 is obtained by

- removing each rule such that $C_1, \ldots, C_n$ are all the negative body literals and $C_1 \vee \cdots \vee C_n$ is a classical consequence of $P/At_{\mathcal{L}}$ and
- removing all $\mathit{not}\,(C)$ from the bodies of the remaining rules 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 20. $\mathrm{SEM_c}(P) = \mathit{Dis}_{\min}(P^*)$.

Next we consider the problem of computing P^* within a polynomial space limit. Clearly, the required series of reductions can be implemented if decision procedures for classical entailment and minimal entailment $\models_{\min(\mathrm{N})}$ are available. If the decision procedures work in polynomial space, then the whole series can be implemented within a polynomial space limit as the size of the program decreases after each step. Classical entailment can be implemented in polynomial space and next we show how this is possible also for $\models_{\min(\mathrm{N})}$ by extending a new tableau method for circumscription presented in [Nie96b,Nie96a].

Theorem 21. *The entailment relation* $\models_{\min(\mathrm{N})}$ *can be implemented in polynomial space.*

Proof. (Sketch) The tableau method in [Nie96b,Nie96a] provides an implementation method for circumscription which contrary to other tableau methods for circumscription, e.g. [Oli92,BY96], works in polynomial space.

In order to handle circumscription the novel method extends a standard tableau method for first-order logic in the following way. 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 order to extend a tableau method to handle circumscription one has to tighten the notion of an open branch and close all branches corresponding to non-minimal models of the premises. In the previous work the extension of the closing rule for branches has been based on comparing different branches in the tableau. However, in the novel method the closing rule is extended by exploiting a novel characterization of minimal models which makes it possible to test whether a model is a minimal one independent of other models. This enables an implementation method of the tableau calculus which develops a tableau one branch at a time and, hence, works in polynomial space [Nie96b,Nie96a]. In order to extend the novel method to cope with $\models_{\min(\mathrm{N})}$ the only change is to further strengthen the closing condition for branches and require that only branches corresponding to *NAF-consistent* minimal models are left open. The test for NAF-consistency can be implemented using the following observation: a model is NAF-consistent for P iff

$$P \not\models A_1 \vee \cdots \vee A_m,$$

where $\{\mathit{not}\, A_1, \ldots, \mathit{not}\, A_m\}$ are all the $\mathit{not}\,(\phi)$ atoms in the model. Thus, the strengthened closing rule can be implemented in polynomial space as classical entailment can be decided within polynomial space implying that $\models_{\min(\mathrm{N})}$ can be implemented in polynomial space. $\square$

Hence, we have established the following.

Theorem 22 (D-WFS in Polynomial Space).
D-WFS can be implemented within a polynomial space bound.

6 Conclusion

While the semantics of non-disjunctive programs is fairly well understood and there exist several implementations of the best known semantics (well-founded 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,BD97], seems to significantly enhance our understanding of the theoretical and algorithmic aspects of disjunctive logic programs (in [BD96a] it is shown that also the disjunctive stable semantics can be described by appropriate transformations). Both semantics extend the well-founded semantics of normal programs and have very natural and regular features analogous to those shared by the well-founded semantics. Although the two semantics are based on very different ideas, they turn out to be quite closely related as we have just shown.

In this paper we conducted a detailed study of the relationship between the static and the D-WFS semantics in the class of so called *belief-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 the underlying minimal model reasoning procedure for an efficient implementation.

Acknowledgments

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

References

[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.

[BD96a] Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. *Journal of Logic Programming*, forthcoming, 1996. (Extended abstract appeared in: Characterizations of the Stable Semantics by Partial Evaluation *LPNMR, Proceedings of the Third International Conference, Kentucky*, pages 85–98, 1995. Springer.).

[BD96b] Stefan Brass and Jürgen Dix. Characterizing D-WFS: Confluence and Iterated GCWA. In L.M. Pereira J.J. Alferes and E. Orlowska, editors, *JELIA '96*, LNCS 1126, pages 268–283. Springer, 1996. Extended version will appear in the *Journal of Automated Reasoning* in 1997.

[BD97] Stefan Brass and Jürgen Dix. Semantics of Disjunctive Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, accepted for publication, will appear in 1997. (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, *JELIA '96*, LNCS 1126, pages 1–17. Springer, 1996.

[BY96] F. Bry and A. Yahya. Minimal model generation with positive unit hyperresolution tableaux. In *Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 143–159, Terrasini, Italy, May 1996. Springer-Verlag.

[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.

[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.

[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.

[Min93] Jack Minker. An Overview of Nonmonotonic Reasoning and Logic Programming. *Journal of Logic Programming, Special Issue*, 17, 1993.

[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.

[Oli92] N. Olivetti. A tableaux and sequent calculus for minimal entailment. *Journal of Automated Reasoning*, 9:99–139, 1992.

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

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