On the Equivalence of the Static and Disjunctive Well-Founded Semantics and its Computation ⋆

Stefan Brass

School of Information Sciences, University of Pittsburgh, Pittsburgh, PA 15260, USA
`sbrass@mail.sis.pitt.edu`

Jürgen Dix [1]

Dept. of Computer Science, University of Maryland, College Park, MD 20742, USA
`dix@cs.umd.edu`

Ilkka Niemelä [2]

Dept. of Computer Science and Engineering, Helsinki University of Technology, FIN-02015 HUT, Finland
`Ilkka.Niemela@hut.fi`

Teodor C. Przymusinski [3]

Dept. of 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 Programming* and *Disjunctive Deductive Databases.* While the semantics of non-disjunctive programs is fairly well understood, the declarative and computational foundations of disjunctive logic programming proved to be much more elusive and difficult. Recently, two new and promising semantics have been proposed for the class of disjunctive logic programs. The first one is the *static* semantics STATIC, proposed by Przymusinski, and, the other is the *disjunctive well-founded* semantics D-WFS, proposed by Brass and Dix.

Although the two semantics are based on very different ideas, both of them have been shown to share a number of natural and intuitive properties. In particular, both

Preprint submitted to Elsevier Preprint *4 October 1999*

of them extend the well-founded semantics of normal logic programs. Nevertheless, since the static semantics employs a much richer underlying language than the D-WFS semantics, in general, the two semantics are different. The main result of this paper shows, however, that, when restricted to a common language, the two semantics in fact coincide. This important result provides additional and powerful argument in favor of the two semantics. It also allows us to use a recently developed minimal model theorem prover for an efficient implementation of the two semantics.

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 her/his decision, otherwise she/he 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 [21]);
- *Natural language understanding:* here there are many possibilities for ambiguity and they are represented most naturally by multiple intended models;

[⋆] Preliminary results of this paper appeared in the Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), A. G. Cohn and L. K. Schubert and S. C. Shapiro (editors), Trento, Italy, 1998, pp. 74–85.
[1] On leave from Institut für Informatik, Universität Koblenz-Landau, Rheinau 1, 56075 Koblenz, Deutschland.
[2] The author was partially supported by the Academy of Finland through Project 43963.
[3] The author was partially supported by the National Science Foundation grant #IRI-9313061.

- *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 [9].

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 [14,16,15] significantly simplifies the problem of *translation* of non-monotonic formalisms into logic programs, and, consequently, facilitates using logic programming as an *inference engine* 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 [4]. Quite recently, however, two new and promising semantics have been proposed for the class of disjunctive logic programs. The first one is the *static semantics* proposed by Przymusinski in [30] and the other is the *disjunctive well-founded semantics* D-WFS proposed by Brass and Dix in [3]. 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).

Although the two semantics stem from very different ideas, both of them have been shown to share a number of natural and intuitive properties [3,30]. In particular, both of them extend the well-founded semantics of normal logic programs. It is fairly easy to show that the static semantics is invariant under the program transformations on which the definition of D-WFS is based and therefore it is always at least as strong as D-WFS. Nevertheless, 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 not difficult to give examples of programs whose static semantics is strictly stronger than the D-WFS semantics.

The main result of this paper shows, however, that, when restricted to a

[4] The book by Minker et. al. [22] provides a detailed, although already somewhat outdated, account of the extensive research effort in this area. See also [12,25,30,10].

common *query language* the two semantics become entirely *equivalent*:

$$\mathsf{STATIC}_{\text{query}} \ \equiv \ \mathsf{D\text{-}WFS}\ .$$

This fundamental result is established by using an elegant and powerful characterization of $\mathsf{STATIC}_{\text{query}}$, the static semantics restricted to the query language $\mathfrak{L}_{\text{AEB}}^{\text{query}}$. In its proof we also use a characterization of least static expansions from [8] and a characterization of $\mathsf{D\text{-}WFS}$ contained in [6].

We consider the main result of this paper to be quite significant. The $\mathsf{D\text{-}WFS}$ semantics [3] 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 [30] 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 [3,30] (see also [31,6,7]) contain a thorough discussion of the two semantics, compare them to other semantics and demonstrate 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, one of the authors [27,26] recently introduced a novel and efficient method of computing minimal models (circumscription with fixed predicates) of positive logic programs which has polynomial space complexity and is capable of handling hundreds of thousands of minimal models. Our second main result shows how this method can be nicely adapted for the computation of the $\mathsf{D\text{-}WFS}$ and static semantics.

The paper is organized as follows. In the next Section 2 we briefly recall the definition and a characterization of the *disjunctive well-founded semantics* $\mathsf{D\text{-}WFS}$, originally given in [3]. In Section 3 we recall the definition and basic properties of belief theories in the *autoepistemic logic of beliefs* (AEB) and their *static semantics*, originally introduced in [31]. We also give an example showing that, in general, the static and the $\mathsf{D\text{-}WFS}$ semantics do *not* coincide. In Section 4 we introduce $\mathsf{STATIC}_{\text{query}}$, the static semantics restricted to the query language $\mathfrak{L}_{\text{AEB}}^{\text{query}}$ and we provide its simple fixed-point characterization. In Section 5 we prove that $\mathsf{STATIC}_{\text{query}}$ coincides with $\mathsf{D\text{-}WFS}$. In Section 6 we demonstrate how to use our characterizations of the static and the $\mathsf{D\text{-}WFS}$ semantics for an efficient implementation of the two semantics based on the methods introduced recently by Niemelä [27,26]. Related work is discussed in Section 7. We close with concluding remarks in Section 8.

2 Disjunctive Programs and Disjunctive Well-Founded Semantics

In this section we define disjunctive logic programs and we recall the definition and a characterization of the *disjunctive well-founded semantics* (D-WFS), originally introduced in [3]. Disjunctive logic programs are described using the *language of database logic*, $\mathfrak{L}_{\mathrm{DL}}$, which we introduce first.

Definition 1 (Language of Database Logic, $\mathfrak{L}_{\mathrm{DL}}$) *We call the propositional language generated by the connectives* $|$ *(denoting disjunction),* $\wedge$, $\leftarrow$, not *(denoting negation) and* $\top$ *(denoting empty conjunction) the language of database logic and we denote it*[5] *by* $\mathfrak{L}_{\mathrm{DL}}$.

Definition 2 (Disjunctive Logic Programs) *A disjunctive logic program* P *is a finite set of rules of the form:*

$$A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \mathrm{not}\, C_1 \wedge \cdots \wedge \mathrm{not}\, C_n,$$

written in the language of database logic, $\mathfrak{L}_{\mathrm{DL}}$. *We assume that* $k > 0$. *It is often convenient to abbreviate these rules as follows:*

$$\mathbf{A} \leftarrow \mathbf{B}, \mathrm{not}\, \mathbf{C},$$

where $\mathbf{A} = \{A_1, \ldots, A_k\}$, $\mathbf{B} = \{B_1, \ldots, B_m\}$ *and* $\mathbf{C} = \{C_1, \ldots, C_n\}$ *are finite sets of atoms and* $\mathbf{A} \neq \emptyset$. *We denote by* $\mathrm{Head}(P)$ *the set of all atoms occurring in rule heads of* P*; these are atoms that are* possibly true.

Note that in the above definition we did not use the classical connectives $\neg$, $\rightarrow$, and $\vee$ because their usual meaning does not correspond to the meaning defined by semantics for logic programs. However, when we later introduce the STATIC semantics, we will translate logic programs into modal formulae over the classical propositional language $\mathfrak{L}$:

Definition 3 (Language of Classical Logic, $\mathfrak{L}$) *We call the propositional language generated by the connectives* $\neg$, $\wedge$, $\rightarrow$, $\vee$ *and* $\top$, *with their usual meaning, the language of classical logic and we denote it by* $\mathfrak{L}$.

Observe that we are using the classical conjunction $\wedge$ in both frameworks because its meaning is the same in both contexts.

When we consider answering *queries* about logic programs or general databases, it is most common to evaluate only queries consisting of *pure disjunctions*, i.e., disjunctions of purely positive or purely negative literals. Of course, once we know answers to queries involving pure disjunctions we

[5] Clearly, $\mathfrak{L}_{\mathrm{DL}}$ also depends on the underlying signature $\mathfrak{sig}$ but we do not need to make it explicit because it will be understood from the context.

can immediately extend them to queries involving *conjunctions* of such disjunctions. This leads us to the definition of *query languages* in classical and database logics:

Definition 4 (Query Languages $\mathfrak{L}^{\mathbf{query}}$ and $\mathfrak{L}^{\mathbf{query}}_{\mathbf{DL}}$) *We define* positive *and* negative *formulae in the language of classical logic $\mathfrak{L}$ (respectively, in the language of database logic $\mathfrak{L}_{\mathrm{DL}}$) as follows (here $A_1, A_2, \ldots, A_n$ denote atomic formulae):*

- *disjunctions of the form $A_1 \vee \ldots \vee A_n$ (respectively, $A_1 \mid \ldots \mid A_n$), including the atom $\top$, are positive;*
- *disjunctions of the form $\neg A_1 \vee \ldots \vee \neg A_n$ (respectively, $\mathrm{not}\, A_1 \mid \ldots \mid \mathrm{not}\, A_n$), including $\neg\top$ (respectively, $\mathrm{not}\,\top$), are negative;*
- *conjunctions of positive formulae are positive;*
- *conjunctions of negative formulae are negative.*

By the query language of classical logic*, $\mathfrak{L}^{query}$ (respectively,* query language of database logic*, $\mathfrak{L}^{query}_{\mathrm{DL}}$) we mean the language consisting of all positive and all negative formulae.*

In other words, in the query language mixed disjunctions like "$A_1 \vee \neg A_2$" or "$A_1 \mid \mathrm{not}\, A_2$"are *not* allowed. Note that a given formula may not be in the above form, but it may be tautologically equivalent to such a formula. For example, the formula $\neg(A \to \neg B)$ is tautologically equivalent to $A \wedge B$ (here A and B are atoms) and hence it is positive. Positive or negative disjunctions will be jointly referred to as *pure disjunctions*.

We note that most semantics described by procedural means and lacking a model theoretical characterization only allow the derivation of pure disjunctions (and therefore also their conjunctions). Deriving mixed disjunctions often requires additional effort and is not straightforward. If on the other hand, a semantics is defined by a set of models, then mixed disjunctions can be evaluated in these models and such semantics allow the definition of an entailment relation on a larger class than just pure disjunctions. This is the case for the STATIC semantics (to be defined in Section 3.1) which is defined for arbitrary formulae of the underlying modal language.

One possible way of describing a semantics of a logic program P is by defining a suitable *semantic operator* $\mid\!\sim$ which satisfies some natural conditions and determines which disjunctions belong to the semantics of a given program. The semantics $\mathsf{S}_{\mid\sim}(P)$ determined by such an operator consists of all (conjunctions of) derivable disjunctions. In order for the resulting semantics to be compatible with various already existing semantics of logic programs, the conditions imposed on the semantic operator must be rather weak so that they are satisfied by a great majority of existing semantics. Moreover, the domain of such a semantic operator must be restricted to *pure disjunctions* so that it can be compared with other semantics defined only as subsets of the *query language* $\mathfrak{L}_{\mathrm{DL}}^{\mathrm{query}}$ of database logic.

Definition 5 (Semantic Operator $\mid\!\sim$) *By a semantic operator $\mid\!\sim$ we mean a binary relation between logic programs and the set of pure disjunctions of database logic which satisfies the following three arguably obvious conditions (here Δ and Δ' denote pure disjunctions):*

(1) Right Weakening: *If $P \mid\!\sim \Delta$ and $\Delta \subseteq \Delta'$, then $P \mid\!\sim \Delta'$;*
(2) Necessarily True: *If $\mathbf{A} \leftarrow true \in P$, for a disjunction $\mathbf{A}$, then $P \mid\!\sim \mathbf{A}$;*
(3) Necessarily False: *If $A \notin \mathrm{Head}(P)$, for some atom A, then $P \mid\!\sim \mathrm{not}\, A$.*

We can immediately extend *the semantic operator $\mid\!\sim$ to arbitrary formulae in the query language of database logic $\mathfrak{L}_{\mathrm{DL}}^{query}$ by stating that the operator $\mid\!\sim$ derives a formula F that is a conjunction of pure disjunctions if and only if it derives all of its pure disjunctions. Given a semantic operator $\mid\!\sim$ and a logic program P, by the semantics $\mathsf{S}_{\mid\sim}(P)$ of P* determined by $\mid\!\sim$ *we mean the set of all formulae in the query language of database logic $\mathfrak{L}_{\mathrm{DL}}^{query}$ that are derivable from P by $\mid\!\sim$, i.e.,*

$$\mathsf{S}_{\mid\sim}(P) := \{\Delta \in \mathfrak{L}_{\mathrm{DL}}^{query} : P \mid\!\sim \Delta\}.$$

According to this definition, any semantics $\mathsf{S}_{\mid\sim}(P)$ determined by a semantic operator $\mid\!\sim$ is a *subset* of the query language of database logic, $\mathfrak{L}_{\mathrm{DL}}^{\mathrm{query}}$.

2.1 Definition of D-WFS

The three conditions imposed on the semantic operator in the previous definition are too weak to generate, by themselves, an interesting semantics. The question therefore arises: what additional reasonable properties should be satisfied by any semantic operator? Below we list four natural *program transformations* with the property that all semantics proposed in the last 10 years are *invariant* under these transformations:

TAUT: Delete a rule $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}$ with $\mathbf{A} \cap \mathbf{B} \neq \emptyset$.
NMIN: Delete a rule $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}$ if there is another rule $A' \leftarrow \mathbf{B}' \wedge \mathrm{not}\, \mathbf{C}'$ with $\mathbf{A}' \subseteq \mathbf{A}$, $\mathbf{B}' \subseteq \mathbf{B}$, and $\mathbf{C}' \subseteq \mathbf{C}$. We assume that at least one inclusion $\subseteq$ is strict.
RED$^+$: Replace a rule $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}$ by $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, (\mathbf{C} \cap \mathrm{Head}(P))$.
RED$^-$: Delete a rule $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}$, if there is a rule $(\mathbf{A}' \leftarrow)$ with $\mathbf{A}' \subseteq \mathbf{C}$.

Note that although all of these transformations are defined for disjunctive programs, they can also be used for non-disjunctive programs. However, they are still not sufficient to characterize even the well-founded semantics of non-disjunctive programs. The key observation made in [3,7] was that a simple additional transformation, called *unfolding* or *partial evaluation*, which corresponds to hyperresolution (see also [4]), suffices to characterize not only WFS but also the stable semantics ([5]) and allows us to extend WFS to the disjunctive case.

Generalized Partial Evaluation (GPPE): Replace a rule $\mathbf{A} \leftarrow \mathbf{B} \wedge \operatorname{not}\mathbf{C}$, where $\mathbf{B}$ contains a distinguished atom B, by the rules

$$\mathbf{A} \cup \big(\mathbf{A}_i \setminus \{B\}\big) \;\leftarrow\; \big(\mathbf{B} \setminus \{B\}\big) \cup \mathbf{B}_i \;\wedge\; \operatorname{not}\big(\mathbf{C} \cup \mathbf{C}_i\big),$$

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

GPPE has been independently introduced by Sakama and Seki and by Brass and Dix in [32,3]. It allows us to get rid of positive atoms in the body. Given a program rule, we are free to select a specific positive occurrence of an atom B in the body and then resolve this rule with all the 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 does not appear in any head, in which case the whole rule is simply deleted. GPPE is almost identical to *hyperresolution* in automated reasoning, the only difference being that the original rule is removed, while in hyperresolution it is not.

We are now ready to define the D-WFS as the weakest semantics that is invariant under all of our transformations. Note that the following definition is in fact a theorem:

Definition 6 (Disjunctive Well-Founded Semantics, [7]) *Let P be a disjunctive logic program. There exists the weakest semantics among all semantics $\mathsf{S}_{\mid\!\sim}(P)$ that are determined by some semantic operator $\mid\!\sim$ and are invariant under the above five program transformations TAUT, NMIN, RED$^+$, RED$^-$ and GPPE.*

Moreover, this unique semantics is consistent and closed under logical consequences (within the query language $\mathfrak{L}^{query}_{\mathrm{DL}}$). We call it the Disjunctive Well-Founded Semantics *D-WFS. For normal programs, it coincides with the well-founded semantics (WFS), and, for positive disjunctive programs, it coincides with the minimal model semantics (GCWA) restricted*[6] *to $\mathfrak{L}^{query}_{\mathrm{DL}}$.*

---

[6] Note that the original definition of GCWA was given by means of a certain set of negative literals and that negative disjunctions were not allowed. Only the subsequent extension of GCWA, called EGCWA and defined in [20], allowed for negative disjunctions to be part of the semantics. Nevertheless, it is easy to extend D-WFS so that it also derives the same pure disjunctions as those that are true in all minimal

It turns out that the above set of transformations is both *confluent* and *terminating* [6]. This means that, given a program P, we can apply these transformations in an arbitrary order and eventually we will reach a *normal form* $res(P)$ of the original program P that cannot be further transformed (reduced).

Theorem 7 (Computation of D-WFS) *The set of the above listed transformations is confluent. Moreover, the D-WFS semantics can be immediately determined from the normal form $res(P)$ of a program P as follows:*

$$P \mid\!\sim_{\textsf{D-WFS}} \Delta \ \textit{iff} \begin{cases} \exists\, \mathbf{A} \subseteq \Delta \ \textit{with} \ (\mathbf{A} \leftarrow) \in res(P), \ \textit{or}, \\ \exists \mathrm{not}\, A \in \Delta \ \textit{and} \ A \notin \mathrm{Head}(res(P)). \end{cases}$$

This result provides us with a very good method of actually computing the D-WFS semantics for small programs and we will frequently use it in the sequel.

Remark 8 *Observe that while the D-WFS semantics may yield irreducible positive disjunctions $A_1 \mid \ldots \mid A_n$, with $n \geq 2$, the only irreducible negative disjunctions deducible from D-WFS are single negative literals* not A.

2.2 Characterization of D-WFS

In order to compare the D-WFS to the static semantics, we will need a completely different and much more technical characterization of D-WFS taken from [6]. It is similar to Gelfond-Lifschitz' characterization of stable models [17] and is based on the notion of *program reduction modulo a set of disjunctions Dis*:

Definition 9 (Program Reduction P/Dis)
Let P be a disjunctive logic program and let Dis be a set of pure disjunctions in $\mathfrak{L}_{\mathrm{DL}}^{query}$. Let P/Dis be the program obtained from P by performing the following reductions, for all not C *and $C_1 \mid \ldots \mid C_k$:*

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

P/Dis is obviously a slight generalization of the Gelfond-Lifschitz transformation. While the latter is defined relative to a set $N \subseteq \mathrm{At}_{\mathfrak{L}}$ in such a way that P/N is always positive*, our P/Dis still is a disjunctive program possibly*

models.

containing negative atoms. In fact, the GL-transform can be obtained from our transform by setting $P/N = P/Dis_N$*, where* $Dis_N := N \cup \{\text{not}\, X : X \in \mathrm{At}_{\mathfrak{L}} \setminus N\}$.

The idea of the following definition is to view a program as a classical theory and consider classical derivability in subtle ways. As already elaborated in Definition 2, our programs are not formulated over classical logic $\mathfrak{L}$, due to the nonstandard meaning of $\leftarrow$, $|$ and not . However, it is straightforward to translate a program into a classical theory:

Definition 10 (Translation $\mathfrak{tr}(P)$ of P into Classical Logic) *Given a program P viewed as a theory in $\mathfrak{L}_{\mathrm{DL}}$, we can translate P into classical logic over the same signature by defining*

$$\begin{aligned} \mathfrak{tr}(\leftarrow) &:= \rightarrow \\ \mathfrak{tr}(|) &:= \vee \\ \mathfrak{tr}(\text{not}\,) &:= \neg \\ \mathfrak{tr}(\wedge) &:= \wedge \\ \mathfrak{tr}(\top) &:= \top \end{aligned}$$

The mapping $\mathfrak{tr}$ *straightforwardly extends to arbitrary disjunctive logic programs so that*

$$\mathfrak{tr}(\mathbf{A} \leftarrow \mathbf{B}, \text{not}\, \mathbf{C}) := \; B_1 \wedge \ldots \wedge B_m \wedge \neg C_1 \wedge \ldots \wedge \neg C_n \rightarrow A_1 \vee \cdots \vee A_k.$$

Definition 11 (SEM_0)
Let P be a disjunctive logic program over $\mathfrak{L}_{\mathrm{DL}}$. We define the set $Dis(P) \subseteq \mathfrak{L}_{\mathrm{DL}}^{query}$ of pure disjunctions $Dis(P) := Dis^+(P) \cup Dis^-(P)$, where:

$$Dis^+(P) := \{A_1 \mid \ldots \mid A_k : \mathfrak{tr}(P/\mathrm{At}_{\mathfrak{L}}) \models A_1 \vee \ldots \vee A_k\}$$

$$\begin{aligned} Dis^-(P) := \{\text{not}\, A : \mathfrak{tr}(P/Dis_N) \models_{min} \neg A \quad & \textit{for all}\, N \subseteq \mathrm{At}_{\mathfrak{L}_{\mathrm{DL}}} \\ & \textit{s. t. } N \models \mathfrak{tr}(Dis^+(P))\,\}. \end{aligned}$$

It turns out[7] *that the series D_i, where $D_0 := \emptyset$ and $D_{n+1} := Dis(P/D_n)$, grows monotonically and eventually becomes constant. We define $\mathsf{SEM}_0(P)$ to be the limit of this series.*

The underlying idea in the definition of SEM_0 is to use $P/\mathrm{At}_{\mathfrak{L}_{\mathrm{DL}}^{\mathrm{query}}}$ (obtained by deleting all rules containing negative literals) as a classical theory, i.e. using

[7] We need to consider only those $N \subseteq \mathrm{At}_{\mathfrak{L}}$ that are compatible with $Dis^+(P)$. For example, for the program "$A \mid B,\; C \leftarrow \text{not}\, A \wedge \text{not}\, B$" we do not want to consider $N_0 := \emptyset$ because then not C would not be derivable.

$\mathbf{tr}(P/\mathrm{At}_{\mathfrak{L}_{\mathrm{DL}}^{\mathrm{query}}})$ for deriving positive disjunctions and to use P/Dis_N (obtained by setting all negative literals outside N to true), i.e., to use $\mathbf{tr}(P/Dis_N)$ in order to derive negative literals. Here $\models_{min}$ stands for the usual notion of *"truth in all minimal models"*.

Theorem 12 (D-WFS = SEM$_0$, [6]) *For any logic program P the* D-WFS *semantics coincides with the set of all positive (respectively, negative) conjunctions of disjunctions from* SEM$_0$.

3 Belief Theories and STATIC Semantics

In this section we recall the notion of a *belief theory* in the *Autoepistemic Logic of Beliefs (*AEB*)*, introduced in [31], and the definition and basic properties of the *static semantics* of belief theories. Disjunctive logic programs will be viewed as special belief theories. We also give an example showing that, in general, the static semantics and the D-WFS semantics do *not* coincide.

We start with the classical propositional language $\mathfrak{L}$ introduced in Definition 3. We write $\mathrm{At}_{\mathfrak{L}}$ for the set of all the atoms different from $\bot$.

Definition 13 ($\mathfrak{L}_{\mathbf{AEB}}$) *We extend $\mathfrak{L}$ to a propositional modal language $\mathfrak{L}_{\mathrm{AEB}}$ by augmenting it with a modal operator $\mathcal{B}$, called the* belief *operator. The set of formulae in $\mathfrak{L}_{\mathrm{AEB}}$ is defined as usual by induction on the size of the formula. Formulae of the form $\mathcal{B}F$, where F is an arbitrary formula of $\mathfrak{L}_{\mathrm{AEB}}$, are called* belief atoms *and are considered to be atomic formulae in the extended propositional modal language $\mathfrak{L}_{\mathrm{AEB}}$. The formulae of the original language $\mathfrak{L}$ are called* objective. *Observe that arbitrarily deep levels of* nested beliefs *are allowed in the language.*

A (propositional) interpretation of $\mathfrak{L}_{\mathrm{AEB}}$ is a mapping

$$\mathcal{I}\colon\ \mathrm{At}_{\mathfrak{L}} \cup \{\mathcal{B}(F) \mid F \in \mathfrak{L}_{\mathrm{AEB}}\} \longrightarrow \{true, false\},$$

s. t. $\mathcal{I}(\top) = true$, i.e., we simply treat the formulas $\mathcal{B}(F)$ as new propositions. Therefore, the notion of a model carries over from propositional logic. A formula $F \in \mathfrak{L}_{\mathrm{AEB}}$ is a propositional consequence of $T \subseteq \mathfrak{L}_{\mathrm{AEB}}$ iff for every interpretation $\mathcal{I}$: $\mathcal{I} \models T$ implies $\mathcal{I} \models F$.

A propositional theory in the modal language $\mathfrak{L}_{\mathrm{AEB}}$ will be called a *belief theory*. Thus a belief theory is a (possibly infinite) set of clauses of the form:

$$B_1 \wedge \ldots \wedge B_m \wedge \mathcal{B}G_1 \wedge \ldots \wedge \mathcal{B}G_k \to A_1 \vee \ldots \vee A_l \vee \mathcal{B}F_1 \vee \ldots \vee \mathcal{B}F_n$$

where $m, n, k, l \geq 0$, A_i's and B_i's are objective atoms and F_i's and G_i's are arbitrary formulae of $\mathfrak{L}_{\mathrm{AEB}}$.

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

(**CA**) **Consistency Axiom:** For any formula $F \in \mathfrak{L}_{\mathrm{AEB}}$:

$$\mathcal{B}(\top) \quad \text{and} \quad \mathcal{B}F \rightarrow \neg\mathcal{B}\neg F \tag{1}$$

(**DA**) **Distributive Axiom:**
For any formulae F and G from $\mathfrak{L}_{\mathrm{AEB}}$:

$$\mathcal{B}(F \wedge G) \leftrightarrow \mathcal{B}F \wedge \mathcal{B}G \tag{2}$$

(**IR**) **Invariance Inference Rule:**
For any formulae F and G from $\mathfrak{L}_{\mathrm{AEB}}$:

$$\frac{F \leftrightarrow G}{\mathcal{B}F \leftrightarrow \mathcal{B}G} \tag{3}$$

The consistency axiom (**CA**) states that *true* is believed and that if a formula F is believed then its negation $\neg F$ is not believed. The second axiom (**DA**) states that beliefs $\mathcal{B}$ are distributive with respect to conjunctions. The invariance inference rule (**IR**) states that if two formulae are known to be equivalent then so are beliefs in these formulae. In other words, the meaning of $\mathcal{B}F$ does not depend on the specific form of the formula F, e.g., the formula $\mathcal{B}(F \wedge \neg F)$ is equivalent to $\mathcal{B}(\neg\top)$ and thus is *false* by (**CA**).

Definition 14 (Derivable Formulae [31]) *For any belief theory T we denote by $\mathrm{Cn}_{\mathrm{AEB}}(T)$ the smallest set of formulae of the language $\mathfrak{L}_{\mathrm{AEB}}$ which contains T, all (substitution instances of) the axioms* (**CA**) *and* (**DA**) *and is closed under both propositional consequence and the invariance rule* (**IR**).

We say that a formula F is derivable *from the belief theory T if F belongs to* $\mathrm{Cn}_{\mathrm{AEB}}(T)$. *We denote this fact by $T \vdash_{\mathrm{AEB}} F$. A belief theory T is* consistent *if $\mathrm{Cn}_{\mathrm{AEB}}(T)$ is consistent, i.e., if $T \not\vdash_{\mathrm{AEB}} \neg\top$.*

Remark 15 *The notion of a* belief theory *defined above is completely equivalent to that originally introduced by Przymusinski in [31]. See [8] for a detailed discussion.*

Proposition 16 ([8]) *For any belief theory T and for any two formulae F and G of $\mathfrak{L}_{\mathrm{AEB}}$:*

$$T \vdash_{\mathrm{AEB}} \mathcal{B}(F \rightarrow G) \rightarrow (\mathcal{B}F \rightarrow \mathcal{B}G)$$

$$\textit{If} \ \ T \vdash_{\mathrm{AEB}} F \ \ \textit{then} \ \ T \vdash_{\mathrm{AEB}} \mathcal{B}F.$$

The first statement says that if we believe both in F and in $F \to G$ then we should also believe in G. The second inference rule says that if a formula F is derivable then it is believed (necessitation).

The intended meaning of belief atoms $\mathcal{B}F$ is based on the principle of *predicate minimization* (see [24,19] and [23]):

$$\mathcal{B}F \;\text{ holds if }\; F \text{ is minimally entailed}$$

or, equivalently:

$$\mathcal{B}F \;\text{ holds if }\; F \text{ is true in all minimal models.}$$

In order to make this intended meaning precise we first have to define what we mean by a minimal model of a belief theory.

Definition 17 (Minimal Models [31]) *A model M of a belief theory is* smaller *than another model N if it contains the same belief atoms but has fewer objective atoms, i.e.*

$$\{A \in \mathrm{At}_{\mathfrak{L}} \mid M \models A\} \subset \{A \in \mathrm{At}_{\mathfrak{L}} \mid N \models A\},$$
$$\{F \in \mathfrak{L}_{\mathrm{AEB}} \mid M \models \mathcal{B}(F)\} = \{F \in \mathfrak{L}_{\mathrm{AEB}} \mid N \models \mathcal{B}(F)\}.$$

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. 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 *by T.*

In other words, minimal models are obtained by first assigning *arbitrary* truth values to the belief atoms and then *minimizing* the objective atoms [8] (see also [35]).

As in Moore's Autoepistemic Logic, the intended meaning of belief atoms in belief theories is enforced by defining their suitable expansions.

Definition 18 (Static Expansions of Belief Theories [31]) *A belief theory $T^\diamond$ is called a* static expansion *of a belief theory T if it satisfies the following fixed-point equation:*

$$T^\diamond = \mathrm{Cn}_{\mathrm{AEB}}\Big(T \;\cup \{\mathcal{B}F : T^\diamond \models_{\min} F\}\Big),$$

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

[8] For readers familiar with *circumscription*, this means that we are considering circumscription $\mathrm{CIRC}(T; \mathrm{At}_{\mathfrak{L}})$ of the belief theory T in which objective atoms are minimized while the belief atoms $\mathcal{B}F$ are fixed, i.e., $T \models_{\min} F$ is equivalent to $\mathrm{CIRC}(T; \mathrm{At}_{\mathfrak{L}}) \models F$.

A static expansion $T^\diamond$ of T must therefore coincide with the theory obtained by: *(i)* adding to T beliefs $\mathcal{B}F$ in any formula F that is true in all minimal models of $T^\diamond$, and, *(ii)* closing the resulting theory under the consequence operator $\mathrm{Cn}_{\mathrm{AEB}}$.

It turns out that every belief theory T has the *least* (in the sense of set-theoretic inclusion) static expansion $\overline{T}$ which has an *iterative* definition as the *least fixed point* of the monotonic belief closure operator (we refer to [8, Theorem 2.7] as to the monotonicity of this operator):

$$\Psi_T(S) = \mathrm{Cn}_{\mathrm{AEB}}\Big(T \cup \{\mathcal{B}F : S \models_{\min} F\}\Big),$$

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

Theorem 19 (Least Static Expansion [31,30]) *Every belief theory T has the* least *static expansion, namely, the least fixed point $\overline{T}$ of the belief closure operator Ψ_T.*

More precisely, the least static expansion $\overline{T}$ of T can be constructed as follows. Let $T^0 = T$ and suppose that T^α has already been defined for any ordinal number $\alpha < \beta$. If $\beta = \alpha + 1$ is a successor ordinal then define:

$$T^{\alpha+1} = \Psi_T(T^\alpha) =_{def} \mathrm{Cn}_{\mathrm{AEB}}\Big(\; T \cup \{\mathcal{B}F : T^\alpha \models_{\min} F\}\;\Big),$$

where F ranges over all formulae in $\mathfrak{L}_{\mathrm{AEB}}$. Else, if β is a limit ordinal then define $T^\beta = \bigcup_{\alpha<\beta} T^\alpha$. The sequence $\{T^\alpha\}$ is monotonically increasing and has a unique fixed point $\overline{T} = T^\lambda = \Psi_T(T^\lambda)$, for some ordinal λ.

Observe that the *least* static expansion $\overline{T}$ of T contains those and only those formulae which are true in *all* static expansions of T. Like the predicate completion semantics of a logic program P is completely determined by its Clark's completion $\mathrm{comp}(P)$, the *static semantics* of a belief theory T is fully determined by its least static expansion $\overline{T}$.

3.1 STATIC Semantics of Disjunctive Logic Programs

Recall that disjunctive logic programs were defined in the language of database logic $\mathfrak{L}_{\mathrm{DL}}$. In Definition 10 we mapped logic programs into classical logic $\mathfrak{L}$. *STATIC semantics* of disjunctive programs is obtained by using another mapping $\mathfrak{tr}_{\mathrm{AEB}}$ which maps logic programs into belief theories of $\mathfrak{L}_{\mathrm{AEB}}$:

Definition 20 (Translation $\mathfrak{tr}_{\mathbf{AEB}}(P)$ of P into AEB) *Given a program P viewed as a theory in $\mathfrak{L}_{\mathrm{DL}}$, we can translate P into autoepistemic logic* *of beliefs over the same signature by defining*

$$\mathfrak{tr}_{\mathrm{AEB}}(\mathbf{A} \leftarrow \mathbf{B}, \mathrm{not}\,\mathbf{C}) := \quad B_1 \wedge \ldots \wedge B_m \wedge \mathcal{B}\neg C_1 \wedge \ldots \wedge \mathcal{B}\neg C_n \rightarrow A_1 \vee \cdots \vee A_k.$$

The translation, $\mathfrak{tr}_{\mathrm{AEB}}(P)$, gives therefore the following meaning to the default negation $\mathrm{not}\,C$:

$\mathrm{not}\,C$ defined by $\mathcal{B}\neg C$ iff C is believed *false* iff $\neg C$ is minimally entailed.

Since any disjunctive logic program P can now be simply viewed as a belief theory $\mathfrak{tr}_{\mathrm{AEB}}(P)$, according to Theorem 19, it has a well-defined *static semantics* given by the least static expansion $\overline{\mathfrak{tr}_{\mathrm{AEB}}(P)}$ of $\mathfrak{tr}_{\mathrm{AEB}}(P)$:

$$\mathsf{STATIC}(P) := \overline{\mathfrak{tr}_{\mathrm{AEB}}(P)}.$$

Note that this definition of $\mathsf{STATIC}(P)$ takes place entirely in $\mathfrak{L}_{\mathrm{AEB}}$. In order to compare it with other semantics in the language of database logic $\mathfrak{L}_{\mathrm{DL}}$ we have to switch back to the induced semantic operator $\mid\!\sim_{\mathrm{AEB}}$:

$$\begin{array}{lll} P \mid\!\sim_{\mathrm{AEB}} A_1 \mid \ldots \mid A_n & \text{iff} & A_1 \vee \ldots \vee A_n \in \overline{\mathfrak{tr}_{\mathrm{AEB}}(P)} \\ P \mid\!\sim_{\mathrm{AEB}} \mathrm{not}\,A_1 \mid \ldots \mid \mathrm{not}\,A_n & \text{iff} & \mathcal{B}\neg A_1 \vee \ldots \vee \mathcal{B}\neg A_n \in \overline{\mathfrak{tr}_{\mathrm{AEB}}(P)}. \end{array} \tag{4}$$

However, we will sometimes identify STATIC with $\mid\!\sim_{\mathrm{AEB}}$ if the precise meaning is clear from context.

The static semantics has been shown to have a number of natural properties which relate it to other semantics of logic programs and which are summarized in the following theorem (see [31,30,8] for a more detailed discussion and examples).

Theorem 21 (Basic Properties of STATIC Semantics [31,30]) *The semantics $\mathsf{STATIC}(P)$ of a disjunctive logic program P, defined as the least static expansion $\overline{\mathfrak{tr}_{\mathrm{AEB}}(P)}$ of the belief theory $\mathfrak{tr}_{\mathrm{AEB}}(P)$, has the following basic properties:*

- *$\mathsf{STATIC}(P)$ is always consistent, and, for any formula $F \in \mathfrak{L}_{\mathrm{AEB}}$, it satisfies the condition:*

  $$\mathsf{STATIC}(P) \models \mathcal{B}F \;\; \textit{iff} \;\; \mathsf{STATIC}(P) \models_{\min} F;$$

  in particular, for any positive formula F from $\mathfrak{L}_{\mathrm{AEB}}$:

  $$\mathsf{STATIC}(P) \models \mathcal{B}F \quad \textit{iff} \quad \mathsf{STATIC}(P) \models F;$$

moreover, for any two formulae F and G from $\mathfrak{L}_{\mathrm{AEB}}$:

$$\mathit{STATIC}(P) \models \mathcal{B}F \ \vee \ \mathcal{B}G \ \textit{ iff } \ (\mathit{STATIC}(P) \models \mathcal{B}F \textit{ or } \mathit{STATIC}(P) \models \mathcal{B}G).$$

- *For normal logic programs, static semantics coincides with the well-founded semantics [34]. More precisely, there is a one-to-one correspondence between consistent static expansions of a normal program and its partial stable models [28];*
- *For positive disjunctive logic programs, static semantics coincides with the minimal model semantics (GCWA).*

From the last theorem we immediately infer:

$$P \mathrel{|\!\sim}_{\mathrm{AEB}} \ \mathrm{not}\, A_1 \mid \ldots \mid \mathrm{not}\, A_n \ \text{iff} \ \ \exists i \ P \mathrel{|\!\sim}_{\mathrm{AEB}} \mathrm{not}\, A_i \ \text{iff} \ \ \exists i \ \mathcal{B}\neg A_i \in \overline{\mathfrak{tr}_{\mathrm{AEB}}(P)}.$$

The following result shows that $\mathrel{|\!\sim}_{\mathrm{AEB}}$ is indeed a semantic operator in the sense of Definition 5 and that it is invariant under all the five program transformations used in the definition of D-WFS semantics (Definition 6).

Theorem 22 (STATIC Semantics and D-WFS) *The operator $\mathrel{|\!\sim}_{\mathrm{AEB}}$ is indeed a semantic operator in the sense of Definition 5. Moreover, it is invariant under all the five program transformations (TAUT), (NMIN), (RED$^+$), (RED$^-$) and (GPPE). Consequently, the static semantics is always at least as strong as the D-WFS semantics, i.e.:*

$$\mathit{D\text{-}WFS}(P) \subseteq \mathit{STATIC}(P).$$

PROOF. We have to translate our conditions formulated for an abstract operator $\mathrel{|\!\sim}_{\mathrm{AEB}}$ via Equation (4) to the language of STATIC. Thus all statements translate into statements on belief theories formulated in $\mathfrak{L}_{\mathrm{AEB}}$.

TAUT: It suffices to show that

$$\overline{\mathfrak{tr}_{\mathrm{AEB}}(P')} = \overline{\mathfrak{tr}_{\mathrm{AEB}}(P)},$$

for $P' = P \cup \{\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}\}$ where $\mathbf{A} \cap \mathbf{B} \neq \emptyset$. But this follows immediately from

$$\vdash_{\mathrm{AEB}} \mathfrak{tr}_{\mathrm{AEB}}(\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}) \leftrightarrow \top$$

because of $\mathbf{A} \cap \mathbf{B} \neq \emptyset$ ($\mathfrak{tr}_{\mathrm{AEB}}(\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C})$ is a tautology).

NMIN: It suffices to show that

$$\overline{\mathfrak{tr}_{\mathrm{AEB}}(P')} = \overline{\mathfrak{tr}_{\mathrm{AEB}}(P)},$$

for $P' = P \cup \{\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}\}$ if there is another rule $A' \leftarrow \mathbf{B}' \wedge \mathrm{not}\, \mathbf{C}'$ with $\mathbf{A}' \subseteq \mathbf{A}$, $\mathbf{B}' \subseteq \mathbf{B}$, $\mathbf{C}' \subseteq \mathbf{C}$ and with at least one $\subseteq$ strict. But this follows immediately from

$$\vdash_{\mathrm{AEB}} \mathfrak{tr}_{\mathrm{AEB}}(A' \leftarrow \mathbf{B}' \wedge \mathrm{not}\, \mathbf{C}') \rightarrow \mathfrak{tr}_{\mathrm{AEB}}(\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}).$$

because of $\mathbf{A}' \subseteq \mathbf{A}$, $\mathbf{B}' \subseteq \mathbf{B}$, and $\mathbf{C}' \subseteq \mathbf{C}$.

RED$^+$: Suppose $c \notin \mathrm{Head}(P)$ and $c \in \mathbf{C}$ and $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C} \ \in P$. Let

$$P' = (P \setminus \{\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}\}) \cup \{\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C} \setminus \{c\}\}.$$

Then we have $\overline{\mathfrak{tr}_{\mathrm{AEB}}(P)} \vdash_{\mathrm{AEB}} \mathcal{B}(\neg c)$ because $\neg c$ is true in all minimal models of P (apply Theorem 21). Thus

$$\overline{\mathfrak{tr}_{\mathrm{AEB}}(P')} = \overline{\mathfrak{tr}_{\mathrm{AEB}}(P)},$$

because belief theories are deductively closed.

RED$^-$: Let $P' = P \cup \{\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}\}$ where there is a rule $(\mathbf{A}' \leftarrow)$ with $\mathbf{A}' \subseteq \mathbf{C}$. Then we have

$$\overline{\mathfrak{tr}_{\mathrm{AEB}}(\mathbf{A}' \leftarrow)} \vdash_{\mathrm{AEB}} \mathfrak{tr}_{\mathrm{AEB}}(\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}) \leftrightarrow \top$$

and thus $\overline{\mathfrak{tr}_{\mathrm{AEB}}(P')} = \overline{\mathfrak{tr}_{\mathrm{AEB}}(P)}$.

To prove GPPE is trivial because its translation into belief theories gives us a weaker form of the classical *hyperresolution rule* (see comments before Definition 6). It is weaker, because in GPPE we remove the original rule. It is well known that hyperresolution is correct for classical logic and it *preserves minimal models with varying and fixed atoms* (see [4,7,6]), so its weaker form is correct as well (weakening affects only the completeness). Therefore, because we are always taking deductive closures of our belief theories, GPPE is trivially satisfied. $\square$

3.2 STATIC Semantics is Strictly Stronger than D-WFS

Theorem 22 showed that the static semantics $\mathsf{STATIC}(P)$ of any disjunctive logic program P is always at least as strong as the $\mathsf{D\text{-}WFS}$ semantics:

$$\mathsf{D\text{-}WFS}(P) \subseteq \mathsf{STATIC}(P).$$

We now give an example of a simple disjunctive logic program P, whose static semantic STATIC (P) is strictly stronger than its $\mathsf{D\text{-}WFS}$ semantics, thus showing that Theorem 22 cannot be, in general, improved.

Example 23 *Let P be the program given by:*

$$\begin{aligned} a \mid c &\leftarrow \text{not}\, d \\ c \quad &\leftarrow \text{not}\, c \\ d \quad &\leftarrow c\ . \end{aligned}$$

First of all, we show that the static semantics $\mathit{STATIC}(P)$ implies not a. *Indeed, the translation $\mathfrak{tr}_{\mathrm{AEB}}(P)$ of P into a belief theory is given by:*

$$\begin{aligned} \mathcal{B}(\neg d) &\rightarrow a \vee c \\ \mathcal{B}(\neg c) &\rightarrow c \\ c \quad &\rightarrow d\ . \end{aligned}$$

Since $\overline{\mathfrak{tr}_{\mathrm{AEB}}(P)} \models \neg d \rightarrow \neg c$, it follows from the Proposition 1 that $\overline{\mathfrak{tr}_{\mathrm{AEB}}(P)} \models \mathcal{B}(\neg d \rightarrow \neg c)$. From Proposition 16 and the definition of static expansions we infer that $\overline{\mathfrak{tr}_{\mathrm{AEB}}(P)} \models \mathcal{B}(\neg d) \rightarrow \mathcal{B}(\neg c)$. This implies that a must be false in all minimal models of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ which in turn shows that $\overline{\mathfrak{tr}_{\mathrm{AEB}}(P)} \models \mathcal{B}(\neg a)$. Consequently, $\mathit{STATIC}(P) \models \text{not}\, a$.

On the other hand, of the five program transformations specified in the definition of D-WFS, only (GPPE) is applicable to P. It allows us to reduce the program P into an irreducible program $res(P)$ given by:

$$\begin{aligned} a \mid c &\leftarrow \text{not}\, d \\ c \quad &\leftarrow \text{not}\, c \\ a \mid d &\leftarrow \text{not}\, d \\ d \quad &\leftarrow \text{not}\, c\ . \end{aligned}$$

The semantic operator $P \mid\!\sim_{\text{D-WFS}}$ defined in Section 2.1 does not allow us to infer any pure disjunctions from $res(P)$ which shows that D-WFS$(P) \not\models \text{not}\, a$.

4 STATIC Semantics Restricted to the Query Language

In the previous section we gave an example of a disjunctive logic program P, whose static semantic $\mathsf{STATIC}(P)$ is strictly stronger than its D-WFS semantics, thus showing that Theorem 22 cannot be, in general, improved.

It is not difficult to see why this is the case. While the definition of the D-WFS semantics is strictly limited to the query language of database logic, $\mathfrak{L}_{\mathrm{DL}}^{\mathrm{query}}$, the language $\mathfrak{L}_{\mathrm{AEB}}$ used in the definition of belief theories and static semantics is incomparably richer. It allows arbitrarily complex belief atoms of the form $\mathcal{B}F$, where F is any formula of $\mathfrak{L}_{\mathrm{AEB}}$ itself. In particular, it allows nested beliefs, like $\mathcal{B}(\mathcal{B}A)$, and, complex beliefs, like $\mathcal{B}(\neg d \to \neg c)$, that appeared in Example 23. It turns out that it is precisely this difference in the scope of the languages used that causes the two semantics to differ.

In this section, we restrict the underlying language $\mathfrak{L}_{\mathrm{AEB}}$, used in the definition of belief theories and static semantics, to a narrower language $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$, which we call the query language of $\mathfrak{L}_{\mathrm{AEB}}$. The restricted query language $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$ allows only belief atoms of the form $\mathcal{B}F$, where F belongs to the query language $\mathfrak{L}^{\mathsf{query}}$:

$$\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}} = \mathfrak{L} \ \cup \{\mathcal{B}F : F \in \mathfrak{L}^{\mathsf{query}}\}.$$

In other words, the language $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$ allows only beliefs in positive or negative formulae of classical logic, $\mathfrak{L}$. In particular, it does not allow nested beliefs or beliefs in implications, like, $\mathcal{B}(\neg d \to \neg c)$.

It turns out that the static semantics restricted to the language $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$, denoted by $\mathsf{STATIC}_{\mathrm{query}}$, admits a simpler and more natural characterization which enables us to prove (in the next Section 5) that $\mathsf{STATIC}_{\mathrm{query}}$ in fact coincides with $\mathsf{D\text{-}WFS}$.

By a *belief theory* in the query language $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$ we mean any propositional theory in the language $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$. We assume the same axiom schemata (**CA**) and (**DA**) and the inference rule (**IR**) as before except that they are now restricted to formulae of the language $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$. It is important to note that the axioms (**CA**) and (**DA**) belong to $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$ (given that F, G are from $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$). Also (**IR**) applied to formulae F, G from $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$ derives new formulae belonging to $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$.

Definition 24 ($\mathbf{Cn}_{\mathbf{AEB}}^{\mathbf{query}}(\mathbf{T})$) *The set $\mathrm{Cn}_{\mathrm{AEB}}^{query}(T)$ of formulae derivable from a belief theory T is defined as in Definition 14 but it includes only formulae of $\mathfrak{L}_{\mathrm{AEB}}^{query}$.*

Similarly, a belief theory $T^{\diamond}$ is a *static expansion* of a belief theory T if it satisfies the fixed-point equation:

$$T^{\diamond} = \mathrm{Cn}_{\mathrm{AEB}}^{\mathrm{query}}\Big(T \ \cup \{\mathcal{B}F : T^{\diamond} \models_{\min} F\}\Big),$$

where F ranges over all formulae of $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$. As shown by the following theorem, every belief theory T has the *least* static expansion $\widehat{T}$, restricted to the language $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$, which has an *iterative* definition as the *least fixed point* of the monotonic belief closure operator:

$$\Phi_T(S) = \mathrm{Cn}_{\mathrm{AEB}}^{\mathrm{query}}\Big(T \cup \{\mathcal{B}F : S \ \models_{\min} F\}\Big),$$

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

Theorem 25 (Least Static Expansion $\widehat{T}$ in $\mathfrak{L}_{\mathbf{AEB}}^{\mathbf{query}}$) *Every belief theory T in $\mathfrak{L}_{\rm AEB}^{query}$ has the* least *static expansion, namely, the least fixed point $\widehat{T}$ of the belief closure operator Φ_T.*

More precisely, the least static expansion $\widehat{T}$ of T can be constructed as follows. Let $T^0 = T$ and suppose that T^α has already been defined for any ordinal number $\alpha < \beta$. If $\beta = \alpha + 1$ is a successor ordinal then define:

$$T^{\alpha+1} = \Phi_T(T^\alpha) =_{def} \mathrm{Cn}_{\rm AEB}^{query}\Big(T \cup \{\mathcal{B}F : T^\alpha \ \models_{\min} \ F\} \Big),$$

where F ranges over all formulae in $\mathfrak{L}_{\rm AEB}^{query}$. Else, if β is a limit ordinal then define $T^\beta = \bigcup_{\alpha<\beta} T^\alpha$. The sequence $\{T^\alpha\}$ is monotonically increasing and has a unique fixed point $\widehat{T} = T^\lambda = \Phi_T(T^\lambda)$, for some ordinal λ.

PROOF. The construction of the least fixpoint follows from the construction of the least fixpoint of Ψ_T in the full language $\mathfrak{L}_{\rm AEB}$. $\Box$

Corollary 26 (Another Characterization of $\widehat{T}$ in $\mathfrak{L}_{\mathbf{AEB}}^{\mathbf{query}}$) *The least static expansion $\widehat{T}$ in $\mathfrak{L}_{\rm AEB}^{query}$ of a theory T coincides with the smallest theory $\tilde{T}$ satisfying the conditions:*

(1) $T \subseteq \tilde{T}$;
(2) $\tilde{T} = \mathrm{Cn}_{\rm AEB}^{query}(\tilde{T})$;
(3) if $\tilde{T} \models_{\min} F$ where $F \in \mathfrak{L}^{query}$, then $\mathcal{B}F \in \tilde{T}$.

PROOF. Clearly the least static expansion $\widehat{T}$ satisfies the above 3 conditions, therefore $\widehat{T} \subseteq \tilde{T}$. But these conditions also imply $\Phi_T(\tilde{T}) \subseteq \tilde{T}$. By monotonicity of Φ_T we conclude that

$$T^1 = \Phi_T(T) \subseteq \Phi_T(\tilde{T}) \subseteq \tilde{T},$$

and, more generally, $T^\alpha \subseteq \tilde{T}$ for any ordinal α. Therefore $\tilde{T} \subseteq \widehat{T}$. $\Box$

As before we define the semantic operator $\mid\!\sim_{\rm AEB}^{\rm query}$ corresponding to the static semantics $\mathsf{STATIC}_{\rm query}(P)$ of a disjunctive program P:

$$P \mid\!\sim_{\rm AEB}^{\rm query} A_1 \mid \ldots \mid A_n \ \text{iff} \ \ A_1 \vee \ldots \vee A_n \in \widehat{\mathfrak{tr}_{\rm AEB}(P)};$$

$$P \mid\!\sim_{\rm AEB}^{\rm query} \text{not } A_1 \mid \ldots \mid \text{not } A_n \ \text{iff} \ \ \mathcal{B}\neg A_1 \vee \ldots \vee \mathcal{B}\neg A_n \in \widehat{\mathfrak{tr}_{\rm AEB}(P)}.$$

The following result gives a simple characterization of the semantic operator $\mathrel{|\!\!\sim}^{\text{query}}_{\text{AEB}}$ which will be used in the next section to show that $\mathrel{|\!\!\sim}^{\text{query}}_{\text{AEB}}$ coincides with D-WFS. The characterization uses the following *simplified belief closure operator*:

$$\begin{aligned}\Theta_T(S) := \mathrm{Cn}(T \cup \{\ \neg\mathcal{B}\neg A_1 \vee \ldots \vee \neg\mathcal{B}\neg A_k &: S \models \quad A_1 \vee \ldots \vee A_k\ \} \\ \cup \{\ \mathcal{B}\neg A_1 \qquad\qquad\qquad &: S \models_{min} \neg A_1 \qquad\qquad \}),\end{aligned} \tag{5}$$

where T and S are arbitrary belief theories in $\mathfrak{L}^{\text{query}}_{\text{AEB}}$, $\mathrm{Cn}(T)$ denotes the set of all *propositional* consequences of T and the A_i are objective atoms. The important step taken here is that we consider Cn and not $\mathrm{Cn}^{\text{query}}_{\text{AEB}}$.

Lemma 27 (Properties of Θ_T) *The simplified belief closure operator $\Theta_T(S)$ has the least fixed point*

$$\overrightarrow{T}$$

obtained as a limit of a non-decreasing sequence of belief theories $\{T^\alpha\}$, where

$$T^0 = T,\ T^{\alpha+1} = \Theta_T(T^\alpha)$$

and $T^\beta = \bigcup_{\alpha<\beta} T^\alpha$, if β is a limit ordinal. For finite theories T the fixed point $\overrightarrow{T}$ is reached after finitely many steps.

PROOF. It is immediate that the operator is monotonic (in the sense that $T^\alpha \subseteq T^{\alpha+1}$): it is entirely in classical logic, belief atoms are just treated as new atom. Thus it has a least fixed point $\overrightarrow{T}$ as described.

Note that the use of minimal model reasoning within the definition of Θ_T does not affect its monotonicity: this is because the notion of a minimal model is defined with respect to *fixed* belief atoms: "$S \models_{min} \neg A_1$" means that *for all assignments of belief atoms* the formula $\neg A_1$ holds.

□

Theorem 28 (Characterization of the Semantic Operator $\mathrel{|\!\!\sim}^{\textbf{query}}_{\textbf{AEB}}$) *The least static expansion $\widehat{\mathfrak{tr}_{\text{AEB}}(P)}$ of a disjunctive logic program P is always consistent and its semantic operator $\mathrel{|\!\!\sim}^{query}_{\text{AEB}}$ can be computed as follows:*

$$\begin{aligned} P \mathrel{|\!\!\sim}^{query}_{\text{AEB}} A_1 \mid \ldots \mid A_n \quad &\textit{iff} \quad A_1 \vee \ldots \vee A_n \in \overrightarrow{\mathfrak{tr}_{\text{AEB}}(P)}; \\ P \mathrel{|\!\!\sim}^{query}_{\text{AEB}} \text{not}\, A_1 \mid \ldots \mid \text{not}\, A_n \quad &\textit{iff} \quad \textit{there is } 1 \le i \le n : \mathcal{B}\neg A_i \in \overrightarrow{\mathfrak{tr}_{\text{AEB}}(P)}. \end{aligned}$$

PROOF. The proof proceeds in two steps.

1. We consider a slightly modified operator Θ'_T:

$$\begin{aligned}\Theta'_T(S) := \mathrm{Cn}(T \cup \{\ \neg\mathcal{B}\neg A_1 \vee \ldots \vee \neg\mathcal{B}\neg A_k \ &: S \models \quad A_1 \vee \ldots \vee A_k \quad \} \\ \cup \{\ \mathcal{B}(\neg A_1 \vee \cdots \vee \neg A_l) \quad &: S \models_{min} \neg A_1 \vee \ldots \vee \neg A_l\ \}),\end{aligned}$$

and show that

$$\widehat{\mathfrak{tr}_{\mathrm{AEB}}(P)} = \mathrm{Cn}^{\mathrm{query}}_{\mathrm{AEB}}(\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)})).$$

We show this equivalence by using the characterization of $\widehat{\mathfrak{tr}_{\mathrm{AEB}}(P)}$ given in Corollary 26. First we note that the notion of a minimal model as used in the definition of Θ'_T (minimal model in propositional logic where belief atoms are viewed as new atoms) corresponds exactly to that of Definition 17 and thus to the one used in Corollary 26.

Note that positive disjunctions true in all models are also true in all minimal models, therefore belief atoms of the form

$$\mathcal{B}(A_1 \vee \ldots \vee A_k),$$

where $S \models A_1 \vee \ldots \vee A_k$ are required to be contained in $\widehat{\mathfrak{tr}_{\mathrm{AEB}}(P)}$. By (**CA**)

$$\neg\mathcal{B}(\neg A_1 \wedge \ldots \wedge \neg A_k)$$

must be in $\widehat{\mathfrak{tr}_{\mathrm{AEB}}(P)}$ and thus, by (**DA**), also

$$\neg\mathcal{B}\neg A_1 \vee \ldots \vee \neg\mathcal{B}\neg A_k.$$

Furthermore, belief atoms of the form

$$\mathcal{B}(\neg A_1 \vee \ldots \vee \neg A_l),$$

where $S \models_{\mathrm{min}} \neg A_1 \vee \ldots \vee \neg A_l$ are required to be contained in $\widehat{\mathfrak{tr}_{\mathrm{AEB}}(P)}$.

This proves that

$$\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)}) \subseteq \widehat{\mathfrak{tr}_{\mathrm{AEB}}(P)}.$$

It is obvious that $\widehat{\mathfrak{tr}_{\mathrm{AEB}}(P)} \subseteq \mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)})$ does **not** hold, because in the construction of $\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)})$ only formulae where belief atoms occur in the form $\mathcal{B}A$ or $\mathcal{B}\neg A$ (where A is an atom) were added. But closing this set under the axioms and the inference rule (**IR**), i.e.

$$\mathrm{Cn}^{\mathrm{query}}_{\mathrm{AEB}}(\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)}))$$

coincides with $\widehat{\mathfrak{tr}_{\mathrm{AEB}}(P)}$. To see this, we apply Corollary 26. Condition 1, 2 as well as the minimality are trivial, so it remains to check Condition 3.

It suffices to show that no new formulae are minimally entailed: for formulae $F \in \mathfrak{L}^{\mathsf{query}}$:

$$\mathrm{Cn}_{\mathrm{AEB}}^{\mathrm{query}}(\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)})) \models_{\min} F \text{ implies } \mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)}) \models_{\min} F.$$

In order to prove this, we have to compare the minimal models of $\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)})$ with the minimal models of $\mathrm{Cn}_{\mathrm{AEB}}^{\mathrm{query}}(\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)}))$. Note that in $\mathrm{Cn}_{\mathrm{AEB}}^{\mathrm{query}}(\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)}))$ much more belief atoms occur, for example those of the form $\mathcal{B}A$ where A is an atom and $\mathcal{B}F$ where F is a positive formula contained in $\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)})$. All these new belief atoms are treated as *new* atoms and therefore a lot more minimal models exist.

In particular any minimal model of $\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)})$ can be extended to a minimal model of $\mathrm{Cn}_{\mathrm{AEB}}^{\mathrm{query}}(\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)}))$ (or to many such minimal models). This is because the closure of $\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)})$ introduces only belief atoms that are consistent with $\mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)})$. For example, the new

$$\neg\mathcal{B}A_1 \vee \ldots \vee \neg\mathcal{B}A_l \text{ -formulae}$$

resulting from $\mathcal{B}(\neg A_1 \vee \ldots \vee \neg A_l)$ are consistent. Otherwise $(\mathcal{B}A_1 \wedge \ldots \wedge \mathcal{B}A_l) \in \mathrm{lfp}(\Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)})$ and therefore there is no minimal model satisfying $\neg A_1 \vee \ldots \vee \neg A_l$ and thus $\mathcal{B}(\neg A_1 \vee \ldots \vee \neg A_l)$ would have never been introduced.

2. We reduce the statement about the computation of $\mid\!\sim_{\mathrm{AEB}}^{\mathrm{query}}$ to **1.** Let us compare Θ'_T with Θ_T. In the construction of $\mathrm{lfp}(\Theta_{\mathfrak{tr}_{\mathrm{AEB}}(P)})$ we only recursively add negative formulae of the special form $\mathcal{B}\neg A$ where A is an atom, whereas for $\mathrm{lfp}(\Theta_{\mathfrak{tr}_{\mathrm{AEB}}(P)})$ we allow pure disjunctions $\mathcal{B}(\neg A_1 \vee \ldots \vee \neg A_l)$. But these formulae $\mathcal{B}(\neg A_1 \vee \ldots \vee \neg A_l)$ do not contribute to the derivation of other pure disjunctions, because of the special form of the theory $\mathfrak{tr}_{\mathrm{AEB}}(P)$: only belief atoms $\mathcal{B}\neg A$ appear. Belief atoms $\mathcal{B}(\neg A_1 \vee \ldots \vee \neg A_l)$ are not related to those, unless $l = 1$. Formally, it is straightforward to prove ($n \in \mathbb{N}$)

$$F \in \Theta'_{\mathfrak{tr}_{\mathrm{AEB}}(P)} \uparrow^n \text{ if and only if } F \in \Theta_{\mathfrak{tr}_{\mathrm{AEB}}(P)} \uparrow^n$$

where F is of the form $\neg\mathcal{B}\neg A_1 \vee \ldots \vee \neg\mathcal{B}\neg A_k$ or $\mathcal{B}\neg A_1 \vee \ldots \vee \mathcal{B}\neg A_k$. Thus we get the same result for the least fixed points of these operators.

Clearly, applying the closure $\mathrm{Cn}_{\mathrm{AEB}}^{\mathrm{query}}$ does not change these formulae either (only belief formulae of the form $\mathcal{B}F$ where F is not an atom or a literal are newly introduced). $\square$

5 The Equivalence of $\mathsf{STATIC}_{\mathrm{query}}$ and D-WFS

In this section we prove that for any disjunctive logic program P, the static semantics $\mathsf{STATIC}_{\mathrm{query}}$, restricted to the query language $\mathfrak{L}_{\mathrm{AEB}}^{\mathrm{query}}$, coincides with the $\mathsf{D\text{-}WFS}$ semantics:

$$\mathsf{STATIC}_{\text{query}}(P) = \mathsf{D\text{-}WFS}(P).$$

First we reformulate the characterization of $\mathsf{D\text{-}WFS}$ given in Theorem 12 in order to bring it closer to the notion of minimal entailment $\models_{min}$ of Definition 17 which is used in Theorem 28.

In order to capture the notion of minimal entailment used in Definition 11 we strengthen the notion of models for a belief theory by requiring that a model has to be NAF-*consistent*.

Definition 29 (Minimal entailment w.r.t. NAF-consistent models) *Let T be a belief theory.*

(i) An interpretation $\mathcal{I}$ is NAF*-consistent for T if for every positive disjunction $A_1 \vee \cdots \vee A_n$,*

$$T \models A_1 \vee \cdots \vee A_n \textit{ implies that for some } A \in \{A_1, \ldots, A_n\}, \mathcal{I} \not\models \mathcal{B}\neg A.$$

(ii) For a formula F, we write $T \models_{\min(N)} F$ iff *F is true in every* NAF-*consistent minimal model of T.*

In the sequel we often denote an interpretation by the set of atomic formulae (both objective and belief atoms) true in it. For example, $\{a, c, \mathcal{B}\neg a, \mathcal{B}\neg b\}$ is not a NAF-consistent model of $T = \{a \vee b, \mathcal{B}\neg a \wedge \mathcal{B}\neg b \rightarrow c\}$ but $\{a, c, \mathcal{B}\neg a\}$ is.

By employing the notion of minimal entailment w.r.t. NAF-consistent models $\mathsf{D\text{-}WFS}$ can be captured using a similar operator as in the characterization of $\mathsf{D\text{-}WFS}$ in Theorem 12.

Definition 30 (SEM_c) *Let P be a disjunctive program and*

$$\begin{aligned} Dis_{\min}(P) = \quad & \{A_1 \mid \cdots \mid A_k \;:\; \mathfrak{tr}_{\text{AEB}}(P) \models A_1 \vee \cdots \vee A_k\} \\ \cup\, & \{\text{not } A \qquad\;\; :\; \mathfrak{tr}_{\text{AEB}}(P) \models_{\min(N)} \neg A\} \end{aligned} \tag{6}$$

We denote by $\mathsf{SEM}_c(P)$ the limit of the monotonically growing series $D'_0, D'_1, \ldots$ where $D'_0 := \emptyset$ and $D'_{i+1} := Dis_{\min}(P/D'_i)$.

This construction gives us exactly $\mathsf{D\text{-}WFS}$.

Theorem 31 $\mathsf{SEM}_0(P) = \mathsf{SEM}_c(P)$.

The theorem can be proved by using Theorem 12 and comparing the original construction of SEM_0 in Definition 11 and Definition 30. To this end, we establish the following two propositions which imply that $Dis(P) = Dis_{\min}(P)$ for any program P.

Proposition 32 *For any set of atoms* $\{A_1, \ldots, A_k\}$,

$$\mathfrak{tr}(P/\mathrm{At}_{\mathfrak{L}}) \models A_1 \vee \cdots \vee A_k \text{ iff } \mathfrak{tr}_{\mathrm{AEB}}(P) \models A_1 \vee \cdots \vee A_k.$$

PROOF. ($\Rightarrow$) Since $P/\mathrm{At}_{\mathfrak{L}} \subseteq P$ and $\mathfrak{tr}(P/\mathrm{At}_{\mathfrak{L}}) = \mathfrak{tr}_{\mathrm{AEB}}(P/\mathrm{At}_{\mathfrak{L}})$, $\mathfrak{tr}(P/\mathrm{At}_{\mathfrak{L}}) \models A_1 \vee \cdots \vee A_k$ implies $\mathfrak{tr}_{\mathrm{AEB}}(P) \models A_1 \vee \cdots \vee A_k$ by the monotonicity of $\models$.

($\Leftarrow$) Assume $\mathfrak{tr}(P/\mathrm{At}_{\mathfrak{L}}) \not\models A_1 \vee \cdots \vee A_k$. Then there is a model M in which $\mathfrak{tr}(P/\mathrm{At}_{\mathfrak{L}})$ is true and $A_1 \vee \cdots \vee A_k$ is false. Choose a minimal model M' of $\mathfrak{tr}(P/\mathrm{At}_{\mathfrak{L}})$ for which $M' \subseteq M$. Now $M' \not\models A_1 \vee \cdots \vee A_k$. Moreover, M' is a model of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ because all belief atoms are false in M' and if $\mathbf{A} \leftarrow \mathbf{B} \in P$, then $\mathbf{A} \leftarrow \mathbf{B} \in P/\mathrm{At}_{\mathfrak{L}}$. Hence, M' is a minimal model of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ and $\mathfrak{tr}_{\mathrm{AEB}}(P) \not\models_{\mathit{min}} A_1 \vee \cdots \vee A_k$ holds. Thus, $\mathfrak{tr}_{\mathrm{AEB}}(P) \not\models A_1 \vee \cdots \vee A_k$ holds. □

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

$$\mathfrak{tr}(P/\mathrm{At}_{\mathfrak{L}}) \models A_1 \vee \cdots \vee A_k \textit{ iff } \mathfrak{tr}_{\mathrm{AEB}}(P) \models_{\mathit{min}} A_1 \vee \cdots \vee A_k$$

holds as well. This means that when defining the operators $Dis(P)$ (Definition 11) and $Dis_{\min}(P)$ (Definition 30) we have at least three alternative ways of choosing the consequence relation for obtaining positive disjunctions.

Proposition 33 *Let P be a disjunctive logic program and ϕ a propositional formula. Then* $\mathfrak{tr}(P/Dis_N) \models_{\mathit{min}} \phi$ *for all* $N \subseteq \mathrm{At}_{\mathfrak{L}}$ *such that* $N \models \mathfrak{tr}(Dis^{+}(P))$ iff $\mathfrak{tr}_{\mathrm{AEB}}(P) \models_{\min(N)} \phi$.

PROOF. ($\Rightarrow$) Assume $\mathfrak{tr}_{\mathrm{AEB}}(P) \not\models_{\min(N)} \phi$. Then there is a NAF-consistent minimal model M of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ such that $M \not\models \phi$. Let $N = \{C \in \mathrm{At}_{\mathfrak{L}} : M \not\models \mathcal{B}\neg C\}$ and let $M' = M \cap \mathrm{At}_{\mathfrak{L}}$. Then $M' \not\models \phi$ and it is easy to establish by Proposition 32 that NAF-consistency of M implies $N \models \mathfrak{tr}(Dis^{+}(P))$. We show that (i) $M' \models \mathfrak{tr}(P/Dis_N)$ and (ii) M' is a minimal model of $\mathfrak{tr}(P/Dis_N)$. This implies that $\mathfrak{tr}(P/Dis_N) \not\models_{\mathit{min}} \phi$ holds.

(i) Let $B_1 \wedge \ldots \wedge B_m \rightarrow A_1 \vee \cdots \vee A_k \in \mathfrak{tr}(P/Dis_N)$. Then there is a rule r of the form

$$A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \mathrm{not}\, C_1 \wedge \cdots \wedge \mathrm{not}\, C_n \tag{7}$$

in P such that $N \cap \{C_1, \ldots, C_n\} = \emptyset$. Hence, $M \models \mathcal{B}\neg C_1 \wedge \cdots \wedge \mathcal{B}\neg C_n$. If $M' \models B_1 \wedge \cdots \wedge B_m$, then $M \models B_1 \wedge \cdots \wedge B_m$ which implies $M \models A_1 \vee \cdots \vee A_k$ since $M \models \mathfrak{tr}_{\mathrm{AEB}}(r)$. But then $M' \models A_1 \vee \cdots \vee A_k$ and we have shown that $M' \models \mathfrak{tr}(P/Dis_N)$ holds.

(ii) Assume that there is a model M'' of $\mathfrak{tr}(P/Dis_N)$ for which $M'' \subset M'$. Now we show that $M''' = M'' \cup \{\mathcal{B}\neg F : M \models \mathcal{B}\neg F\}$ is a model of $\mathfrak{tr}_{\mathrm{AEB}}(P)$. Let $r \in P$ be of the form (7). If $M''' \models B_1 \wedge \cdots \wedge B_m \wedge \mathcal{B}\neg C_1 \wedge \cdots \wedge \mathcal{B}\neg C_n$, then $N \cap \{C_1, \ldots, C_n\} = \emptyset$, $A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m \in P/Dis_N$ and $M'' \models B_1 \wedge \cdots \wedge B_m$. This implies $M'' \models A_1 \vee \cdots \vee A_k$ and hence $M''' \models A_1 \vee \cdots \vee A_k$. Thus, $M''' \models \mathfrak{tr}_{\mathrm{AEB}}(P)$. This contradicts the assumption that M is a minimal model of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ because $M''' \cap \mathrm{At}_{\mathfrak{L}} = M'' \subset M' = M \cap \mathrm{At}_{\mathfrak{L}}$ holds but $\{\mathcal{B}F : M''' \models \mathcal{B}F\} = \{\mathcal{B}F : M \models \mathcal{B}F\}$.

($\Leftarrow$) Assume $\mathfrak{tr}(P/Dis_N) \not\models_{min} \phi$ for some $N \subseteq \mathrm{At}_{\mathfrak{L}}$ such that $N \models \mathfrak{tr}(Dis^+(P))$. Then there is a minimal model M of $\mathfrak{tr}(P/Dis_N)$ such that $M \not\models \phi$. Let $M' = M \cup \{\mathcal{B}\neg C : C \in \mathrm{At}_{\mathfrak{L}} - N\}$. Then $M' \not\models \phi$ and it is straightforward to establish by Proposition 32 that M' is NAF-consistent for $\mathfrak{tr}_{\mathrm{AEB}}(P)$. We show that (i) $M' \models \mathfrak{tr}_{\mathrm{AEB}}(P)$ and (ii) M' is a minimal model of $\mathfrak{tr}_{\mathrm{AEB}}(P)$. This implies that $\mathfrak{tr}_{\mathrm{AEB}}(P) \not\models_{\min(N)} \phi$ holds.

(i) Let $r \in P$ be of the form (7) and $M' \models B_1 \wedge \cdots \wedge B_m \wedge \mathcal{B}\neg C_1 \wedge \cdots \wedge \mathcal{B}\neg C_n$. Then $N \cap \{C_1, \ldots, C_n\} = \emptyset$, $A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m \in P/Dis_N$ and $M \models B_1 \wedge \cdots \wedge B_m$. Now $M \models A_1 \vee \cdots \vee A_k$ as $M \models \mathfrak{tr}(P/Dis_N)$. This implies $M' \models A_1 \vee \cdots \vee A_k$ and, hence, $M' \models \mathfrak{tr}_{\mathrm{AEB}}(P)$.

(ii) Assume that there is a model M'' of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ for which $M'' \cap \mathrm{At}_{\mathfrak{L}} \subset M' \cap \mathrm{At}_{\mathfrak{L}}$ but $\{\mathcal{B}F : M'' \models \mathcal{B}F\} = \{\mathcal{B}F : M' \models \mathcal{B}F\}$. We show that $M''' = M'' \cap \mathrm{At}_{\mathfrak{L}}$ is a model of $\mathfrak{tr}(P/Dis_N)$. Consider a rule r of the form (7) such that $A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m \in P/Dis_N$. If $M''' \models B_1 \wedge \cdots \wedge B_m$, then $M'' \models B_1 \wedge \cdots \wedge B_m$ which implies that $M'' \models A_1 \vee \cdots \vee A_k$. But then $M''' \models A_1 \vee \cdots \vee A_k$ and $M''' \models \mathfrak{tr}(P/Dis_N)$. However, this contradicts the assumption that M is a minimal model of $\mathfrak{tr}(P/Dis_N)$ because $M''' = M'' \cap \mathrm{At}_{\mathfrak{L}} \subset M' \cap \mathrm{At}_{\mathfrak{L}} = M$. □

Theorem 34 ($\mathsf{STATIC}_{\mathbf{query}} = \mathsf{D\text{-}WFS}$)
STATIC_{query} coincides with $\mathsf{D\text{-}WFS}$.

PROOF. We have to compare the construction of D-WFS based on $\models_{\min(N)}$ (Theorem 31) and the construction of STATIC $_{\mathrm{query}}$ based on Theorem 28.

This means we have to *(a)* consider Equation (6) and *(b)* compare it with the least fixed point of the simplified belief closure operator given in Equation (5).

While in *(a)* every step in the construction is split into deriving certain disjunctions and doing certain program transformations (to prepare the next step), in *(b)* every step is to apply an operator and getting larger and larger theories.

To make this precise, we show the following by induction on $i \geq 0$:

$$\begin{array}{lrcl} \text{Case 1: } & \neg\mathcal{B}\neg A_1 \vee \ldots \vee \neg\mathcal{B}\neg A_n \in \Theta_{\mathtt{tr}_{\mathrm{AEB}}(P)}\uparrow^i & \text{iff} & A_1 \mid \ldots \mid A_n \in D'_i \\ \text{Case 2: } & \mathcal{B}\neg A \in \Theta_{\mathtt{tr}_{\mathrm{AEB}}(P)}\uparrow^i & \text{iff} & \text{not}\, A \in D'_i. \end{array}$$

Note that although in Equation (6), $\mathtt{tr}_{\mathrm{AEB}}(P)$ is explicitly contained right from the beginning and in Equation (5) the set S contains $\mathtt{tr}_{\mathrm{AEB}}(P)$ only from the second iteration on, our definitions are such that $D'_0 = \emptyset$ and thus D'_i corresponds to $\Theta_{\mathtt{tr}_{\mathrm{AEB}}(P)}\uparrow^i$.

For $i = 0, 1$, Case 1 is trivial. To establish Case 2, we have to compare the notion $\models_{\min(N)}$ with $\models_{min}$. At first sight it seems that $\models_{\min(N)}$ is stronger than $\models_{min}$ because not all minimal models are checked but only those of a special kind (NAF-consistent). But in the construction of the simplified belief operator in Equation (5) the theory T is the translation $\mathtt{tr}_{\mathrm{AEB}}(P)$ of a logic program and thus contains belief atoms of the form $\mathcal{B}\neg A$ only in bodies of rules. Thus it can not happen that no NAF-consistent models exist (this would be the case if T contains both $A \vee B$ and $\mathcal{B}\neg A \wedge \mathcal{B}\neg B$). Moreover, the addition of the disjunctions

$$\neg\mathcal{B}\neg A_1 \vee \ldots \vee \neg\mathcal{B}\neg A_k$$

in Equation (5) ensures that all models of the least fixed point of this operator are NAF-consistent (compare with condition *(i)* in Definition 29). NAF-consistent models are nothing else than arbitrary minimal models (as used in Equation (5)) satisfying the additional constraints formalized by the belief atoms $\neg\mathcal{B}\neg A_1 \vee \ldots \vee \neg\mathcal{B}\neg A_k$ and $\mathcal{B}\neg A_1$. To be more precise,

$$\Theta_{\mathtt{tr}_{\mathrm{AEB}}(P)}\uparrow^i \models_{min} \mathcal{B}\neg A \iff \text{not}\, A \in D'_i.$$

To show our statement for i given $i-1$ is completely analogous. The program reduction P/D_{i-1} (from Definition 9) applied in Definition 30 is simulated in the simplified belief operator by applying Cn and using the belief atoms that have been added in the last round. To be more precise, the first transformation in Definition 9 is captured by adding $\mathcal{B}\neg A_1$ in the definition of the operator Θ_T. The second transformation in Definition 9 is simulated by adding $\neg\mathcal{B}\neg A_1 \vee$

$\ldots \vee \neg\mathcal{B}\neg A_k$. The result then follows because the disjunctions added in round i are, by induction hypothesis, the same for both constructions. $\square$

6 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 $\mathsf{STATIC}_{\text{query}}$) 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 31. Then we discuss how the required notion of minimal entailment $\models_{\min(N)}$ can be implemented using a new tableau method developed for circumscription [27,26,2]. 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 SEM_c.

$$\begin{aligned}\mathrm{R_D}(P) = \{ & \mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\,\mathbf{C}' : \mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\,\mathbf{C} \in P,\\ & \mathfrak{tr}(P/\mathrm{At}_{\mathfrak{L}}) \not\models C_1 \vee \cdots \vee C_n \text{ where}\\ & \mathbf{C} = \{C_1, \ldots, C_n\},\\ & \mathbf{C}' = \{C \in \mathbf{C} : \mathfrak{tr}_{\mathrm{AEB}}(P) \not\models_{\min(N)} \neg C\}\}\end{aligned}$$

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 P^* is obtained by

- removing each rule such that $\mathrm{not}\,C_1, \ldots, \mathrm{not}\,C_n$ 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 $\mathrm{not}\,C$ for which $\neg C$ is minimally entailed by the remaining program

until no reduction is possible. Then we have the following theorem.

Theorem 35 $\mathit{SEM}_c(P) = \mathit{Dis}_{\min}(P^*)$.

PROOF. Let $D_0 = \emptyset$ and $D_{i+1} = \mathit{Dis}_{\min}(P/D_i)$. We show that for all $i = 0, 1, \ldots,$

$$P_i = P/D_i \tag{8}$$

holds by induction on i. This implies $\mathsf{SEM}_c(P) = \mathit{Dis}_{\min}(P^*)$.

For $i = 0$, clearly (8) holds. Let $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}' \in P_{i+1}$. Then there is $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C} \in P$ such that $\mathfrak{tr}(P_i/\mathrm{At}_{\mathfrak{L}}) \not\models C_1 \vee \cdots \vee C_n$ and $\mathbf{C}' = \{C \in \mathbf{C} : \mathfrak{tr}_{\mathrm{AEB}}(P_i) \not\models_{\min(N)} \neg C\}$. We show that $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}' \in P/D_{i+1}$ by establishing that (i) for all subsets $\{C'_1, \ldots, C'_p\}$ of $\mathbf{C}$, $C'_1 \mid \cdots \mid C'_p \notin D_{i+1}$ and (ii) for all $C \in \mathbf{C}$, $C \in \mathbf{C}'$ *iff* $\mathrm{not}\, C \notin D_{i+1}$.

(i) Let $\{C'_1, \ldots, C'_p\} \subseteq \mathbf{C}$. Then $\mathfrak{tr}(P_i/\mathrm{At}_{\mathfrak{L}}) \not\models C'_1 \vee \cdots \vee C'_p$ as $\mathfrak{tr}(P_i/\mathrm{At}_{\mathfrak{L}}) \not\models C_1 \vee \cdots \vee C_n$. By the inductive hypothesis, $P_i = P/D_i$ and, hence, $C'_1 \mid \cdots \mid C'_p \notin D_{i+1} = \mathit{Dis}_{\min}(P/D_i)$.
(ii) For all $C \in \mathbf{C}$, $C \in \mathbf{C}'$ *iff* $\mathfrak{tr}_{\mathrm{AEB}}(P_i) \not\models_{\min(N)} \neg C$ *iff* $\mathfrak{tr}_{\mathrm{AEB}}(P/D_i) \not\models_{\min(N)} \neg C$ (by the inductive hypothesis) *iff* $\mathrm{not}\, C \notin D_{i+1}$.

Similarly, it can be shown that $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}' \in P/D_{i+1}$ implies $\mathbf{A} \leftarrow \mathbf{B} \wedge \mathrm{not}\, \mathbf{C}' \in P_{i+1}$. Hence, (8) holds. □

The theorem combined with Proposition 32 implies that a positive query can be answered using the rules of the reduced program that do not contain negative body literals and a classical theorem prover but for negative queries minimal model reasoning is needed.

Corollary 36 *Let P be a disjunctive program and P^* the corresponding reduced program. Then*

(i) $A_1 \mid \cdots \mid A_k \in \mathit{D\text{-}WFS}(P)$ iff $\mathfrak{tr}(P^*/\mathrm{At}_{\mathfrak{L}}) \models A_1 \vee \cdots \vee A_k$ *and*
(ii) $\mathrm{not}\, A \in \mathit{D\text{-}WFS}(P)$ iff $\mathfrak{tr}_{\mathrm{AEB}}(P^*) \models_{\min(N)} \neg A$.

The series of reductions is straightforward to implement [9] if decision procedures for classical entailment and minimal entailment $\models_{\min(N)}$ are available. The latter procedure can be implemented using a new tableau method for circumscription presented in [27,26]. Next we explain briefly the main ideas of the method.

[9] See `http://www.uni-koblenz.de/ag-ki/DLP/systems/dwfs-mm.html`.

A standard tableau method deciding whether a query follows (classically) from a set of premises can be seen as a systematic procedure for constructing a counter-model, i.e., a model where the premises are true and the query is 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 a set of clauses P

$$\begin{aligned} &a_1 \vee b_1, \ldots, a_n \vee b_n, \\ &a_0 \rightarrow a_1, \ldots, a_{n-1} \rightarrow a_n \end{aligned} \tag{9}$$

and the task of deciding whether $\neg a_0$ is minimally entailed by P. A tableau method like that in [27,26] 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 [27,26] 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.

Proposition 37 (Minimal models [26]) *Let P be a disjunctive program. A model $\mathcal{I}$ of $\mathbf{tr}_{\mathrm{AEB}}(P)$ is a minimal one* iff

$$\mathbf{tr}_{\mathrm{AEB}}(P) \cup \mathrm{N}(\mathcal{I}) \models \{A \in \mathrm{At}_{\mathfrak{L}} : \mathcal{I} \models A\} \tag{10}$$

where

$$\mathrm{N}(\mathcal{I}) = \{\neg A : A \in \mathrm{At}_{\mathfrak{L}}, \mathcal{I} \not\models A\} \cup \{\mathcal{B}F : \mathcal{I} \models \mathcal{B}F\}.$$

For example, given P in (9) 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 $\models_{\min(N)}$ 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.

Proposition 38 (Determining NAF-consistency) *Let P be a disjunctive program. A model $\mathcal{I}$ of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ is* NAF-*consistent for* $\mathfrak{tr}_{\mathrm{AEB}}(P)$ iff

$$\mathfrak{tr}_{\mathrm{AEB}}(P) \cup \{\neg C : C \in \mathrm{At}_{\mathfrak{L}}, \mathcal{I} \models \mathcal{B}\neg C\} \text{ has a model.} \tag{11}$$

PROOF. ($\Rightarrow$) Let $\mathcal{I}$ be a NAF-consistent model of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ but assume that (11) does not hold. By compactness there is a finite subset $\{\neg C_1, \ldots, \neg C_m\} \subseteq \{\neg C : C \in \mathrm{At}_{\mathfrak{L}}, \mathcal{I} \models \mathcal{B}\neg C\}$ for which no model exists. Hence, $\mathfrak{tr}_{\mathrm{AEB}}(P) \models C_1 \vee \cdots \vee C_m$ holds. Then by the NAF-consistency of $\mathcal{I}$, for some $C \in \{C_1, \ldots, C_m\}$, $\mathcal{I} \not\models \mathcal{B}\neg C$, a contradiction. Thus (11) holds.

($\Leftarrow$) Let (11) hold and assume that for some $A_1 \vee \cdots \vee A_n$, $\mathfrak{tr}_{\mathrm{AEB}}(P) \models A_1 \vee \cdots \vee A_n$ but for all $A \in \{A_1, \ldots, A_n\}, \mathcal{I} \models \mathcal{B}\neg A$. Then $\{\neg A_1, \ldots, \neg A_n\} \subseteq \{\neg C : C \in \mathrm{At}_{\mathfrak{L}}, \mathcal{I} \models \mathcal{B}\neg C\}$ and, hence, $\mathfrak{tr}_{\mathrm{AEB}}(P) \not\models A_1 \vee \cdots \vee A_n$, a contradiction. Thus, $\mathcal{I}$ is NAF-consistent for $\mathfrak{tr}_{\mathrm{AEB}}(P)$. $\square$

Hence, a model of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ is a NAF-consistent minimal model of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ *iff* the conditions (10) and (11) hold for the model.

Example 39 *Consider the program P*

$$\begin{array}{ll} a \mid b \leftarrow & \\ c & \leftarrow \mathrm{not}\, a \\ d & \leftarrow \mathrm{not}\, b \\ e & \leftarrow c \wedge d \\ f & \leftarrow \mathrm{not}\, e \\ f & \leftarrow \mathrm{not}\, a \wedge \mathrm{not}\, b \wedge \mathrm{not}\, c \end{array}$$

and models $M_1 = \{a, \mathcal{B}\neg b, c, d, e\}$ and $M_2 = \{a, \mathcal{B}\neg a, \mathcal{B}\neg b, c, d, e\}$ of $\mathfrak{tr}_{\mathrm{AEB}}(P)$. For M_1 the condition (10) fails as $\mathrm{N}(M_1) = \{\neg b, \neg f, \mathcal{B}\neg b\}$ and, e.g., $\mathfrak{tr}_{\mathrm{AEB}}(P) \cup \mathrm{N}(M_1) \not\models c$. For M_2, $\mathrm{N}(M_2) = \{\neg b, \neg f, \mathcal{B}\neg a, \mathcal{B}\neg b\}$ and the condition (10) holds but (11) does not as $\mathfrak{tr}_{\mathrm{AEB}}(P) \models a \vee b$. In fact, there are no NAF-*consistent minimal models of $\mathfrak{tr}_{\mathrm{AEB}}(P)$ where e is true and, hence, $\mathfrak{tr}_{\mathrm{AEB}}(P) \models_{\min(N)} \neg e$. 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 $\models_{\min(N)}$ can 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.

7 Related Work

Several systems for disjunctive logic programming are already either implemented or currently under development. Besides systems for computing the classical disjunctive stable semantics ([17,29,18]), D. Seipel ([33]) implemented a system (see `http://www-info1.informatik.uni-wuerzburg.de/database/DisLog/introduction.html`) for computing various variants of the disjunctive semantics discussed in [22].

A group in Vienna (T. Eiter, G. Gottlob, N. Leone) is currently building a system called `dlv` [13]. 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) ([1], see also `http://www.uni-koblenz.de/ag-ki/DLP/`). In contrast to the other approaches mentioned above, it aims at extending certain theorem-proving calculi (*restart model elimination* and *hyper tableau calculi*) for disjunctive logic programming.

Finally, we refer the reader to [11], which contains a description of several non-monotonic reasoning systems (including those handling disjunction) and to the following web page which is actively maintained and contains current information on various logic programming systems and their non-monotonic aspects (different kinds of negation, disjunction, abduction etc.): `http://www.uni-koblenz.de/ag-ki/LP/`.

Semantics	**Construction**	**Closure**	**Theorem**
STATIC: $\mid\!\sim_{\mathrm{AEB}}$	$\mathrm{lfp}(\Psi_{\mathtt{tr}_{\mathrm{AEB}}(P)})$	$\mathrm{Cn}_{\mathrm{AEB}}$	Theorem 19
STATIC: $\mid\!\sim^{\mathrm{query}}_{\mathrm{AEB}}$	$\mathrm{lfp}(\Phi_{\mathtt{tr}_{\mathrm{AEB}}(P)})$	$\mathrm{Cn}^{\mathrm{query}}_{\mathrm{AEB}}$	Theorem 25
	$\mathrm{lfp}(\Theta_{\mathtt{tr}_{\mathrm{AEB}}(P)})$	Cn	Theorem 28
D-WFS: $\mid\!\sim_{\textsf{D-WFS}}$	chain D'_i	$\mathrm{Cn}^{\mathrm{query}}$	Theorem 31
	chain D_i	Cn	Theorem 12
	smallest SEM wrt transformations		Definition 6

Table 1
Proving the Equivalence of $\mathsf{STATIC}_{\mathrm{query}}$ and D-WFS

8 Concluding Remarks

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 logic 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 [30] and the D-WFS semantics proposed by Brass and Dix [3,7], seems to significantly enhance our understanding of the theoretical and algorithmic aspects of disjunctive logic programming.

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. Specifically, the main result of this paper shows that, when restricted to a common query language, the two semantics become entirely *equivalent*:

$$\mathsf{STATIC}_{\mathrm{query}} \equiv \textsf{D-WFS} .$$

This fundamental result uses an elegant and powerful characterization of $\mathsf{STATIC}_{\mathrm{query}}$, the static semantics restricted to the query language $\mathfrak{L}^{\mathrm{query}}_{\mathrm{AEB}}$.

Table 1 illustrates the overall idea of the proof of the equivalence between $\mathsf{STATIC}_{\mathrm{query}}$ and D-WFS. The first line contains information about the full STATIC semantics. The last line describes the original definition of D-WFS. While the second and third line give characterizations of $\mathsf{STATIC}_{\mathrm{query}}$ based on least fixed points of certain operators, the fifth and fourth give characterizations of D-WFS in terms of certain chains of pure disjunctions. Our main Theorem 34 builds on the third line ($\mathrm{lfp}(\theta_{\mathtt{tr}_{\mathrm{AEB}}(P)})$) and the fourth line (chain D'_i) to show the overall equivalence.

We demonstrated the applicability and importance of the above results by using the underlying minimal model reasoning procedure for an efficient implementation of the D-WFS and static semantics that has been carried out in the DisLoP-project [10] at the University of Koblenz.

Acknowledgements

We thank Alexander Bochman and several anonymous reviewers for their useful comments on parts of the paper.

References

[1] Chandrabose Aravindan, Jürgen Dix, and Ilkka Niemelä. Dislop: A research project on disjunctive logic programming. *AI Communications*, 10(3/4):151–165, 1997.

[2] 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.

[3] 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.

[4] Stefan Brass and Jürgen Dix. A General Approach to Bottom-Up Computation of Disjunctive Semantics. In J. Dix, L. Pereira, and T. Przymusinski, editors, *Nonmonotonic Extensions of Logic Programming*, LNAI 927, pages 127–155. Springer, Berlin, 1995.

[5] 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.).

[6] 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.).

[10] see `http://www.uni-koblenz.de/ag-ki/DLP/index.html`

[7] Stefan Brass and Jürgen Dix. Semantics of (Disjunctive) Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, 38(3):167–213, 1999. (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.).

[8] Stefan Brass, Jürgen Dix, and Teodor Przymusinski. Computation of the semantics of autoepistemic belief theories. *Artificial Intelligence*, 112(1-2), 1999.

[9] 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.

[10] Gerhard Brewka and Jürgen Dix. Knowledge representation with extended logic programs. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition, Volume 6, Methodologies*, chapter 6. Reidel Publ., 2000. Shortened version also appeared in Dix, Pereira, Przymusinski (Eds.), *Logic Programming and Knowledge Representation*, Springer LNAI 1471, pages 1–55, 1998.

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

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] Michael Gelfond, Halina Przymusinska, and Teodor Przymusinski. On the Relationship between Circumscription and Negation as Failure. *Artificial Intelligence*, 38:75–94, 1989.

[20] Michael Gelfond, Halina Przymusinska, and Teodor Przymusinski. On the Relationship between CWA, minimal model and minimal Herbrand model Semantics. *International Journal of Intelligent Systems*, 5(5):549–564, 1990.

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

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

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] Teodor Przymusinski. Well-founded Semantics Coincides With Three-Valued Stable Semantics. *Fundamenta Informaticae*, XIII:445–463, 1990.

[29] Teodor Przymusinski. Well-founded completions of logic programs. In *Proceedings of the Eigth International Logic Programming Conference, Paris, France*, pages 726–744, Cambridge, Mass., July 1991. MIT Press.

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

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

[32] 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.

[33] 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.

[34] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38:620–650, 1991.

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