A Comparative Study of Well-founded Semantics for Disjunctive Logic Programs

Kewen Wang*

Institut für Informatik, Universität Potsdam
Postfach 60 15 53, D–14415 Potsdam, Germany
kewen@cs.uni-potsdam.de

Abstract. Much work has been done on extending the well-founded semantics to general disjunctive logic programs and various approaches have been proposed. However, no consensus has been reached about which semantics is the most intended. In this paper we look at disjunctive well-founded reasoning from different angles. We show that there is an intuitive form of the well-founded reasoning in disjunctive logic programming which can be equivalently characterized by several different approaches including program transformations, argumentation, unfounded sets (and resolution-like procedure). We also provide a bottom-up procedure for this semantics. The significance of this work is not only in clarifying the relationship among different approaches, but also in providing novel arguments in favor of our semantics.

1 Introduction

The importance of representing and reasoning about disjunctive information has been addressed by many researchers. Disjunctive logic programming (DLP) is widely believed to be a suitable tool for formalizing disjunctive reasoning and it has received extensive study in recent years. Since DLP admits both default negation and disjunction, the issue of finding a suitable semantics for disjunctive programs is more difficult than it is in the case of normal (i. e. non-disjunctive) logic programs. Usually, skepticism and credulism represent two major semantic intuitions for knowledge representation in artificial intelligence. The well-founded semantics [12] is a formalism of skeptical reasoning in normal logic programming while the stable semantics [6] formalizes credulous reasoning. Recently, considerable effort has been paid to generalize these two semantics to disjunctive logic programs. However, the task of generalizing the well-founded model to disjunctive programs has proven to be complex. There have been various proposals for defining the well-founded semantics for general disjunctive logic programs [8]. As argued by some authors (for instance [2, 10, 13]), each of the previous versions of the disjunctive well-founded semantics bears its own drawbacks. Moreover, no consensus has been reached about what constitutes an intended well-founded semantics for disjunctive logic programs. The semantics D-WFS [1, 2], STATIC [10] and WFDS [13] are among the most recent approaches to defining disjunctive well-founded semantics. D-WFS is based on a series of abstract properties and it is the weakest (least)

* On leave from Tsinghua University, Beijing.

semantics that is invariant under a set of program transformations. STATIC has its root in autoepistemic logic and is based on the notion of *static expansions* for belief theories. The semantics STATIC(P) for a disjunctive program P is defined as the least static expansion of P_{AEB} where P_{AEB} is the belief theory corresponding to P. The basic idea of WFDS is to transform P into an argumentation framework and WFDS(P) is specified by the least acceptable hypothesis of P. Although these semantics stem from very different intuitions, all of them share a number of attractive properties. In particular, each of these semantics extends both the well-founded semantics [12] for normal logic programs and the generalized closed world assumption (GCWA) [9] for positive disjunctive programs (i. e. without default negation).

It has been proven that D-WFS is equivalent to a restricted version of STATIC [3]. But the relation of these semantics to the argumentation-based semantics and unfounded sets are as yet unclear. In this paper, we modify some existing semantics to make them more intuitive and report further equivalence results. First, we define a transformation-based semantics denoted D-WFS* by introducing a new transformation into Brass and Dix's set $\mathbf{T}_{\text{WFS}}$ of program transformations. This semantics naturally extends D-WFS and enjoys all the important properties that have been proven for D-WFS. We prove that WFDS is equivalent to D-WFS*. We also provide a bottom-up evaluation procedure for WFDS (and D-WFS*). Second, we define a new notion of unfounded sets which is a generalization of the unfounded sets defined in [7, 5]. Based on this new notion of unfounded sets, we define a well-founded semantics U-WFS for disjunctive programs. We show that U-WFS is equivalent to WFDS (and thus D-WFS*). Moreover, in [14] we have developed a top-down procedure D-SLS Resolution which is sound and complete with respect to our semantics. D-SLS extends both SLS-resolution [11] and SLI-resolution [8]. Altogether we obtain the following equivalence results:

$$\text{WFDS} \equiv \text{D-WFS}^* \equiv \text{U-WFS} \equiv \text{D-SLS}.$$

We consider these results to be quite significant: (1) Our results clarify the relationship among quite several different approaches to defining disjunctive well-founded semantics, including argumentation-based, transformation-based, unfounded sets-based and resolution-based approaches. (2) Since the four semantics are based on very different intuitions, these equivalent characterizations in turn provide yet more powerful arguments in favor of our semantics. (3) Both the top-down procedure D-SLS Resolution [14] and the bottom-up query evaluation proposed in this paper pave two different ways for implementing our semantics.

The rest of this paper is arranged as follows. In Section 2 we recall some basic definitions and notation; we present in Section 3 a slightly restricted form of the well-founded semantics WFDS. In Section 4 we introduce a new program transformation *Head reduction* and then define the transformation-based semantics D-WFS*, which naturally extends D-WFS. In Section 5, we first provide a bottom-up query evaluation for D-WFS* (and WFDS) and then prove the equivalence of D-WFS* and WFDS. Section 6 introduces the new notion of unfounded sets and defines the well-founded semantics U-WFS. We also show that U-WFS is equivalent to WFDS. Section 7 is our conclusion. Proofs of the theorems are given in the full version of this paper.

2 Preliminaries

We briefly review most of the basic notions used throughout this paper.

A *disjunctive logic program* is a finite set of rules of the form

$$a_1 \vee \cdots \vee a_n \leftarrow b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_t, \tag{1}$$

where a_i, b_i, c_i are atoms and $n > 0$. The default negation '$not\ a$' of an atom a is called a *negative literal*. In this paper we consider only propositional programs although many definitions and results hold for predicate logic programs.

P is a *normal* logic program if it contains no disjunctions.

If a rule of form (1) contains no negative body literals, it is called *positive*; P is a positive program if every rule of P is positive.

If a rule of form (1) contains no body atoms, it is called *negative*; P is a negative program if every rule of P is negative.

Following [2], we also say a negative rule r is a *conditional fact*. That is, a conditional fact is of form $a_1 \vee \cdots \vee a_n \leftarrow not\ c_1, \cdots, not\ c_m$, where a_i and c_j are (ground) atoms for $1 \leq k \leq n$ and $0 \leq j \leq m$.

For a rule r of form (1), $body(r) = body^+(r) \cup body^-(r)$ where $body^+(r) = \{b_1, \ldots, b_m\}$ and $body^-(r) = \{not\ c_1, \ldots, not\ c_t\}$; $head(r) = a_1 \vee \cdots \vee a_n$. When no confusion is caused, we also use $head(r)$ to denote the set of atoms in $head(r)$. For instance, $a \in head(r)$ means that a appears in the head of r. If X is a set of atoms, $head(r) - X$ is the disjunction obtained from $head(r)$ by deleting the atoms in X. The set $head(P)$ consists of all atoms appearing in rule heads of P.

As usual, B_P is the *Herbrand base* of disjunctive logic program P, that is, the set of all (ground) atoms in P. A *positive* (*negative*) disjunction is a disjunction of atoms (negative literals) in P. A *pure disjunction* is either a positive one or a negative one. The disjunctive base of P is $\mathrm{DB}_P = \mathrm{DB}_P^+ \cup \mathrm{DB}_P^-$ where DB_P^+ is the set of all positive disjunctions in P and DB_P^- is the set of all negative disjunctions in P. If A and $B = A \vee A'$ are two disjunctions, then we say A is a *sub-disjunction* of B, denoted $A \subseteq B$.

A *model state* of a disjunctive program P is a subset of DB_P. Usually, a well-founded semantics for a disjunctive logic program is defined by a model state.

If S is an expression (a set of literals, a disjunction or a set of disjunctions), $atoms(S)$ denotes the set of all atoms appearing in S.

For simplicity, we assume that all model states are closed under implication of pure disjunctions. That is, for any model state S, if A is a sub-disjunction of a pure disjunction B and $A \in S$, then $B \in S$. For instance, if $S = \{a, b \vee c\}$, then $a \vee b \vee c \in S$.

Given a model state S and a pure disjunction A, we also say A is satisfied by S, denoted $S \models A$, if $A \in S$.

We assume that all disjunctions have been simplified by deleting the repeated literals. For example, the disjunction $a \vee b \vee b$ is actually the disjunction $a \vee b$.

3 Argumentation and well-founded semantics

As illustrated in [13][1], argumentation provides an unifying semantic framework for DLP. The basic idea of the argumentation-based approach for DLP is to translate each disjunctive logic program into an argument framework $\mathbf{F}_P = \langle P, \mathrm{DB}_P^-, \rightsquigarrow_P \rangle$. Here, an *assumption* of P is a negative disjunction of P, and a *hypothesis* is a set of assumptions; $\rightsquigarrow_P$ is an attack relation among the hypotheses. An *admissible hypothesis* Δ is one that can attack every hypothesis which attacks it. The intuitive meaning of an assumption $not\ a_1 \vee \cdots \vee not\ a_m$ is that $a_1 \wedge \cdots \wedge a_m$ can not be proved from the disjunctive program.

Given a hypothesis Δ of disjunctive program P, similar to the GL-transformation [6], we can easily reduce P into another disjunctive program without default negation.

Definition 1. *Let Δ be a hypothesis of disjunctive program P, then the reduct of P with respect to Δ is the disjunctive program*

$$P_\Delta^+ = \{head(r) \leftarrow body^+(r) \mid r \in P \text{ and } body^-(r) \subseteq \Delta\}.$$

The following definition introduces a special resolution $\vdash_P$ which resolves default-negation literals with a disjunction.

Definition 2. *Let Δ be a hypothesis of disjunctive program P and $A \in DB_P^+$. If there exists $B \in DB_P^+$ and $not\ b_1, \ldots, not\ b_m \in \Delta$ such that $B = A \vee b_1 \vee \cdots \vee b_m$ and $P_\Delta^+ \models B$. Then Δ is said to be a* supporting hypothesis *for A, denoted $\Delta \vdash_P A$. Here $\models$ is the inference relation of the classical propositional logic.*

The set of all positive disjunctions supported by Δ is denoted:

$$cons_P(\Delta) = \{A \in DB_P^+ \mid \Delta \vdash_P A\}.$$

To derive suitable hypotheses for a given disjunctive program, some constraints will be required to filter out unintuitive hypotheses.

Definition 3. *Let Δ and Δ' be two hypotheses of disjunctive program P. If at least one of the following two conditions holds:*

1. there exists $\beta = not\ b_1 \vee \cdots \vee not\ b_m \in \Delta',\ m > 0$, such that $\Delta \vdash_P b_i$, for all $i = 1, \ldots, m$; or

2. there exist $not\ b_1, \ldots, not\ b_m \in \Delta', m > 0$, such that $\Delta \vdash_P b_1 \vee \cdots \vee b_m$,
then we say Δ attacks *Δ', and denoted $\Delta \rightsquigarrow_P \Delta'$.*

Intuitively, $\Delta \rightsquigarrow_P \Delta'$ means that Δ causes a direct contradiction with Δ' and the contradiction may come from one of the above two cases.

Example 1.

$$\begin{array}{l} a \vee b \leftarrow \\ \quad c \leftarrow d, not\ a, not\ b \\ \quad d \leftarrow \\ \quad e \leftarrow not\ e \end{array}$$

[1] You et al in [15] also define an argumentative extension to the disjunctive stable semantics but that framework does not lead to an intuitive well-founded semantics for DLP as the authors have observed.

Let $\Delta' = \{not\ c\}$ and $\Delta = \{not\ a, not\ b\}$, then $\Delta \rightsquigarrow_P \Delta'$.

The next definition specifies what is an acceptable hypothesis.

Definition 4. *Let Δ be a hypothesis of disjunctive program P. An assumption B of P is* admissible *with respect to Δ if $\Delta \rightsquigarrow_P \Delta'$ holds for any hypothesis Δ' of P such that $\Delta' \rightsquigarrow_P \{B\}$.*

Denote $\mathcal{A}_P(\Delta) = \{not\ a_1 \vee \cdots \vee not\ a_m \in DB_P^- \mid not\ a_i$ is admissible wrt Δ for some $i, 1 \leq i \leq m\}$.

Originally, $\mathcal{A}_P$ also includes some other negative disjunctions. To compare with different semantics, we omit them here. Another reason for doing this is that information in form of negative disjunctions does not participate in inferring positive information in DLP.

For any disjunctive program P, $\mathcal{A}_P$ is a monotonic operator. Thus, if P is finite then $\mathcal{A}_P$ has the least fixpoint $\mathrm{lfp}(\mathcal{A}_P)$ and $\mathrm{lfp}(\mathcal{A}_P) = \mathcal{A}_P^k(\emptyset)$ for some $k \geq 0$.

Definition 5. *The* well-founded disjunctive hypothesis *WFDH(P) of disjunctive program P is defined as the least fixpoint of the operator $\mathcal{A}_P$. That is, WFDH$(P) = \mathcal{A}_P \uparrow \omega$.*

The well-founded disjunctive semantics *WFDS for P is defined as the model state WFDS(P) = WFDH$(P) \cup cons_P$(WFDH(P)).*

By the above definition, WFDS(P) is uniquely determined by WFDH(P).

For the disjunctive program P in Example 1, WFDH$(P) = \{not\ \ c\}$ and WFDS$(P) = \{a \vee b, d, not\ c\}$. Notice that e is unknown.

4 Transformation-based semantics

In this section we study the relation of the argumentation-based semantics to the transformation-based semantics. We first introduce a new program transformation so as to simplify the rule heads of disjunctive programs and then define a new transformation-based semantics (called D-WFS*) as the most skeptical semantics that satisfies both our new program transformation and Brass and Dix's set $\mathbf{T}_{\mathrm{WFS}}$ of program transformations. Our new semantics D-WFS* naturally extends the D-WFS in [2] and is no less skeptical than D-WFS. In fact, this extension is meaningful because D-WFS seems too skeptical to derive useful information from some disjunctive programs as the next example shows.

Example 2. John is traveling in Europe but we are not sure which city he is visiting. We know that, if there is no evidence to show that John is in Paris, he should be either in London or in Berlin. Also, we are informed that John is now visiting either London or Paris. This knowledge base can be conveniently expressed as the following disjunctive logic program P:

$$b \vee l \leftarrow not\ p$$
$$l \vee p \leftarrow$$

Here, b, l and p denote that John is visiting *Berlin, London* and *Paris*, respectively.

Intuitively, $not\ b$ (i. e. John is not visiting Berlin) should be inferred from P. It can be verified that neither b nor its negation $not\ b$ can be derived from P under D-WFS and STATIC while $not\ b$ can be derived under WFDS.

The intuition behind Minker's Generalized Closed World Assumption (GCWA) [9] can be read off its proof-theoretic characterization:

If, for every positive disjunction A, $P \vdash a \vee A$ implies $P \vdash A$, then $not\ a$ is derivable from P, where $\vdash$ is the inference relation in the classical logic and P is considered as a classical logic theory.

The above principle for positive DLP can be reformulated in general DLP as follows:

If, for every conditional fact $a \vee A \leftarrow not\ C$, $P \vdash (a \vee A \leftarrow not\ C)$ implies $P \vdash (A \leftarrow not\ C)$, then $not\ a$ is derivable from P, where $\vdash$ is the inference relation in the classical logic and P is considered as a classical logic theory.

However, D-WFS does not to obey the above principle as Example 2 shows. In fact, $P \vdash (b \vee l \leftarrow not\ p)$ implies $P \vdash (l \leftarrow not\ p)$ since $l \vee p \leftarrow$ is in P. But $b \notin$ D-WFS(P).

According to [2], an abstract semantics can be defined as follows.

Definition 6. *A semantics $\mathcal{S}$ is a mapping which assigns to every disjunctive program P a set $\mathcal{S}(P)$ of pure disjunctions such that the following conditions are satisfied:*

1. *if Q' is a sub-disjunction of pure disjunction Q and $Q' \in \mathcal{S}(P)$, then $Q \in \mathcal{S}(P)$;*
2. *if the rule $A \leftarrow$ is in P for a (positive) disjunction A, then $A \in \mathcal{S}(P)$;*
3. *if a is an atom and $a \notin head(P)$ (i. e. a does not appear in the rule heads of P), then $not\ a \in \mathcal{S}(P)$.*

It should be noted that a semantics satisfying the above conditions is not necessarily a *suitable* one because Definition 6 is still very general.

Besides the program transformations $\mathbf{T}_{\text{WFS}}$ in [2], we also need a new program transformation called *Head reduction* to define our semantics. This definition is designed just to reflect the semantic intuition behind the GCWA as mentioned at the beginning of this section.

Definition 7. *An atom a in disjunctive program P is called* GCWA-negated *if, for any rule r in P of form $a \vee A \leftarrow B, not\ c_1, \ldots, not\ c_t$, there is a rule $A' \leftarrow$ in P such that A' is a sub-disjunction of $A \vee c_1 \vee \cdots \vee c_t$.*

For instance, b can be *GCWA-negated* for the disjunctive program in Example 2.

Definition 8. *A rule r is an* implication *of another rule r' if $head(r') \subseteq head(r)$, $body(r') \subseteq body(r)$ and at least one inclusion is proper.*

The definition of our new semantics D-WFS* will be based on the set $\mathbf{T}^*_{\text{WFS}}$ of the following six program transformations. In the sequel, P_1 and P_2 are disjunctive programs:

– **Unfolding**: P_2 is obtained from P_1 by unfolding if there is a rule $A \leftarrow b, B, not\ C$ in P_1 such that

$$\begin{aligned} P_2 = P_1 - \{A \leftarrow b, B, not\ C\} \\ \cup \{A \vee (A' - \{b\}) \leftarrow B, B', not\ C, not\ C') \mid \\ \text{there is a rule of } P_1 : A' \leftarrow B', not\ C' \text{ such that } b \in A'\}. \end{aligned}$$

– **Elimination of tautologies**: P_2 is obtained from P_1 by elimination of tautologies if there is a rule $A \leftarrow B, not\ C$ in P_1 such that $A \cap B \neq \emptyset$ and $P_2 = P_1 - \{A \leftarrow B, not\ C\}$.
– **Elimination of nonminimal rules**: P_2 is obtained from P_1 by elimination of nonminimal rules if there are two distinct rules r and r' of P_1 such that r is an implication of r' and $P_2 = P_1 - \{r\}$.
– **Positive reduction**: P_2 is obtained from P_1 by positive reduction if there is a rule $A \leftarrow B, not\ C$ in P_1 and $c \in C$ such that $c \notin head(P_1)$ and $P_2 = P_1 - \{A \leftarrow B, not\ C\} \cup \{A \leftarrow B, not\ (C - \{c\})\}$.
– **Negative reduction**: P_2 is obtained from P_1 by negative reduction if there are two rules $A \leftarrow B, not\ C$ and $A' \leftarrow$ in P_1 such that $A' \subseteq C$ and $P_2 = P_1 - \{A \leftarrow B, not\ C\}$.
– **Head reduction** P_2 is obtained from P_1 by head reduction if there is a rule $a \vee A \leftarrow B, not\ C$ in P_1 such that a is GCWA-negated and $P_2 = P_1 \cup \{A \leftarrow B, not\ C\} - \{a \vee A \leftarrow B, not\ C\}$.

Example 3. Consider the disjunctive program P in Example 2. Since the atom b is GCWA-negated, P can be transformed into the following disjunctive program P' by *Head reduction*:

$$l \leftarrow not\ p$$
$$l \vee p \leftarrow$$

Suppose that $\mathcal{S}$ is a semantics. Then by Definition 6, $l \vee p \in \mathcal{S}$ and $not\ b \in \mathcal{S}$.

We say a semantics $\mathcal{S}$ satisfies a program transformation T (or, $\mathcal{S}$ is invariant under T) if $\mathcal{S}(P_1) = \mathcal{S}(P_2)$ for any two disjunctive programs P_1 and P_2 with $P_2 = T(P_1)$.

Let $\mathcal{S}$ and $\mathcal{S}'$ be two semantics. $\mathcal{S}$ is *weaker* than $\mathcal{S}'$ if $\mathcal{S}(P) \subseteq \mathcal{S}'(P)$ for any disjunctive program P.

We present the main definition of this section as follows.

Definition 9. *(D-WFS*) The semantics D-WFS* for disjunctive programs is defined as the weakest semantics allowing all program transformations in $\mathbf{T}^*_{WFS}$.*

This definition is not constructive and thus it can not be directly used to compute the semantics D-WFS* (a bottom-up procedure will be given in the next section). In the rest of this section, we first look at some properties of D-WFS*.

As the following theorem shows, D-WFS$^*(P)$ is well-defined for every disjunctive program P. This is guaranteed by the following two lemmas.

Lemma 1. *There is a semantics that satisfies all the program transformations in $\mathbf{T}^*_{WFS}$.*

Lemma 2. *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two semantics satisfying $\mathbf{T}^*_{WFS}$. Then their intersection $\mathcal{S} = \mathcal{S}_1 \cap \mathcal{S}_2$ is also a semantics and satisfies $\mathbf{T}^*_{WFS}$.*

Therefore, we have the following result which shows that semantics D-WFS* assigns the unique model state D-WFS$^*(P)$ for each disjunctive program P.

Theorem 1. *For any disjunctive program P, D-WFS$^*(P)$ is well-defined.*

Since the set $\mathbf{T}_{\text{WFS}}$ of program transformations in [2] is a subset of $\mathbf{T}^*_{\text{WFS}}$, our D-WFS* extends the original D-WFS in the following sense.

Theorem 2. *Let P be a disjunctive program. Then*

$$\textit{D-WFS}(P) \subseteq \textit{D-WFS}^*(P).$$

The converse of Theorem 2 is not true in general. As we will see in Section 5, for the disjunctive program P in Example 2, $not\ b \in$ D-WFS$^*(P)$ but $not\ b \notin$ D-WFS(P). This theorem also implies that D-WFS* extends the restricted STATIC since the D-WFS* is equivalent to the restricted STATIC [3].

5 Bottom-up Computation

Parallel to the computation for D-WFS [2], we will first provide a bottom-up procedure for D-WFS* and then show the equivalence of D-WFS* and WFDS. As a result, we actually provide a bottom-up computation for WFDS.

Let P be a disjunctive program. Our bottom-up computation for D-WFS$^*(P)$ consists of two stages. At the first stage, P is equivalently transformed into a negative program Lft(P) called the *least fixpoint transformation*. The details of this transformation can be found in [2, 13]. The basic idea is to first evaluate body atoms of the rules in P but delay the negative body literals. The second stage is to further simplify Lft(P) into res$^*(P)$ from which the semantics D-WFS$^*(P)$ can be directly read off.

5.1 Strong Residual Program

In general, the negative program Lft(P) can be further simplified by deleting unnecessary rules, unnecessary body literals and unnecessary head atoms. This leads to the idea of so-called *reductions*, which was firstly studied in [4] and then generalized to the case of disjunctive logic programs in [2]. The reduction of a disjunctive program P is called the residual program of P. The following is a generalization of Brass and Dix's residual programs.

Let N_{GCWA} be the set of atoms that are GCWA-negated in disjunctive program N. The reduction operator R^* is defined as, for any negative program N (i. e. a set of conditional facts),

$$\begin{aligned} R^*(N) = \{\ & (A - a) \leftarrow not\ (C \cap head(N)) \mid \\ & \text{there is rule } r \in N : A \leftarrow not\ C \text{ such that} \\ & (1) \text{ no rule of form } (A' \leftarrow) \text{ with } A' \subseteq C, \\ & (2) \text{ no rule } r' \text{ s.t. } r \text{ is an implication of } r', \\ & (3) a \in N_{\text{GCWA}} \text{ and } A - N_{\text{GCWA}} \neq \emptyset\}. \end{aligned}$$

The notion of the *implication* of rules can be found in Definition 8. For any disjunctive program P, we can first transform it into the negative disjunctive program Lft(P). Then, *fully* perform the reduction R^* on Lft(P) to obtain a simplified negative program $res^*(P)$ (the *strong residual program* of P). The iteration procedure of R^* will finally stop in finite steps because B_P contains finite number of atoms and the total number of atoms occurring in each N is reduced by R^*. This procedure is precisely formulated in the next definition, which has the same form as Definition 3.4 in [2] (the difference is only in that we have a new reduction operator R^* here).

Definition 10. *(strong residual program) Let P be a disjunctive program. Then we have a sequence of negative programs $\{N_i\}_{i\geq 0}$ with $N_0 = Lft(P)$ and $N_{i+1} = R^*(N_i)$. Let $N_{t+1} = R^*(N_t)$. Then we call N_t is the strong residual program of P and denote it as $res^*(P)$.*

Since the Head reduction has been directly embedded into the operator R^*, the following result can be obtained from Theorem 4.3 in [2], which guarantees the completeness of our bottom-up computation.

Theorem 3. *Let P and P' be two disjunctive programs. If P is transformed into P' by a program transformation in $\mathbf{T}^*_{WFS}$, then $res^*(P) = res^*(P')$.*

This theorem has the following interesting corollary.

Corollary 1. *Let $\mathcal{S}$ be a semantics satisfying $\mathcal{S}(P) = \mathcal{S}(res^*(P))$ for all disjunctive program P. Then $\mathcal{S}$ allows all program transformations in $\mathbf{T}^*_{WFS}$.*

This corollary implies that, if $\mathcal{S}_0$ is a mapping from the set of all strong residual programs to the set of model states and it satisfies all properties in Definition 6, then the mapping defined by $\mathcal{S}(P) = \mathcal{S}(\text{res}^*(P))$ is a semantics. Therefore, the following lemma is obtained from the fact that D-WFS* is the weakest semantics.

Lemma 3. *Given disjunctive program P, we have*

$$\textit{D-WFS}^*(\textit{res}^*(P)) = \textit{D-WFS}^*_+(P) \cup \textit{D-WFS}^*_-(P)$$

where

$\textit{D-WFS}^*_+(\textit{res}^*(P)) = \{A \in DB^+_P \mid \textit{rule } A' \leftarrow \textit{ is in } res^*(P) \textit{ for some sub-disjunction } A' \textit{ of } A\}$

$\textit{D-WFS}^*_-(\textit{res}^*(P)) = \{A \in DB^-_P \mid \textit{if } a \notin head(res^*(P)) \textit{ for some atom } a \textit{ appearing in } A.\}$

Thus, for any disjunctive program P, it is an easy task to get the semantics D-WFS*(res$^*(P)$) of its strong residual program.

The main theorem in this section can be stated as follows.

Theorem 4. *For any disjunctive program P, we have*

$$\textit{D-WFS}^*(P) = \textit{D-WFS}^*_+(P) \cup \textit{D-WFS}^*_-(P)$$

where

$\textit{D-WFS}^*_+(P) = \{A \in DB^+_P \mid \textit{rule } A' \leftarrow \textit{ is in } res^*(P) \textit{ for some sub-disjunction } A' \textit{ of } A\}$

$\textit{D-WFS}^*_-(P) = \{A \in DB^-_P \mid \textit{if } a \notin head(res^*(P)) \textit{ for some atoma appearing in } A.\}$

Example 4. Consider again the disjunctive program P in Example 2. The strong residual program $res^*(P)$ is as follows:

$$l \leftarrow not\ p$$
$$l \vee p \leftarrow$$

Thus, D-WFS$^*(P) = \{l \vee p, not\ b\}$ [2].

[2] D-WFS$^*(P)$ should include all pure disjunctions implied by either $l \vee p$ or $not\ b$. However, the little abusing of notion here simplifies our notation.

5.2 Equivalence of WFDS and D-WFS*

Before we present the main theorem of this section, we need some properties of WFDS. First, we can justify that WFDS is a semantics in the sense of Definition 6. Moreover, it possesses the following two properties which can be verified directly.

Proposition 1. *WFDS satisfies all program transformations in* $\mathbf{T}^*_{WFS}$.

This proposition implies that the argumentation-based semantics WFDS is always at least as strong as the transformation-based semantics D-WFS*.

The next result convinces that the strong residual program $res^*(P)$ of disjunctive program P is equivalent to P w.r.t. the semantics WFDS. Therefore, we can first transform P into $res^*(P)$ and then compute WFDS$(res^*(P))$.

Proposition 2. *For any disjunctive program* P,

$$WFDS(P) = WFDS(res^*(P)).$$

It has been shown in [2] that Lft and their reduction operator R can be simulated by $\mathbf{T}_{\text{WFS}} = \mathbf{T}^*_{\text{WFS}} - \{\text{Head reduction}\}$, we have that Lft and R^* can be simulated by $\mathbf{T}^*_{\text{WFS}}$. Thus, the above proposition holds.

Now we can state the main result of this section, which asserts the equivalence of D-WFS* and WFDS.

Theorem 5. *For any disjunctive logic program* P,

$$WFDS(P) = D\text{-}WFS^*(P).$$

An important implication of this result is that the well-founded semantics WFDS also enjoys a bottom-up procedure similar to the D-WFS.

6 Unfounded Sets

The first definition of the well-founded model [12] is given in term of *unfounded sets* and it has been proved that the notion of unfounded sets constitutes a powerful and intuitive tool for defining semantics for logic programs. This notion has also been generalized to characterizing stable semantics for disjunctive logic programs in [7, 5]. However, the two kinds of unfounded sets defined in [7, 5] can not be used to define an intended well-founded semantics for disjunctive programs.

Example 5. [3]

$$a \vee b \leftarrow$$
$$c \leftarrow not\ a, not\ b$$

[3] This example is due to Jürgen Dix (personal communication)

Intuitively, $not\ c$ should be derived from the above disjunctive program and actually, many semantics including DWFS, STATIC and WFDS assign a truth value '*false*' for c. However, according to the definitions of unfounded sets in [7, 5], c is not in any n-fold application of the well-founded operators on the empty set. For this reason, a more reasonable definition of the unfounded sets for disjunctive programs is in order.

In this section, we will define a new notion of unfounded sets for disjunctive programs and show that the well-founded semantics U-WFS defined by our notion is equivalent to D-WFS* and WFDS.

We say $body(r)$ of $r \in P$ is *true* wrt model state S, denoted $S \models body(r)$, if $body(r) \subseteq S$; $body(r)$ is *false* wrt model state S, denoted $S \models \neg body(r)$ if either (1) the complement of a literal in $body(r)$ is in S or (2) there is a disjunction $a_1 \vee \cdots \vee a_n \in S$ such that $\{not\ a_1, \ldots, not\ a_n\} \subseteq body(r)$.

In Example 5, the second rule is false wrt $S = \{a \vee b\}$.

Definition 11. *Let S be a model state of disjunctive program P, a set X of ground atoms is an* unfounded set *for P wrt S if, for each $a \in X$ and each rule $r \in P$ such that $a \in head(r)$, at least one of the following conditions holds:*

1. the body of r is false wrt S;

2. there is $x \in X$ such that $x \in body^+(r)$;

3. if $S \models body(r)$, then $S \models (head(r) - X)$.

Notice that the above definition generalized the notions of unfounded sets in [7, 5] in two ways. Firstly, the original ones are defined only for interpretations (sets of ground literals) rather than for model states. An interpretation is a model state but not vice versa. Secondly, though one can redefine the original notions of unfounded sets for model states, such unfounded sets are still too weak to capture the intended well-founded semantics of some disjunctive programs. Consider Example 5, let $S = \{a \vee b\}$. According to definition 11, the set $\{c\}$ is an unfounded set of P wrt S, but $\{c\}$ is not an unfounded set in the sense of Leone or Eiter.

Having the new notion of unfounded sets, we are ready to define the well-known operator $\mathcal{W}_P$ for any disjunctive program P.

If P has the greatest unfounded set wrt a model state, we denote it $\mathcal{U}_P(S)$. However, $\mathcal{U}_P(S)$ may be undefined for some S. For example, let $P = \{a \vee b\}$ and $S = \{a, b\}$. Then $X_1 = \{a\}$ and $X_2 = \{b\}$ are two unfounded sets wrt S but $X = \{a, b\}$ is not. Here we will not discuss the operator $\mathcal{U}_P(S)$ in detail.

Definition 12. *Let P be a disjunctive program, the operator $\mathcal{T}_P$ is defined as, for any model state S,*

$$\mathcal{T}_P(S) = \{A \in DB_P \mid \text{there is a rule } r \in P : A \vee a_1 \vee \cdots \vee a_n \leftarrow body(r) \text{ such that } S \models body(r) \text{ and } not\ a_1, \ldots, not\ a_n \in S\}.$$

Notice that $\mathcal{T}_P(S)$ is a set of positive disjunctions rather than just a set of atoms.

Definition 13. *Let P be a disjunctive program, the operator $\mathcal{W}_P$ is defined as, for any model state S,*

$$\mathcal{W}_P(S) = \mathcal{T}_P(S) \cup not.\mathcal{U}_P(S).$$

where $not.\mathcal{U}_P(S) = \{not\ p \mid p \in \mathcal{U}_P(S)\}$.

In general, $\mathcal{W}_P$ is a partial function because there may be no greatest unfounded set wrt model state S as mentioned previously.

However, we can prove that $\mathcal{W}_P$ has the least fixpoint. Given a disjunctive program P, we define a sequence of model states $\{W_k\}_{k\in\mathcal{N}}$ where $W_0 = \emptyset$ and $W_k = \mathcal{W}_P(W_{k-1})$ for $k > 0$.

Similar to Proposition 5.6 in [7], we can prove the following proposition.

Proposition 3. *Let P be a disjunctive program. Then*

1. Every model state W_k is well-defined and the sequence $\{W_k\}_{k\in\mathcal{N}}$ is increasing.

2. the limit $\cup_{k\geq 0} W_k$ of the sequence $\{W_k\}_{k\in\mathcal{N}}$ is the least fixpoint of $\mathcal{W}_P$.

Since we consider only finite propositional programs in this paper, there is some $t \geq 0$ such that $W_t = W_{t+1}$.

The well-founded semantics U-WFS is defined by

$$\text{U-WFS}(P) = \text{lfp}(\mathcal{W}_P).$$

For the program P in Example 5, U-WFS$(P) = \{a \vee b, not\ c\}$.

An important result is that WFDS (and thus D-WFS*) can also be equivalently characterized in term of the unfounded sets defined in this section.

Theorem 6. *For any disjunctive program P,*

$$\textit{WFDS}(P) = \textit{U-WFS}(P).$$

Theorem 6 provides further evidence for suitability of WFDS (equivalently, D-WFS*) as the intended well-founded semantics for disjunctive logic programs.

By the following lemma, we can directly prove Theorem 6.

Lemma 4. *Let P be a disjunctive program. Then $W_k = S_k$ for any $k \geq 0$.*

This lemma also reveals a kind of correspondence between the well-founded disjunctive hypotheses and the unfounded sets.

7 Conclusion

In this paper we have investigated recent approaches to defining well-founded semantics for disjunctive logic programs. We first provided a minor modification of the argumentative semantics WFDS defined in [13]. Based on some intuitive program transformations, we proposed an extension to the D-WFS in [2]. In our approach, we introduce a new program transformation called Head reduction. This transformation plays a similar role in DLP as the GCWA [9] in positive DLP. We have also given a new definition of the unfounded sets for disjunctive programs, which is a generalization of the unfounded sets investigated by [7, 5]. This new notion of unfounded sets fully takes disjunctive information into consideration and provides another characterization for disjunctive well-founded semantics. The main contribution of this paper is the equivalence of U-WFS, D-WFS and WFDS. We have also provided a bottom-up computation for our semantics. A top-down procedure is presented in [14], which is sound and complete with respect to our semantics. These results show that there exists a disjunctive

well-founded semantics which can be characterized in terms of argumentation, program transformations, unfounded sets and resolution. The fact that different starting points lead to the same semantics provides strong support for WFDS. Future work will concentrate on more efficient algorithms and applications.

Acknowledgments The author would like to thank Philippe Besnard, James Delgrande, Thomas Linke and Torsten Schaub for helpful comments on this work. This work was supported by DFG under grant FOR 375/1-1, TP C and NSFC under grant 69883008.

References

1. S. Brass, J. Dix. Characterizations of the Disjunctive Well-founded Semantics: Confluent Calculi and Iterated GCWA. *Journal of Automated Reasoning*, 20(1):143–165, 1998.
2. S. Brass, J. Dix. Semantics of disjunctive logic programs based on partial evaluation. *Journal of Logic programming*, 38(3):167-312, 1999.
3. S. Brass, J. Dix, I. Niemelä, T. Przymusinski. On the equivalence of the Static and Disjunctive Well-founded Semantics and its computation. *Theoretical Computer Science*, 258(1-2): 523-553, 2001.
4. F. Bry. Negation in logic programming: A formalization in constructive logic. In: D. Karagiannis ed. *Information Systems and Artificial Intelligence: Integration Aspects (LNCS 474)*, Springer, pages 30-46, 1990.
5. T. Eiter, N. Leone and D. Sacca. On the partial semantics for disjunctive deductive databases. *Annals of Math. and AI.*, 19(1-2): 59-96, 1997.
6. M. Gelfond, V. Lifschitz. The stable model semantics for logic programming. In: *Proceedings of the 5th Symposium on Logic Programming*, MIT Press, pages 1070-1080, 1988.
7. N. Leone, P. Rullo and F. Scarcello. Disjunctive stable models: unfounded sets, fixpoint semantics, and computation. *Information and Computation*, 135(2): 69-112, 1997.
8. J. Lobo, J. Minker and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, 1992.
9. J. Minker. On indefinite databases and the closed world assumption. LNCS 138, pages 292-308, 1982.
10. T. Przymusinski. Static semantics of logic programs. *Annals of Math. and AI.*, 14: 323-357, 1995.
11. K. Ross. A procedural semantics for well-founded negation in logic programs. *Journal of Logic programming*, 13(1): 1-22, 1992.
12. A. Van Gelder, K. A. Ross and J. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3): 620-650, 1991.
13. K. Wang. Argumentation-based abduction in disjunctive logic programming. *Journal of Logic programming*, 45(1-3): 105-141, 2000.
14. K. Wang. A top-down procedure for disjunctive well-founded semantics. In: *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR'01)*, Springer, 2001.
15. J. You, L. Yuan and R. Goebel. An abductive approach to disjunctive logic programming. *Journal of Logic programming*, 44(1-3): 101-127, 2000.