A Top-down Procedure for Disjunctive Well-founded Semantics

Kewen Wang[⋆]

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

Abstract. Skepticism is one of the most important semantic intuitions in artificial intelligence. The semantics formalizing skeptical reasoning in (disjunctive) logic programming is usually named *well-founded semantics*. However, the issue of defining and computing the well-founded semantics for disjunctive programs and databases has proved to be far more complex and difficult than for normal logic programs. The argumentation-based semantics WFDS is among the most promising proposals that attempts to define a natural well-founded semantics for disjunctive programs. In this paper, we propose a top-down procedure for WFDS called D-SLS Resolution, which naturally extends the Global SLS-resolution and SLI-resolution. We prove that D-SLS Resolution is sound and complete with respect to WFDS. This result in turn provides a further yet more powerful argument in favor of the WFDS.

1 Introduction

Disjunctive logic programming (DLP) has gained wide acceptance as an important tool for knowledge representation. One critical reason is that DLP is more expressive and natural to use than normal (i.e. non-disjunctive) logic programming. The additional expressive power allows direct encodings of a great number of application domains into logic programs. However, the issue of defining and computing semantics for disjunctive programs and databases has proved to be far more complex and difficult than for normal logic programs. The skepticism and credulism are two major semantic intuitions for knowledge representation. A skeptical reasoner does not infer any conclusion in uncertainty conditions while a credulous reasoner tries to give conclusions as much as possible. Therefore, a skeptical reasoner usually get more feasible conclusions. In normal logic programming, these two opposite semantic intuitions are suitably captured by the well-founded semantics [11] and the stable semantics [6], respectively. There has already been a widely accepted stable semantics for disjunctive programs [9]. To date, there is no widely accepted well-founded semantics for DLP and no consensus has been reached about what constitutes an intended semantics for skeptical reasoning in DLP. Based on a comparative study of some recent approaches to defining well-founded semantics for disjunctive programs in [2, 9, 5, 7, 12], it has been proved in [13] that these approaches become equivalent when some "minor" modifications are made on them.

[⋆] On leave from Tsinghua University, Beijing.

Specifically, there exists a semantics (i. e. WFDS*) for well-founded reasoning in DLP which can be equivalently characterized by argumentation, program transformations and unfounded sets.

In the same style as the D-WFS defined in [1, 2], a bottom-up computation procure has also been provided in [13]. In this paper, we investigate the problem of top-down computation for disjunctive well-founded semantics. Specifically, we propose a top-down procedure for disjunctive well-founded semantics called D-SLS Resolution, which naturally extends the Global SLS-resolution and SLI-resolution. We prove that D-SLS Resolution is sound and complete with respect to WFDS*.

Since logic programming is essentially goal-oriented, the existence of an elegant top-down procedure is surely a significant feature for query answering under any semantics. Our results in turn provide further yet more powerful arguments in favor of the semantics WFDS*.

The paper is organized as follows. In the next section we briefly recall related definitions in logic programming and specify our notations. In Section 3 we give the argumentative definition of the semantics WFDS*. Then in Section 4 we present the D-SLS Resolution procedure. Our procedure is not only a combination of Ross's Global SLS-resolution and SLI-resolution, it also elegantly incorporates the intuition of resolving default negation with disjunctive information. To illustrate our resolution procedure and its relation to some other semantic intuitions, two examples are given in Section 5. In Section 6 we state the soundness and completeness of D-SLS Resolution with respect to WFDS*. Finally, in Section 7 we conclude the paper.

2 Preliminaries

We assume the existence of an arbitrary, but fixed propositional language, generated from a selected set of propositional symbols (atoms). An expression (disjunction, formula, rule, or set of rules, etc) with variables is understood as an abbreviation for the set of all its grounded instances. If S is an expression, $atoms(S)$ denotes the set of all atoms appearing in S.

A *general disjunctive logic program* (simply, *disjunctive program*) P is defined as a finite set of rules of the form:

$$p_1 \vee \cdots \vee p_t \leftarrow p_{t+1}, \ldots, p_s, not\ p_{s+1}, \ldots, not\ p_n. \tag{1}$$

Here, $n \geq s \geq t > 0$ and p_i's are atoms for $i = 1, \ldots, n$. The symbols '$\vee$' and 'not' denote (non-classical) disjunction and default negation, respectively.

A literal is either an atom p or its default negation $not\ p$ while $not\ p$ is called a *negative* literal.

The informal meaning of rule (1) is that "if $p_{t+1}, \ldots, p_s$ are true and $p_{s+1}, \ldots, p_n$ are all not provable, then one of $\{p_1, \ldots, p_t\}$ is true". For example, $male(greg) \vee female(greg) \leftarrow animal(greg), not\ ab(greg)$ means, informally, that if $greg$ is an animal and it is not provable that $greg$ is abnormal, then $greg$ is either male or female.

If $t = 1$, rule (1) is said to be *normal*. P is a *normal program* if each rule of P is normal.

If $n = s$, rule (1) is said to be *positive*. P is a *positive disjunctive program* if each rule of P is positive.

If $t = s$, rule (1) is said to be *negative*. P is a *negative disjunctive program* if each rule of P is negative.

As usual, B_P is the Herbrand base of disjunctive program P (i. e. 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 $DB_P = DB_P^+ \cup DB_P^-$ 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 α and $\beta = \alpha \vee \alpha'$ are two disjunctions, then we say α is a *sub-disjunction* of β.

A *model state* of disjunctive program P is a subset of DB_P. Usually, a well-founded semantics for disjunctive logic programs is defined as a mapping such that each disjunctive program P is assigned a model state.

For simplicity, we also express a rule of form (1) as $\Sigma \leftarrow \Pi_1, not.\Pi_2$, where Σ is a disjunction of atoms in B_P, Π_1 a finite subset of B_P denoting a conjunction of atoms, and $not.\Pi_2 = \{not\ q \mid q \in \Pi_2\}$ for $\Pi_2 \subseteq B_P$ denoting a conjunction of negative literals.

3 Skeptical Argumentation

As illustrated in [12], argumentation can be used to define a unifying semantic framework for DLP. In this section, we first briefly recall the well-founded extension semantics WFDS in [12] and give a minor modification WFDS* of WFDS.

The basic idea of the argumentation-based approach for DLP is to translate each disjunctive logic program into an argument framework $\mathbf{F}_P = \langle P, DB_P^-, \leadsto_P \rangle$. Here, an *assumption* of P is a negative disjunction of P, and a *hypothesis* is a set of assumptions; $\leadsto_P$ 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*

$$P_\Delta^+ = \{A \leftarrow B \mid \textit{there is a rule of form } A \leftarrow B, not.C \textit{ in } P \textit{ s.t. } not.C \subseteq \Delta\}.$$

Based on Definition 1, we will first introduce a special resolution $\vdash_P$ which resolves default-negation literals with a disjunction and can be intuitively illustrated by the following principle:

If there is an agent who holds the assumptions $not\ b_1, \ldots, not\ b_m$ and can infer the disjunctive information $b_1 \vee \cdots \vee b_m \vee b_{m+1} \vee \cdots \vee b_n$, then the agent should be able to infer $b_{m+1} \vee \cdots \vee b_n$.

The following definition precisely formulates this principle in the setting of DLP.

Definition 2. *Let $\varDelta$ be a hypothesis of disjunctive program P and $\alpha \in DB_P^+$. If there exists $\beta \in DB_P^+$ and $not\ b_1, \ldots, not\ b_m \in \varDelta$ such that $\beta = \alpha \vee b_1 \vee \cdots \vee b_m$ and $P_\varDelta^+ \vdash \beta$. Then $\varDelta$ is said to be a* supporting hypothesis *for α, denoted $\varDelta \vdash_P \alpha$. Here $\vdash$ is the classical inference; $P_\varDelta^+$ is considered as a classical logic theory while β is considered as a formula in classical logic.*

The consequence set of $\varDelta$ consists of all positive disjunctions that are supported by $\varDelta$:

$cons_P(\varDelta) = \{\alpha \mid \alpha \in DB_P^+, \varDelta \vdash_P \alpha\}$.

For example, if $P = \{a \vee b \leftarrow c, not\ d;\ \ c \leftarrow\}$ and $\varDelta = \{not\ b, not\ d\}$, then $\varDelta \vdash_P a$.

The task of defining a semantics for a disjunctive logic program P is to determine the state that can represent the intended meaning of P. Here we first specify the negative information in the semantics and then derive the positive part. To derive suitable hypotheses for a given disjunctive program, some constraints will be required to filter out unintuitive hypotheses.

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

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

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

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

Example 1.

$$\begin{aligned}
a \vee b &\leftarrow d \\
c &\leftarrow d, not\ a, not\ b \\
d &\leftarrow \\
e &\leftarrow not\ e
\end{aligned}$$

Let $\varDelta' = \{not\ c\}$ and $\varDelta = \{not\ a, not\ b\}$, then $\varDelta \leadsto_P \varDelta'$.

The next definition specifies what is an acceptable hypothesis.

Definition 4. *Let $\varDelta$ be a hypothesis of disjunctive program P. An assumption $\beta \in DB_P^-$ is* admissible *with respect to $\varDelta$ if $\varDelta \leadsto_P \varDelta'$ holds for any hypothesis $\varDelta'$ of P such that $\varDelta' \leadsto_P \{\beta\}$.*

Denote $\mathcal{A}_P(\varDelta) = \{\alpha \in DB_P^- \mid \alpha$ is admissible with respect to $\varDelta\}$.

For any disjunctive program P, $\mathcal{A}_P$ is a monotonic operator: $\varDelta \subseteq \varDelta'$ implies $\mathcal{A}_P(\varDelta) \subseteq \mathcal{A}_P(\varDelta')$ for any two hypotheses $\varDelta$ and $\varDelta'$ of P. Thus, $\mathcal{A}_P$ has the least fixpoint *lfp*$(\mathcal{A}_P)$. Since B_P is finite in this paper, the fixpoint can be obtained in finite steps by iterating $\mathcal{A}_P$ from the emptyset. That is, *lfp*$(\mathcal{A}_P) = \mathcal{A}_P^k(\emptyset)$ where $\mathcal{A}_P^{k+1}(\emptyset) = \mathcal{A}_P^k(\emptyset)$.

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 extension 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 program P in Example 1, WFDS$(P) = \{a \vee b, d, not\ c, not\ a \vee not\ b\}$. To compare with different semantics, $\mathcal{A}_P$ can be modified by defining

$$\mathcal{A}_P(\Delta) = \{\beta \in DB_P^- \mid not\ q \text{ is admissible w.r.t. } \Delta \text{ for some literal } not\ q \text{ in } \beta\}.$$

Parallel to the definition of WFDS, we can get a new well-founded semantics denoted WFDS* for disjunctive programs, which is a modification of WFDS. For instance, let P be the program given in Example 1, then WFDS$^*(P) = \{a \vee b, d, not\ \ c\}$. Now $not\ a \vee not\ b$ is no longer in WFDS$^*(P)$.

We can prove that WFDS is no less strong than WFDS* in the following sense.

Proposition 1. *For any disjunctive program* P *and* $\alpha \in DB_P$*, if* $\alpha \in$ *WFDS*$^*(P)$*, then* $\alpha \in$ *WFDS*(P).

As we have seen above, the converse of this proposition is not true in general. Specifically, WFDS allows more negative disjunctions to be inferred. However, this is not a big difference as the following results show. In fact, except for this difference, these two semantics coincide.

Proposition 2. *Let* Δ *be an admissible hypothesis of disjunctive program* P*. If* $\beta \in DB_P^-$ *but is not a literal, then*

1. *a hypothesis* α *is admissible w.r.t.* Δ *iff it is admissible w.r.t.* $\Delta - \{\beta\}$.
2. *for any positive disjunction* D*,* $D \in cons_P(\Delta)$ *iff* $D \in cons_P(\Delta - \{\beta\})$.

An interesting result is the equivalence of WFDS and WFDS*.

Theorem 1. *Let* P *be a disjunctive program. Then*

1. $not\ p \in$ *WFDS*$^*(P)$ *iff* $not\ p \in$ *WFDS*(P) *for any atom* p.
2. $\alpha \in$ *WFDS*$^*(P)$ *iff* $\alpha \in$ *WFDS*(P) *for any positive disjunction* α.

This theorem convinces that the difference of WFDS* from WFDS is only in that they derive different sets of true negative disjunctions.

4 D-SLS Resolution

In this section, we will define a top-down procedure, called D-SLS Resolution, for disjunctive well-founded semantics. This procedure combines the idea of Global SLS-resolution in [10] with a linear resolution procedure. The linear resolution is a generalization of the SLI-resolution presented in [8]. One key part in our procedure is the incorporation of resolving default negation with disjunctive information into SLS-resolution.

D-SLS Resolution will be based on the notion of D-SLS *tree*, which in turn depends on the notion of *positive trees*. In the next section, we prove that D-SLS Resolution is sound and complete with respect to the disjunctive well-founded semantics WFDS and WFDS*.

To achieve completeness of D-SLS Resolution, we adopt the so-called *positivistic* computation rule, that is, we always select positive literals ahead of negative ones.

A goal G is of the form $\leftarrow D_1, \ldots, D_r, \neg b_1, \ldots, \neg b_m, not\ c_1, \ldots, not\ c_n$, where each D_i is a positive disjunction; all b_i and c_i are atoms. To distinguish from default literals, we shall say that l is a classic literal if $l = p$ or $l = \neg p$.

In our resolution-like procedure, given a rule $C:\ \Sigma \leftarrow \Pi_1, not.\Pi_2$, we transform C into a goal $gt(C)\ :\ \leftarrow \neg\Sigma, \Pi_1, not.\Pi_2$ and call it the *goal transformation* of C, where $\neg\Sigma = \{\neg p \mid p \in \Sigma\}$.

Since our resolution is to resolve literals in both heads and bodies of rules, this transformation allows a unifying and simple approach to defining resolution-like procedure for disjunctive logic programs as we shall see.

The special goal $\leftarrow$ is called an *empty goal*. The empty goal $\leftarrow$ is also written as the familiar symbol $\Box$. A non-empty goal of form $\leftarrow \neg\Sigma, not.\Pi$ is said to be a negative goal.

Given a disjunctive program P, set $gt(P) = \{gt(C) :\ C \in P\}$. The traditional goal resolution can be generalized as follows.

Disjunctive Goal Resolution (DGR) If $G\ :\ \leftarrow\ b_1 \vee \cdots \vee b_r, G_1$ and $G'\ :\ \leftarrow \neg b_1, \ldots, \neg b_s, G_2$ are two goals with $s \leq r$, then the DGR-*resolvent* of G with G' on selected disjunction $b_1 \vee \cdots \vee b_r$ is the goal $\leftarrow G_1, G_2$.

It should be noted that resolution rule DGR incorporates several resolution rules including *Goal resolution*, *Ancestor resolution* and *Body literal resolution* [8, 15]. Since we allow positive disjunctions in goals and the resolution rule, DGR is more powerful than the above mentioned three resolution rules as the following example shows.

Example 2. Let P be the following disjunctive program:

$$\begin{aligned} a \vee b &\leftarrow \\ c &\leftarrow not\ a, not\ b \end{aligned}$$

Then P can be transformed into the following set $gt(P)$ of goals:

$$\begin{aligned} &\leftarrow \neg a, \neg b \\ &\leftarrow \neg c, not\ a, not\ b \end{aligned}$$

Then the DGR-resolvent of $G\ :\ \leftarrow\ c$ with the second goal is $\leftarrow\ not\ a, not\ b$, which can be obtained by the ordinary goal resolution; however, the DGR-resolvent of goal $\leftarrow a \vee b$ with the first goal is $\Box$, which can not be obtained by any of those three resolution rules.

Definition 6. *Let P be a disjunctive program and G a goal. A positive tree T_G^+ for G is defined as follows:*

1. *The root of T_G^+ is G.*

2. *For each node $G' : \leftarrow \neg\Sigma', p, \Pi_1', not.\Pi_2'$, and each goal G_i in $gt(P)$, if G_i' is the* DGR-*resolvent of G' with G_i on p and G_i' is different from all nodes in the branch of G', then G' has a child G_i'.*

A node labeled $\leftarrow \neg p_1, \ldots, \neg p_n, not\ p_{n+1}, \ldots, not\ p_m (m \geq n \geq 0)$ is called an active node.

Thus, an active node is either the empty goal or a negative goal. For a negative goal, its success/failure has to be decided in subsequent stages. Now we define the D-SLS tree for a goal in terms of positive trees.

Definition 7. *(*D-SLS *Tree)* *Let P be a disjunctive logic program and G a goal. The* D-SLS *tree Γ_G for G is a tree whose nodes are of two types:* negation nodes *and* tree nodes. *Tree nodes are actually positive trees for intermediate goals. The nodes of Γ_G is defined inductively as follows:*

1. *The root of Γ_G is the positive tree T_G^+ for the goal G.*
2. *For any tree node T_H^+ of Γ_G, The children of T_H^+ are negation nodes, one corresponding to each active leaf of T_H^+ (there will be a negation node corresponding to an empty active leaf).*
3. *Let J be a negation node corresponding to the active leaf $\leftarrow Q$ where $Q = \{not\ q_1, \ldots, not\ q_n\}$ and $n \geq 0$. J is denoted $N(\leftarrow Q)$. Then, if $n > 0$, J has one child which is the positive tree $T^+_{q_1 \vee \cdots \vee q_n}$.*

We distinguish three types of leaves in a D-SLS *tree (*successful nodes, failed nodes *and* intermediate nodes*) according to the following rules. Successful and failed nodes also have an associated* level.

1 *For negation node J,*
   - **(a)** *if the child of a negation node J is a successful tree node, then we say J is* failed. *The level is the level of its successful child.*
   - **(b)** *if the child of a negation node J is a failed tree node, or if J has no children, then we say J is successful. The level of J is the level of the child of J (if J has no children, the level of J is 0).*

2 *For tree node T,*
   - **(a)** *if every child of a tree node T is a failed negation node, or if T is a leaf of Γ_G (i. e. T has no active leaves) then we say T is failed. T has the level 1 if T is a leaf; the level of T is $k+1$ if the maximum level of levels of the children of T is k.*
   - **(b)** *if some child of a tree node T is a successful negation node, then we say T is successful. A non-root tree node T has level $k+1$ if the minimum level of all its successful children is k. The root tree node may have several associated levels, one for each successful child; the level of the root tree node with respect to such a successful child is one more than the level of the child.*

3 *We say a node is* well determined *if it is either successful or failed. Otherwise, we say the node is* indeterminate.

Let L be an active leaf of a tree node in Γ_G. We may say that L is successful (resp. failed or indeterminate) if the corresponding negation node is successful (resp. failed or indeterminate). We may also say that the goal G is successful (resp. failed or indeterminate) if T_G^+ is successful (resp. failed or indeterminate). Compared to Global SLS-resolution for normal logic programs, D-SLS Resolution has the following major features:

1. the underlying reasoning mechanism for D-SLS Resolution is a generalization of SLI-resolution while the underlying reasoning mechanism for SLS-resolution is SLD-resolution;

2. each negation node has just one child in D-SLS Resolution while a negation node may have several children in Global SLS-resolution because disjunctions are allowed now. This makes a simpler form of D-SLS Resolution.

To guarantee the termination of D-SLS Resolution, we also assume that every node is not repeated. That is, whenever a repeated node in D-SLS tree is found, the extending of the tree will be stopped.

5 Examples

Let us look at some illustrating examples.

Example 3. Consider the following disjunctive program P:

$$\begin{aligned} a \vee b &\leftarrow c, \mathit{not}\ d \\ e &\leftarrow \mathit{not}\ a, \mathit{not}\ b \\ e &\leftarrow \mathit{not}\ a, g \\ c &\leftarrow \\ a &\leftarrow \mathit{not}\ c \end{aligned}$$

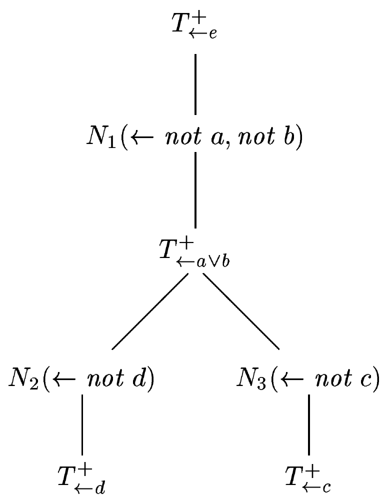
It can be verified that $\mathit{not}\ e \in \mathrm{WFDS}^*(P)$ and thus $\mathit{not}\ e \in \mathrm{WFDS}(P)$.

Now let us see how $\mathit{not}\ e$ is inferred by D-SLS Resolution.

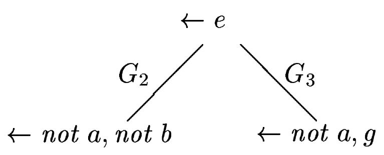
First, P is transformed into $gt(P)$ which consists of the following goals:

$$\begin{aligned} G_1 &: \leftarrow \neg a, \neg b, c, \mathit{not}\ d \\ G_2 &: \leftarrow \neg e, \mathit{not}\ a, \mathit{not}\ b \\ G_3 &: \leftarrow \neg e, \mathit{not}\ a, g \\ G_4 &: \leftarrow \neg c \\ G_5 &: \leftarrow \neg a, \mathit{not}\ c \end{aligned}$$

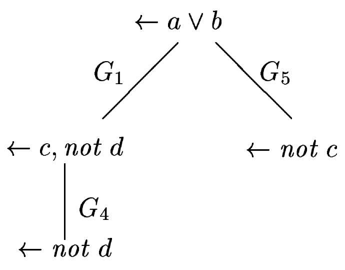
In fact, we have the following D-SLS tree $\Gamma_{\leftarrow e}$ for $\leftarrow e$:



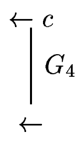
The positive tree $T^+_{\leftarrow e}$ for $\leftarrow e$ is as follows:



The positive tree $T^+_{\leftarrow a \vee b}$ is:



The positive tree $T^+_{\leftarrow d}$ consists of only the root node $\leftarrow d$.
The positive tree $T^+_{\leftarrow c}$ is



By Definition 7,

$T^+_{\leftarrow d}$ is a leaf of $\Gamma_{\leftarrow e}$ $\Rightarrow$
$T^+_{\leftarrow d}$ is a failed tree node $\Rightarrow$
N_2 is a successful negation node (no matter what N_3 is) $\Rightarrow$
the tree node $T^+_{\leftarrow a \vee b}$ is successful $\Rightarrow$
N_1 is a failed node (and N_1 is the only negation child of $T^+_{\leftarrow e}$) $\Rightarrow$
$T^+_{\leftarrow e}$ is a failed (root) node $\Rightarrow$
the goal $\leftarrow e$ is failed.

To guarantee the termination of D-SLS Resolution, we also assume that every node is not repeated. That is, whenever a repeated node in D-SLS tree is found, the extending of the tree will be stopped. For example, if we replace the last rule $a \leftarrow not\ c$ in the above example with the rule $a \leftarrow not\ a, not\ b$, then $N_3 = N_1$ and thus N_3 and its children (if any) will be deleted from $\Gamma_{\leftarrow e}$.

It should be noted that D-SLS Resolution is different from the SLIN-resolution [14]. We demonstrate this by the following example.

Example 4. Let P consist of two rules:

$$\begin{aligned} a \vee b &\leftarrow \\ c &\leftarrow not\ a, not\ b \end{aligned}$$

Although $not\ \ c \in$ WFDS$^*(P)$, c is indeterminate with respect to the SLIN-resolution. This means that SLIN-resolution is not complete for the disjunctive well-founded semantics WFDS*. However, D-SLS tree $\Gamma_{\leftarrow c}$ for the goal $\leftarrow c$ is failed (to save space, the tree is figured in one line because each of its internal nodes has the unique child):

$T^+_{\leftarrow c}$ —- $N_1(\leftarrow not\ a, not\ b)$ —- $T^+_{\leftarrow a \vee b}$ —- $N_2(\leftarrow)$ —- $T^+_{\leftarrow}$

Here, $T^+_{\leftarrow}$ has the unique node $\leftarrow$; $T^+_{\leftarrow c}$ and $T^+_{\leftarrow a \vee b}$ are as follows, respectively:

$$\begin{tikzcd} \leftarrow c \arrow[d, dash, "G_2"] & & \leftarrow a \vee b \arrow[d, dash, "G_1"] \\ \leftarrow not\ a, not\ b & & \leftarrow \end{tikzcd}$$

It is easy to see that $T^+_{\leftarrow a}$ is an indeterminate node of the D-SLS tree for the goal $\leftarrow a$.

It should be noted that, in D-SLS tree, an active node containing classic negative literals can not be ignored[1]. That is, there may be negation node having classic negative literals as child in D-SLS tree. Consider the following program:

$$\begin{aligned} b \vee l &\leftarrow not\ p \\ l \vee p &\leftarrow \end{aligned}$$

[1] This question is proposed by one referee.

The D-SLS tree $\Gamma_{\leftarrow b}$ for the goal $\leftarrow b$ is as follows:

$T^{+}_{\leftarrow b}$ —- $N(\leftarrow \neg l, not\ p)$ —- $T^{+}_{\leftarrow l \vee p}$.

It can be verified that the goal $\leftarrow b$ is failed.

6 Soundness and Completeness of D-SLS Resolution

In this section, we address the soundness and completeness of D-SLS Resolution. We first show that D-SLS Resolution is sound and complete w.r.t. the argumentative semantics WFDS. Then, by Theorem 1, we get the soundness and completeness of D-SLS Resolution w.r.t. WFDS*. Although we allow a goal to have a very general form in our D-SLS Resolution, each goal G considered in this section actually has one of the two forms: either $\leftarrow a_1 \vee \ldots \vee a_r$ or $\leftarrow a_1, \ldots, a_r, \neg b_1, \ldots, \neg b_m, not\ c_1, \ldots, not\ c_n$, where all a_i, b_i and c_i are atoms. Thus, from now on we will always mean either of the above form when a goal is mentioned. The detailed proofs of results in this section are not difficult but tedious, thus we omit them here.

Theorem 2. *(Soundness of* D-SLS *Resolution w.r.t. WFDS)*
Let P be a disjunctive logic program. Then

1. *If goal $G : \leftarrow q_1, \ldots, q_n$ is failed, then $not\ q_1 \vee \cdots \vee not\ q_n \in$ WFDS(P).*
2. *If goal $G : \leftarrow q_1 \vee \cdots \vee q_n$ is successful, then $q_1 \vee \cdots \vee q_n \in$ WFDS(P).*

To prove Theorem 2, we need only to show the following lemma.

Lemma 1. *Let P be a disjunctive logic program. Then*

1. *If goal $G : \leftarrow q_1, \ldots, q_n$ is failed, then $not\ q_1 \vee \cdots \vee not\ q_n \in$ WFDH(P).*
2. *If goal $G : \leftarrow q_1 \vee \cdots \vee q_n$ is successful, then WFDH$(P) \vdash_P q_1 \vee \cdots \vee q_n$.*

Sketch of Proof It suffices to prove the following two propositions hold by using simultaneous induction on the level $l(\Gamma_G)$ of D-SLS tree Γ_G:

S1 $not\ q_1 \vee \cdots \vee not\ q_n \in \mathbf{A}^k_P(\emptyset)$ if the goal $G : \leftarrow q_1, \ldots, q_n$ is failed and $l(\Gamma_G) = k \geq 1$;
S2 $\mathbf{A}^k_P(\emptyset) \vdash_P q_1 \vee \cdots \vee q_n$ if the goal $G : \leftarrow q_1 \vee \cdots \vee q_n$ is successful and $l(\Gamma_G) = k \geq 0$.

Theorem 3. *(Completeness of* D-SLS *Resolution w.r.t. WFDS)*
Let P be a disjunctive logic program. Then

1. *If $q_1 \vee \cdots \vee q_n \in$ WFDS(P), then the goal $\leftarrow q_1 \vee \cdots \vee q_n$ is successful.*
2. *If $not\ q_1 \vee \cdots \vee not\ q_n \in$ WFDS(P), then the goal $G :\leftarrow q_1, \ldots, q_n$ is failed.*

This theorem follows directly from the next lemma.

Lemma 2. *Let P be a disjunctive logic program and G a goal of P. Then*

1. *If WFDH$(P) \vdash_P q_1 \vee \cdots \vee q_n$, then the goal $G : \leftarrow q_1 \vee \cdots \vee q_n$ is successful.*
2. *If $not\ q_1 \vee \cdots \vee not\ q_n \in$ WFDH(P), then the goal $G : \leftarrow q_1, \ldots, q_n$ is failed.*

Sketch of Proof It is enough to show that both of the following C1 and C2 hold by using simultaneous induction on the level $l(\Gamma_G) = k$:

C1 For $k \geq 1$, if $\mathbf{A}^{k-1}(\emptyset) \vdash_P q_1 \vee \cdots \vee q_n$, then the goal $G :\leftarrow q_1 \vee \cdots \vee q_n$ is successful and its level is k.

C2 For $k \geq 1$, if $not\ q_1 \vee \cdots \vee not\ q_n \in \mathbf{A}_P^k(\emptyset)$, then the goal $G :\leftarrow q_1, \ldots, q_n$ is failed and its level is no more than $k+1$.

By the definition of WFDS*, for any atoms $q_1, \ldots, q_n$, $not\ q_1 \vee \cdots \vee not\ q_n \in$ WFDS$^*(P)$ if and only if $q_i \in$ WFDS$^*(P)$ for some q_i. Thus, the following two theorems follows directly from Theorem 1 and the two theorems above.

Theorem 4. *(Soundness of* D-SLS *Resolution w.r.t. WFDS*)*
Let P be a disjunctive logic program. Then

1. *If goal $G :\leftarrow q_i$ is failed for some q_i, then $not\ q_1 \vee \cdots \vee not\ q_n \in$ WFDS$^*(P)$.*
2. *If goal $G :\leftarrow q_1 \vee \cdots \vee q_n$ is successful, then $q_1 \vee \cdots \vee q_n \in$ WFDS$^*(P)$.*

Theorem 5. *(Completeness of* D-SLS *Resolution w.r.t. WFDS*)*
Let P be a disjunctive logic program. Then

1. *If $q_1 \vee \cdots \vee q_n \in$ WFDS$^*(P)$, then the goal $\leftarrow q_1 \vee \cdots \vee q_n$ is successful.*
2. *If $not\ q_1 \vee \cdots \vee not\ q_n \in$ WFDS$^*(P)$, then the goal $G :\leftarrow q_i$ is failed for some q_i.*

7 Conclusion

The main contribution of this paper is that we have proposed a top-down procedure D-SLS Resolution for disjunctive well-founded semantics. This resolution-like procedure extends both the Global SLS-resolution [10] and SLI-resolution [8]. We prove that D-SLS Resolution is sound and complete with respect to the disjunctive well-founded semantics WFDS and WFDS*. We know that the Global SLS-resolution is a classic procedure for the well-founded semantics of normal logic programs while SLI-resolution is the most important procedure for positive disjunctive programs. D-SLS Resolution is actually a novel characterization for WFDS* and thus provides a further yet powerful argument in favor of the semantics WFDS*. On the other hand, the results in this paper pave a promising way to implement the WFDS* by employing some existing theorem provers. Although this point has been made clear for Brass and Dix's D-WFS in [3], no top-down procedure is provided for their semantics.

It is worth noting that D-SLS Resolution in the current form is not efficient yet. We are currently working on more efficient algorithm for D-SLS Resolution by employing some techniques such as the tabling method [4].

Acknowledgments The author would like to thank Peter Baumgartner, Ph. Besnard, J. Dix, Th. Linke and T. Schaub for useful comments and discussions on (the topic of) this work. The author would also like to thank the four anonymous referees for their detailed comments. This work was supported in part by the German Science Foundation (DFG) within Project "Nichtmonotone Inferenzsysteme zur Verarbeitung konfligierender Regeln" under grant FOR 375/1-1, TP C, the Natural Science Foundation of China under the projects 69883008, the National Foundation Research Programme of China under G1999032704.

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*, 251 (to appear), 2001.
4. W. Chen, D. Warren. Efficient top-down computation of queries under the well-founded semantics. *J. ACM*, 43(1): 20-74, 1996.
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), 1997.
8. J. Lobo, J. Minker and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, 1992.
9. T. Przymusinski. Static semantics of logic programs. *Annals of Math. and AI.*, 14: 323-357, 1995.
10. K. Ross. A procedural semantics for well-founded negation in logic programs. *Journal of Logic programming*, 13(1): 1-22, 1992.
11. A. Van Gelder, K. A. Ross and J. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3): 620-650, 1991.
12. K. Wang. Argumentation-based abduction in disjunctive logic programming. *Journal of Logic programming*, 45(1-3):105-141, 2000.
13. K. Wang. Disjunctive well-founded semantics revisited. In: *Proceedings of the 5th Dutch-German Workshop on Nonmonotonic Reasoning Techniques and their Applications (DGNMR'01)*, Potsdam, April 4-6, pages 193-203, 2001.
14. K. Wang, F. Lin. Closed world reasoning and query evaluation in disjunctive deductive databases. In: *Proceedings of the International Conference on Applications of Prolog /DDLP'99*, 1999.
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.