TR-0817

A Uniform Approach to Fixpoint Characterization of Disjunctive and General Logic Programs

by
K. Inoue & C. Sakama (ASTEM)

November, 1992

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1 Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~5
Telex ICOT J32964

Institute for New Generation Computer Technology

A Uniform Approach to Fixpoint Characterization of Disjunctive and General Logic Programs

Katsumi Inoue
ICOT
Mita Kokusai Bldg., 21F
1-4-28 Mita, Minato-ku
Tokyo 108, Japan
phone: +81-3-3456-4365
inoue@icot.or.jp

Chiaki Sakama
ASTEM
Research Institute of Kyoto
17 Chudoji Minami-machi
Shimogyo, Kyoto 600, Japan
phone: +81-75-315-8651
sakama@astem.or.jp

Abstract

This paper presents a uniform framework for constructing a fixpoint for disjunctive and general logic programs. We first introduce a mapping which operates on a set of interpretations and show that the minimal elements of its fixpoint closure is equivalent to the set of minimal models of a positive disjunctive program. Next, we extend the fixpoint semantics to general disjunctive programs by using a suitable program transformation of a general disjunctive program. Then, we characterize the stable model semantics of general disjunctive programs by the fixpoint closure of the transformed program. The result is further extended to the answer set semantics of extended disjunctive programs.

1 Introduction

A declarative semantics of logic programs is usually characterized by using a fixpoint of a program. For definite Horn programs, van Emden and Kowalski [VEK76] have introduced a closure operator which acts over a lattice of Herbrand interpretations and shown that the operator reaches the least fixpoint that is exactly the least Herbrand model of the program. For non-Horn programs, Apt, Blair and Walker [ABW88] have developed a nonmonotonic fixpoint operator for logic programs with negation and shown that its iterative fixpoint characterizes the perfect model semantics of stratified logic programs [Pr88]. The result has further been extended to general logic programs in the context of the stable model semantics [GL88] and the well-founded semantics [VRS91, VG89, Pr89].

A disjunctive logic program is a program in which disjunctive consequences are contained. The semantics of disjunctive logic programs was firstly studied by Minker [Mi82] where he introduced the minimal model semantics of positive disjunctive programs. In the presence of negation in a program, Przymusinski has developed the perfect model semantics for (locally) stratified disjunctive programs [Pr88], and much of recent studies have been devoted to the semantics of general disjunctive programs containing non-stratified negation [BLM90, FM92, GL91, Pr90a, Pr90b, Ro89].

A fixpoint theory of disjunctive logic programs has firstly been developed by Minker and Rajasekar [MR90]. They have extended van Emden and Kowalski's fixpoint operator and characterized the minimal model semantics of a disjunctive program. The idea of their fixpoint operator is to consider a mapping over a set of positive disjunctions (called state), instead of mapping over interpretations. Then they showed that the least fixpoint of the operator consists of positive disjunctions which implies a set of minimal models of a disjunctive program. Such a state based fixpoint semantics has also been extended to stratified disjunctive programs [RM89], and general disjunctive programs [Ro89, BLM90, Pr90b]. Another approaches such as [Sa89, FM91, Dec92] present fixpoint theories of disjunctive programs based upon the manipulation of standard Herbrand interpretations.

This paper presents yet another fixpoint semantics of disjunctive logic programs. The fixpoint semantics presented in this paper provides a uniform framework not only for disjunctive logic programs, but also for general and extended logic programs. Furthermore, our fixpoint theory of disjunctive programs can give a fixpoint characterization of model generation theorem provers such as [MB88, FH91, IKH92].

The organization of this paper is as follows. In section 2, we present the fixpoint semantics of positive disjunctive programs and show its correspondence with the minimal model semantics of disjunctive programs. In section 3, we extend the results to general and extended disjunctive programs and present its connection to the stable model semantics. Section 4 discusses comparison between our fixpoint semantics and

previously proposed approaches.

2 Fixpoint Semantics for Positive Disjunctive Programs

2.1 Positive Disjunctive Programs

A *positive disjunctive program* is a finite set of clauses of the form:

$$A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m \quad (l, m \geq 0) \tag{1}$$

where A_i's and B_j's are atoms and all variables are assumed to be universally quantified at the front of the clause. A clause is called *definite* (resp. *positive*, *negative*) if $l = 1$ (resp. $m = 0$, $l = 0$). A program containing only definite clauses is called a *definite (Horn) program*. Note that in the presence of negative clauses, a program is possibly inconsistent. The disjunction $A_1 \vee \ldots \vee A_l$ is called the *head* and the conjunction $B_1 \wedge \ldots \wedge B_m$ is called the *body* of the clause. Each predicate in the head is said to be *defined* by the predicates in the body. A *ground clause* is a clause which contains no variable. A *ground program* is a program in which every variable is instantiated by the elements of the Herbrand universe of a program in every possible way. A ground program is a possibly infinite set of ground clauses. From the semantical point of view, a program is equivalent to its ground program, thus we consider a ground program in this paper.

An *interpretation* of a program is a subset of the Herbrand base $\mathcal{HB}$ of the program. An interpretation I *satisfies* a ground clause C (denoted by $I \models C$) if for C: $A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m$, $\{B_1, \ldots, B_m\} \subseteq I$ implies $A_i \subset I$ for some $1 \leq i \leq l$. Especially, if $l = 0$ and $B_1, \ldots, B_m$ are in I, I does not satisfy the negative clause. For a program P, a minimal set I which satisfies every ground clause from P is called a *minimal model* of P. If P has a unique minimal model I, it is also called the *least model* of P. When no interpretation satisfies every ground clause from P, we say that P has an *inconsistent model* $I = \mathcal{L}$ where $\mathcal{L}$ is the set of all literals in the language. A program is *consistent* if it has a minimal model different from $\mathcal{L}$, otherwise it is called *contradictory*.

In the next subsection, a program means a positive disjunctive program unless stated otherwise.

2.2 Fixpoint Semantics

This subsection presents a fixpoint semantics for positive disjunctive programs. To characterize the non-deterministic behavior of a disjunctive program, we introduce a closure operator which operates over a lattice of sets of Herbrand interpretations $2^{2^{\mathcal{HB}}}$. Then our first task is to define an ordering over such powerdomain.

Definition 2.1 Let $\mathbf{I}$ and $\mathbf{J}$ be sets of interpretations. Then, $\mathbf{I} \sqsubseteq \mathbf{J}$ iff $\mathbf{I} = \mathbf{J}$ or $\forall J \in \mathbf{J} \setminus \mathbf{I}$, $\exists I \in \mathbf{I} \setminus \mathbf{J}$ such that $I \subset J$.

Lemma 2.1 $\sqsubseteq$ is a partial order.

Proof: Since reflexivity and transitivity are obvious, we show anti-symmetry. Suppose that $\mathbf{I} \sqsubseteq \mathbf{J}$ and $\mathbf{J} \sqsubseteq \mathbf{I}$. Assume $\mathbf{I} \setminus \mathbf{J} \neq \emptyset$. Then, $\mathbf{J} \setminus \mathbf{I} \neq \emptyset$, and $\mathbf{I} \sqsubseteq \mathbf{J}$ implies that $\forall J \in \mathbf{J} \setminus \mathbf{I}$, $\exists I \in \mathbf{I} \setminus \mathbf{J}$ such that $I \subset J$. While, $\mathbf{J} \sqsubseteq \mathbf{I}$ implies that there exists $J' \in \mathbf{J} \setminus \mathbf{I}$ such that $J' \subset I$. Repeating this step generates a decreasing chain $\ldots \subset J' \subset I \subset J$ whose lower bound is $\emptyset$. In case of $\emptyset \in \mathbf{J} \setminus \mathbf{I}$, $\mathbf{I} \sqsubseteq \mathbf{J}$ does not hold. While, in case of $\emptyset \in \mathbf{I} \setminus \mathbf{J}$, $\mathbf{J} \sqsubseteq \mathbf{I}$ does not hold. In either case, it contradicts the assumption. Hence, $\mathbf{I} \setminus \mathbf{J} = \emptyset$. Also $\mathbf{J} \setminus \mathbf{I} = \emptyset$ holds in the same way. Therefore, $\mathbf{I} = \mathbf{J}$. $\square$

Each element in $2^{2^{\mathcal{HB}}}$ makes a complete lattice under the ordering $\sqsubseteq$ with the top element $\emptyset$ and the bottom element $2^{\mathcal{HB}}$.

Definition 2.2 Let P be a program and I be an interpretation. Then a mapping $T_P : 2^{\mathcal{HB}} \to 2^{2^{\mathcal{HB}}}$ is defined as follows:

$$T_P(I) = \begin{cases} \emptyset, & \text{if } I \models B_1 \wedge \ldots \wedge B_m \text{ for some ground negative clause} \\ & \quad \leftarrow B_1 \wedge \ldots \wedge B_m \text{ from } P; \\ \{\ J & \mid \text{for each ground clause } C_i : A_1^i \vee \ldots \vee A_{l_i}^i \leftarrow B_1^i \wedge \ldots \wedge B_{m_i}^i \\ & \quad \text{from } P \text{ such that } I \models B_1^i \wedge \ldots \wedge B_{m_i}^i \text{ and } I \not\models A_1^i \vee \ldots \vee A_{l_i}^i, \\ & \quad J = I \cup \bigcup_{C_i} \{A_j^i\}\ \ (1 \leq j \leq l_i)\ \}, & \text{otherwise.} \end{cases}$$

We say that a ground clause C is *violated* in an interpretation I ([MB88]) if for C: $A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m$, $\{B_1, \ldots, B_m\} \subseteq I$ but $\{A_1, \ldots, A_l\} \cap I = \emptyset$. Definition 2.2 says that, if an interpretation I does not satisfy a ground negative clause then $T_P(I) = \emptyset$, else $T_P(I)$ contains every interpretation obtained from I by adding each single disjunct from every ground clause violated in I.

Example 2.1 Let $P = \{\ a \vee b \leftarrow c, \quad d \leftarrow c, \quad c \leftarrow, \quad \leftarrow b \wedge d\ \}$.
Then $T_P(\{c\}) = \{\{a, c, d\}, \{b, c, d\}\}$ and $T_P(\{\{b, c, d\}\}) = \emptyset$.

Definition 2.3 Let P be a program and $\mathbf{I}$ be a set of interpretations. Then a mapping $\mathbf{T}_P : 2^{2^{\mathcal{HB}}} \to 2^{2^{\mathcal{HB}}}$ is defined by

$$\mathbf{T}_P(\mathbf{I}) = \bigcup_{I \in \mathbf{I}} T_P(I)$$

Especially, $\mathbf{T}_P(\emptyset) = \emptyset$.

Example 2.2 (cont. from Example 2.1) $\mathbf{T}_P(\{\{c\}, \{b, c, d\}\}) = \{\{a, c, d\}, \{b, c, d\}\}$.

The mapping $\mathbf{T}_P$ is not monotonic.

Example 2.3 Let $P = \{\ a \vee b \leftarrow c, \quad c \leftarrow , \quad \leftarrow c \wedge b, \quad e \leftarrow f\ \}$.
Then, $\{\{c\}\} \sqsubseteq \{\{c,a\},\{c,a,e\}\}$, while $\mathbf{T}_P(\{\{c\}\}) = \{\{c,a\},\{c,b\}\}$ and $\mathbf{T}_P(\{\{c,a\},\{c,a,e\}\}) = \{\{c,a\},\{c,a,e\}\}$. Thus, $\{\{c,a\},\{c,b\}\} \not\sqsubseteq \{\{c,a\},\{c,a,e\}\}$.

Lemma 2.2 Let P be a program and I be an interpretation of P. Then, $\mathbf{T}_P(\{I\}) = \{I\}$ iff I is a model of P.

Proof: I is a model of P
iff I satisfies each negative clause in P, and for each ground clause $A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m$ from P, $\{B_1,\ldots,B_m\} \subseteq I$ implies $\exists A_i \in I\ (1 \leq i \leq l)$
iff $T_P(I) = \{I\}$
iff $\mathbf{T}_P(\{I\}) = \{I\}$. □

Lemma 2.3 If $I \in \mathbf{T}_P(\{I\})$ and $J \subset \mathbf{T}_P(\{I\})$, then $I = J$.

Proof: Suppose that $J \in \mathbf{T}_P(\{I\})$ and $I \neq J$. Then, there exists an atom A in $J \setminus I$ and a ground clause $A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m$ from P such that $I \models B_1 \wedge \ldots \wedge B_m$, $I \not\models A_1 \vee \ldots \vee A_l$ and $A = A_i$ for some i $(1 \leq i \leq l)$. Hence, $I \notin \mathbf{T}_P(\{I\})$ by Definition 2.2. □

Lemma 2.4 $\mathbf{T}_P(\mathbf{I}) = \mathbf{I}$ iff for each $I \in \mathbf{I}$, $\mathbf{T}_P(\{I\}) = \{I\}$.

Proof: Since the if-part follows immediately, we prove the other direction. Suppose that $\mathbf{T}_P(\mathbf{I}) = \mathbf{I}$. Assume to the contrary that $\mathbf{T}_P(\{I\}) \neq \{I\}$ for some $I \in \mathbf{I}$. Then, $I \notin \mathbf{T}_P(\{I\})$ by Lemma 2.3. Since I is in $\mathbf{I}$, there exists J in $\mathbf{I}$ such that $J \subset I$ and $I \in \mathbf{T}_P(\{J\})$. If J is in $\mathbf{T}_P(\{J\})$, $\mathbf{T}_P(\{J\}) = \{J\}$ by Lemma 2.3. Since $I \neq J$, this is impossible. Hence, $J \notin \mathbf{T}_P(\{J\})$. Then there exists J' in $\mathbf{I}$ such that $J' \subset J$ and $J \in \mathbf{T}_P(\{J'\})$. Repeating this step generates a decreasing chain $\ldots \subset J' \subset J \subset I$ in $\mathbf{I}$ which has a lower bound $\emptyset$. Since there is no element smaller than $\emptyset$, $\emptyset$ is not in $\mathbf{I}$. This contradicts the assumption. Therefore, $\mathbf{T}_P(\{I\}) = \{I\}$. □

The above lemma presents that if $\mathbf{I}$ is a fixpoint of $\mathbf{T}_P$, each element in $\mathbf{I}$ is a model of P (by Lemma 2.2). Now, the ordinal powers of $\mathbf{T}_P$ are defined as follows.

Definition 2.4

$$\begin{array}{lcl} \mathbf{T}_P \uparrow 0 & = & \{\emptyset\} \\ \mathbf{T}_P \uparrow n+1 & = & \mathbf{T}_P(\mathbf{T}_P \uparrow n) \\ \mathbf{T}_P \uparrow \omega & = & lub\{\mathbf{T}_P \uparrow n \mid n < \omega\} \end{array}$$

where n is a successor ordinal and ω is a limit ordinal.

Although the mapping $\mathbf{T}_P$ is not monotonic, the next lemma shows that powers of $\mathbf{T}_P$ by Definition 2.4 are always increasing.

Lemma 2.5 For any ordinal n, $\mathbf{T}_P \uparrow n \sqsubseteq \mathbf{T}_P \uparrow n+1$.

Proof: By definition, for each interpretation I in $\mathbf{T}_P \uparrow n+1$, there exists an interpretation J in $\mathbf{T}_P \uparrow n$ such that $I \in \mathbf{T}_P(\{J\})$ and $J \subseteq I$. If $I = J$, $\mathbf{T}_P(\{J\}) = \{J\}$ by Lemma 2.3 and there is no I' in $\mathbf{T}_P \uparrow n+1$ such that $I' \in \mathbf{T}_P(\{J\})$ and $J \subset I'$. Then after subtracting every such I from $\mathbf{T}_P \uparrow n+1$ and $\mathbf{T}_P \uparrow n$, if I' is in $\mathbf{T}_P \uparrow n+1$, there exists J' in $\mathbf{T}_P \uparrow n$ such that $J' \subset I'$, hence the result follows. □

Lemma 2.6 Let $\{B_1, \ldots, B_m\}$ be any finite set of ground atoms. If $\{B_1, \ldots, B_m\} \subseteq I$ for some $I \in \mathbf{T}_P \uparrow \omega$, then there is a successor ordinal k such that $\{B_1, \ldots, B_m\} \subseteq I' \subseteq I$ for some $I' \in \mathbf{T}_P \uparrow k$.

Proof: Let $I \in \mathbf{T}_P \uparrow \omega$ such that $\{B_1, \ldots, B_m\} \subseteq I$. Suppose to the contrary that no $\mathbf{T}_P \uparrow k$ ($k < \omega$) contains an interpretation I' such that $\{B_1, \ldots, B_m\} \subseteq I' \subseteq I$. Then, for any successor ordinal k, $\mathbf{T}_P \uparrow k \neq \mathbf{T}_P \uparrow \omega$ and $\mathbf{T}_P \uparrow k \sqsubseteq \mathbf{T}_P \uparrow \omega = lub\{\mathbf{T}_P \uparrow n \mid n < \omega\}$. Since $\sqsubseteq$ is a partial order (Lemma 2.1), $\mathbf{T}_P \uparrow \omega \not\sqsubseteq \mathbf{T}_P \uparrow k$ holds. However, as $\mathbf{T}_P \uparrow k \sqsubseteq \mathbf{T}_P \uparrow k+1$ by Lemma 2.6, $lub\{\mathbf{T}_P \uparrow n \mid n \leq k\} = \mathbf{T}_P \uparrow k$ holds. Therefore, $\mathbf{T}_P \uparrow \omega \not\sqsubseteq lub\{\mathbf{T}_P \uparrow n \mid n \leq k\}$ for any k. Hence, $\mathbf{T}_P \uparrow \omega = lub\{\mathbf{T}_P \uparrow n \mid n < \omega\} \not\sqsubseteq lub\{\mathbf{T}_P \uparrow n \mid n < \omega\}$, a contradiction. □

Lemma 2.7 $\mathbf{T}_P(\mathbf{T}_P \uparrow \omega) \sqsubseteq \mathbf{T}_P \uparrow \omega$.

Proof: Let I be any interpretation in $\mathbf{T}_P \uparrow \omega$. Suppose that $I \notin \mathbf{T}_P(\mathbf{T}_P \uparrow \omega)$. Then, for any $J \in \mathbf{T}_P \uparrow \omega$, $I \notin \mathbf{T}_P(\{J\})$. Especially, $I \notin \mathbf{T}_P(\{I\})$. Two cases arise.

Case 1: $\exists J \in \mathbf{T}_P(\{I\})$ such that $I \subset J$. In this case, there is a ground clause $A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m$ from P such that $\{B_1, \ldots, B_m\} \subseteq I$ and $\{A_1, \ldots, A_l\} \cap I = \emptyset$. By Lemma 2.6, there exists a successor ordinal k such that $\{B_1, \ldots, B_m\} \subseteq I_k \subseteq I$ for some $I_k \in \mathbf{T}_P \uparrow k$. Then, since $A_i \notin I_k$ for any A_i, $\exists I' \in \mathbf{T}_P \uparrow k+1$ such that $I' \setminus I \neq \emptyset$. Thus, any interpretation including $\{B_1, \ldots, B_m\}$ in $\mathbf{T}_P \uparrow n$ for any $n < \omega$ takes some A_i at $\mathbf{T}_P \uparrow n+1$. Therefore, I cannot be in $\mathbf{T}_P \uparrow \omega = lub\{\mathbf{T}_P \uparrow n \mid n < \omega\}$, a contradiction.

Case 2: $\mathbf{T}_P(\{I\}) = \emptyset$. In this case, there is a ground negative clause $\leftarrow B_1 \wedge \ldots \wedge B_m$ from P such that $\{B_1, \ldots, B_m\} \subseteq I$. Again, by Lemma 2.6, there exists a successor ordinal k such that $\{B_1, \ldots, B_m\} \subseteq I_k \subseteq I$ for some $I_k \in \mathbf{T}_P \uparrow k$. Then, $\mathbf{T}_P(\{I_k\}) = \emptyset$. Thus, any interpretation including $\{B_1, \ldots, B_m\}$ in $\mathbf{T}_P \uparrow n$ for any $n < \omega$ is pruned away at $\mathbf{T}_P \uparrow n+1$. Therefore, I cannot be in $\mathbf{T}_P \uparrow \omega = lub\{\mathbf{T}_P \uparrow n \mid n < \omega\}$, a contradiction.

Therefore, $I \in \mathbf{T}_P(\mathbf{T}_P \uparrow \omega)$, and hence $\mathbf{T}_P(\mathbf{T}_P \uparrow \omega) \supseteq \mathbf{T}_P \uparrow \omega$. By the definition of $\sqsubseteq$, the result follows. □

By Lemmas 2.5 and 2.7, we have the following theorem.

Theorem 2.8 $\mathbf{T}_P \uparrow \omega$ is a fixpoint.

We call $\mathbf{T}_P \uparrow \omega$ a *disjunctive fixpoint* of P.

Example 2.4 (cont. from Example 2.1) We get $\mathbf{T}_P \uparrow 1 = \{\{c\}\}$, $\mathbf{T}_P \uparrow 2 = \{\{c,d,a\},\{c,d,b\}\}$, and $\mathbf{T}_P \uparrow 3 = \{\{c,d,a\}\}$. Then, $\mathbf{T}_P \uparrow \omega = \mathbf{T}_P \uparrow 3$.

By Lemmas 2.2 and 2.4 and Theorem 2.8, the following result holds.

Lemma 2.9 Each element in $\mathbf{T}_P \uparrow \omega$ is a model of P. □

Furthermore, the next theorem shows that $\mathbf{T}_P \uparrow \omega$ includes the set of *all* minimal models of P. In the following, let $min(\mathbf{I}) = \{ I \in \mathbf{I} \mid \nexists J \in \mathbf{I} \text{ such that } J \subset I \}$.

Theorem 2.10 Let P be a consistent program and $\mathcal{MM}_P$ be the set of all minimal models of P. Then, $\mathcal{MM}_P = min(\mathbf{T}_P \uparrow \omega)$.

Proof: Since $\mathcal{MM}_P \supseteq min(\mathbf{T}_P \uparrow \omega)$ is clear from Lemma 2.9, we show the other direction. Let I be a minimal model of P. Then for each atom A in I, there is a ground clause $A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m$ from P, such that $\{B_1, \ldots, B_m\} \subseteq I$ and $A \leftarrow A_i$ for some i $(1 \leq i \leq l)$. Then, by the definition of fixpoint construction, I is contained in $\mathbf{T}_P \uparrow \omega$. Since each element in $\mathbf{T}_P \uparrow \omega$ is a model of P, I is a minimal element of $\mathbf{T}_P \uparrow \omega$. Hence, $I \in min(\mathbf{T}_P \uparrow \omega)$. □

Theorem 2.10 characterizes the minimal model semantics of a positive disjunctive program in terms of the disjunctive fixpoint of the program.

Example 2.5 Let $P = \{ a \leftarrow b, \quad a \vee b \leftarrow \ \}$. Then $\mathbf{T}_P \uparrow \omega = \{\{a\},\{a,b\}\}$ and $\{a\}$ is the unique minimal model of P.

Corollary 2.11 A program P is contradictory iff $\mathbf{T}_P \uparrow \omega = \emptyset$.

Proof: If P is contradictory, there exists n such that $\mathbf{T}_P \uparrow n = \emptyset$, hence the result follows. □

Recall that a contradictory program has no model different from $\mathcal{L}$. Therefore, the above corollary also characterizes a test for refutability (unsatisfiability) of a set of non-Horn clauses in first-order logic.

Corollary 2.12 If P is a definite program, $\mathbf{T}_P \uparrow \omega$ contains a unique element I which is the least Herbrand model of P. □

That is, for a definite program our fixpoint construction reduces to van Emden and Kowalski's fixpoint semantics [VEK76].

3 Fixpoint Semantics for General Disjunctive Programs

3.1 General Disjunctive Programs

A general disjunctive program is a disjunctive program which contains negation by failure in a program. In this section, we present a fixpoint semantics for general disjunctive programs using the fixpoint operator introduced in the previous section.

A *general disjunctive program* is a finite set of clauses of the form:

$$A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m \wedge not\ B_{m+1} \wedge \ldots \wedge not\ B_n \quad (l \geq 0; n \geq m \geq 0) \quad (2)$$

where A_i's and B_j's are atoms and all variables are assumed to be universally quantified at the front of the clause. An operator *not* preceded each atom B_k $(m+1 \leq k \leq n)$ denotes *negation by failure* [Cla78]. A clause is called *normal* if $l = 1$. A program containing only normal clauses is called a *general logic program*. A program which contains no predicate defined recursively through its negation is called *stratified*. A program reduces to a positive disjunctive program, when $m = n$ (containing no *not*) for every clause. The notion of head, body, and ground clause (program) are defined in the same way as in the previous section.

As for the semantics of general disjunctive programs, we consider the *stable model semantics* of disjunctive programs which was initially introduced by Gelfond and Lifschitz [GL88] for general logic programs.

Definition 3.1 Let P be a general disjunctive program and I be a subset of $\mathcal{L}$. The positive disjunctive program P_I is defined as follows: A clause

$$A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m$$

is in P_I if there is a ground clause of the form (2) from P such that $B_{m+1}, \ldots, B_n \notin I$. Then, if I coincides with a minimal model of P_I, I is called a *stable model* of P.

Note that the above definition is an extension of the original one in the sense that stable models are defined for a program containing disjunctive clauses as well as negative clauses. Similar extensions are also found in [Pr90a]. Especially, when a program is stratified, it has at least one stable model called a *perfect model*.

The set of all stable models of a program P is denoted by $\mathcal{ST}_P$. In particular, if $\mathcal{ST}_P$ is empty, P is called *incoherent*. Note that an incoherent program is different from a contradictory program. By definition, a contradictory program has an inconsistent stable model $\mathcal{L}$. A program is called *consistent* if it is neither incoherent nor contradictory.

Example 3.1 A program $\{\ a \vee b \leftarrow\ ,\quad c \leftarrow not\ c\ \}$ is incoherent, while a program $\{\ a \vee b \leftarrow\ ,\quad \leftarrow a,\quad \leftarrow b\ \}$ is contradictory.

In the next subsection, a program means a general disjunctive program unless stated otherwise.

3.2 Transformation

To characterize the stable model semantics of a general disjunctive program, Inoue et al have proposed a program transformation which transforms a general (disjunctive) logic program into a semantically equivalent *not*-free disjunctive program [IKH92].

Definition 3.2 [IKH92] Let P be a program and $\mathcal{HB}$ be its Herbrand base. Then P^κ is the program obtained as follows.

(i) For each clause $A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m \wedge not\ B_{m+1} \wedge \ldots \wedge not\ B_n$ from P,

$$(A_1 \wedge \neg \mathsf{K}B_{m+1} \wedge \ldots \wedge \neg \mathsf{K}B_n) \vee \ldots \vee (A_l \wedge \neg \mathsf{K}B_{m+1} \wedge \ldots \wedge \neg \mathsf{K}B_n) \vee \mathsf{K}B_{m+1} \vee \ldots \vee \mathsf{K}B_n \leftarrow B_1 \wedge \ldots \wedge B_m$$

is in P^κ. Especially, if $l = 0$, it becomes $\mathsf{K}B_{m+1} \vee \ldots \vee \mathsf{K}B_n \leftarrow B_1 \wedge \ldots \wedge B_m$.

(ii) For each atom A in $\mathcal{HB}$, P^κ contains a clause $\leftarrow \neg \mathsf{K}A \wedge A$.

(iii) Nothing else is in P^κ. [1]

In the above definition, $\mathsf{K}A$ (resp. $\neg \mathsf{K}A$) is a new literal which denotes A *is believed* (resp. *disbelieved*). In the transformation (i), each $not\ B_i$ is rewritten in $\neg \mathsf{K}B_i$ and shifted to the head of the clause. Moreover, since each disjunct A_j in the head becomes true when each $\neg \mathsf{K}B_i$ in the body is true, the condition $\neg \mathsf{K}B_{m+1} \wedge \ldots \wedge \neg \mathsf{K}B_n$ is added in the head of each disjunct A_j. The constraint (ii) states that it cannot happen that A is true and disbelieved at the same time.

A new Herbrand base $\mathcal{HB}^\kappa$ of a transformed program is defined by

$$\mathcal{HB}^\kappa = \mathcal{HB} \cup \{\mathsf{K}A \mid A \in \mathcal{HB}\} \cup \{\neg \mathsf{K}A \mid A \in \mathcal{HB}\}.$$

An interpretation I^κ is now defined as a subset of $\mathcal{HB}^\kappa$. An atom in I^κ is called *objective* if it is in $\mathcal{HB}$, while a newly introduced literal $(\neg)\mathsf{K}A$ is called *subjective*. The set of objective atoms in I^κ is denoted by $obj(I^\kappa)$. Note here that we consider subjective literals not as new formulas in a suitable modal logic, but as newly introduced *atoms* in a program, hence we are still within the classical propositional calculus.

Example 3.2 Let $P = \{\ a \leftarrow not\ b, \quad b \leftarrow not\ a, \quad c \vee d \leftarrow b, \quad c \leftarrow not\ c\ \}$. Then $P^\kappa = \{\ (a \wedge \neg \mathsf{K}b) \vee \mathsf{K}b \leftarrow, \quad (b \wedge \neg \mathsf{K}a) \vee \mathsf{K}a \leftarrow, \quad c \vee d \leftarrow b, \quad (c \wedge \neg \mathsf{K}c) \vee \mathsf{K}c \leftarrow, \quad \leftarrow \neg \mathsf{K}a \wedge a, \quad \leftarrow \neg \mathsf{K}b \wedge b, \quad \leftarrow \neg \mathsf{K}c \wedge c, \quad \leftarrow \neg \mathsf{K}d \wedge d\ \}$.

[1] For efficient computation, an optional rule $\leftarrow \neg \mathsf{K}A \wedge \mathsf{K}A$ can be added for each atom A in $\mathcal{HB}$, but semantically it is not essential.

In [IKH92], it is shown that a bottom up computation of a transformed program produces the set of stable models of the original program. In the following, we characterize the result using the fixpoint operator presented in the previous section. For this purpose, we first modify a mapping presented in Definition 2.2.

Definition 3.3 Let P^κ be a transformed program and I^κ be an interpretation. A mapping $T_{P^\kappa} : 2^{\mathcal{HB}^\kappa} \rightarrow 2^{2^{\mathcal{HB}^\kappa}}$ is defined as follows. Let Γ^i_j be the conjunction of atoms (or just an atom) in $\mathcal{HB}^\kappa$ and $conj(\Gamma^i_j)$ be the set of conjuncts from Γ^i_j. Then

$$T_{P^\kappa}(I^\kappa) = \begin{cases} \emptyset, & \text{if } I^\kappa \models B_1 \wedge \ldots \wedge B_m \text{ for some ground negative clause} \\ & \leftarrow B_1 \wedge \ldots \wedge B_m \text{ from } P^\kappa; \\ \{\ J^\kappa \mid & \text{for each ground clause } C_i : \Gamma^i_1 \vee \ldots \vee \Gamma^i_{l_i} \leftarrow B^i_1 \wedge \ldots \wedge B^i_{m_i} \\ & \text{from } P^\kappa \text{ such that } I^\kappa \models B^i_1 \wedge \ldots \wedge B^i_{m_i} \text{ and } I^\kappa \not\models \Gamma^i_1 \vee \ldots \vee \Gamma^i_{l_i}, \\ & J^\kappa = I^\kappa \cup \bigcup_{C_i} conj(\Gamma^i_j)\ \ (1 \leq j \leq l_i)\ \}, \quad \text{otherwise}. \end{cases}$$

Using this definition, the mapping $\mathbf{T}_{P^\kappa}$ and its disjunctive fixpoint are also defined in the same way as in Section 2.2 and those properties presented there still hold.

Definition 3.4 An interpretation I^κ is called *canonical* if I^κ satisfies the condition: for each ground atom A, if $\mathsf{K}A \in I^\kappa$ then $A \in I^\kappa$.

Definition 3.5 Let $\mathbf{I}^\kappa$ be a set of interpretations. Then

$$obj_c(\mathbf{I}^\kappa) = \{\ obj(I^\kappa) \mid I^\kappa \in \mathbf{I}^\kappa \text{ and } I^\kappa \text{ is canonical }\}.$$

The following theorem presents the fixpoint characterization of the stable model semantics for disjunctive logic programs.

Theorem 3.1 Let P be a consistent program and P^κ its transformed form. Then

$$ST_P = obj_c(min(\mathbf{T}_{P^\kappa} \uparrow \omega)).$$

Proof: Suppose I is in $obj_c(min(\mathbf{T}_{P^\kappa} \uparrow \omega))$. Let I^κ be a canonical set in $min(\mathbf{T}_{P^\kappa} \uparrow \omega)$ such that $obj(I^\kappa) = I$. Then for each ground clause $A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m \wedge not\ B_{m+1} \wedge \ldots \wedge not\ B_n$ from P, if $\{B_1, \ldots, B_m\} \subseteq I^\kappa$, then either (i) $\exists A_i \in I^\kappa$ $(1 \leq i \leq l)$ and $\neg\mathsf{K}B_{m+1}, \ldots, \neg\mathsf{K}B_n \in I^\kappa$, or (ii) $\exists \mathsf{K}B_j \in I^\kappa$ $(m+1 \leq j \leq n)$.

In case of (i), by condition of Definition 3.2(ii), $B_{m+1}, \ldots, B_n \notin I^\kappa$, and hence $B_{m+1}, \ldots, B_n \notin I$. Then there is a clause C: $A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m$ in P_I. Since $\{B_1, \ldots, B_m\} \subseteq I^\kappa$ and $\exists A_i \in I^\kappa$, it holds that $\{B_1, \ldots, B_m\} \subseteq I$ and $\exists A_i \in I$. Therefore, I satisfies the clause C. In case of (ii), since I^κ is canonical, $\exists \mathsf{K}B_j \in I^\kappa$ implies $B_j \in I^\kappa$ and thus $B_j \in I$. In this case, the clause C is not included in P_I. Thus in both cases, I satisfies every clause in P_I. By definition, I is a minimal set

satisfying each clause in P^κ. Then I is also a minimal model of P_I, and hence a stable model of P.

To show the other direction, suppose I is a stable model of P. Let $I = obj(I^\kappa)$ for some $I^\kappa \subseteq \mathcal{HB}^\kappa$. Then for each atom A_i in I, there is a ground clause C in P_I such that $\{B_1, \ldots, B_m\} \subseteq I^\kappa$. In this case, there is a corresponding clause C': $(A_1 \wedge \neg \mathsf{K}B_{m+1} \wedge \ldots \wedge \neg \mathsf{K}B_n) \vee \ldots \vee (A_l \wedge \neg \mathsf{K}B_{m+1} \wedge \ldots \wedge \neg \mathsf{K}B_n) \vee \mathsf{K}B_{m+1} \vee \ldots \vee \mathsf{K}B_n \leftarrow B_1 \wedge \ldots \wedge B_m$ in P^κ such that $\{A_i, \neg \mathsf{K}B_{m+1}, \ldots, \neg \mathsf{K}B_n\} \subseteq I^\kappa$. Since $B_{m+1}, \ldots, B_n \notin I$ implies $B_{m+1}, \ldots, B_n \notin I^\kappa$, I^κ is not pruned by the condition of Definition 3.2(ii), so $I^\kappa \in \mathbf{T}_{P^\kappa} \uparrow \omega$. Then we can choose I^κ to be a minimal set such that $\mathsf{K}B_{m+1}, \ldots, \mathsf{K}B_n \notin I^\kappa$, hence $I^\kappa \in min(\mathbf{T}_{P^\kappa} \uparrow \omega)$ and I^κ is canonical with respect to $B_{m+1}, \ldots, B_n$ $(*)$.

While, assume that if $\mathsf{K}B'_j \in I^\kappa$ for some j $(m+1 \leq j \leq n)$, then there exists a clause C'' in P^κ such that $(A'_1 \wedge \neg \mathsf{K}B'_{m+1} \wedge \ldots \wedge \neg \mathsf{K}B'_n) \vee \ldots \vee (A'_l \wedge \neg \mathsf{K}B'_{m+1} \wedge \ldots \wedge \neg \mathsf{K}B'_n) \vee \mathsf{K}B'_{m+1} \vee \ldots \vee \mathsf{K}B'_n \leftarrow B'_1 \wedge \ldots \wedge B'_m$ and $\{B'_1, \ldots, B'_m\} \subseteq I^\kappa$. Since I^κ is minimal, $A'_i \notin I^\kappa$ for any i $(1 \leq i \leq l)$. Then a clause $A'_i \vee \ldots \vee A'_l \leftarrow B'_1 \wedge \ldots \wedge B'_m$ is not in P_I, hence $\exists j'$ $(m+1 \leq j' \leq n)$ such that $B'_{j'}$ is in I. In this case, we can choose j' such that $j' = j$, then B'_j is in I^κ $(\dagger)$.

By $(*)$ and $(\dagger)$, I^κ is canonical, thus $obj(I^\kappa) \in obj_c(min(\mathbf{T}_{P^\kappa} \uparrow \omega))$. Therefore $I \in obj_c(min(\mathbf{T}_{P^\kappa} \uparrow \omega))$. □

Corollary 3.2 Let P be a program. Then

(i) P is contradictory iff $\mathbf{T}_{P^\kappa} \uparrow \omega = \emptyset$.

(ii) P is incoherent iff $\mathbf{T}_{P^\kappa} \uparrow \omega \neq \emptyset$ and each element in $min(\mathbf{T}_{P^\kappa} \uparrow \omega)$ is not canonical. □

Example 3.3 (cont. from Example 3.2) It is easily verified that only $\{\mathsf{K}c, \mathsf{K}b, \neg \mathsf{K}a, b, c\}$ is a canonical set which is minimal in the fixpoint closure, then $\{b, c\}$ is the unique stable model of the original program.

For general logic programs, the result of Theorem 3.1 is further simplified as follows.

Corollary 3.3 Let P be a general logic program and P^κ be its transformed form. Then

$$\mathcal{ST}_P = obj_c(\mathbf{T}_{P^\kappa} \uparrow \omega).$$

Proof: Clearly, $I \in obj_c(min(\mathbf{T}_{P^\kappa} \uparrow \omega))$ implies $I \in obj_c(\mathbf{T}_{P^\kappa} \uparrow \omega)$. Then we show that the converse is also true. Assume that the converse does not hold. That is, there is a non-minimal set $I \in obj_c(\mathbf{T}_{P^\kappa} \uparrow \omega)$ and $\exists J \in obj_c(min(\mathbf{T}_{P^\kappa} \uparrow \omega))$ such that $J \subset I$. In this case, there exists an atom A such that $A \in I$ and $A \notin J$. Put $I = obj(I^\kappa)$ and $J = obj(J^\kappa)$. Then, there exists a clause

$$(A \wedge \neg \mathsf{K}B_{m+1} \wedge \ldots \wedge \neg \mathsf{K}B_n) \vee \mathsf{K}B_{m+1} \vee \ldots \vee \mathsf{K}B_n \leftarrow B_1 \wedge \ldots \wedge B_m$$

in P^κ, where $I^\kappa, J^\kappa \models B_1, \ldots, B_n$ and $\exists \mathsf{K} B_i$ $(m+1 \leq i \leq n)$ in J^κ. Since J^κ is canonical, $B_i \in J^\kappa$, hence $B_i \subset I^\kappa$. But $\neg \mathsf{K} B_i$ is also in I^κ, this is impossible by the condition of Definition 3.2(ii). □

3.3 Extended Disjunctive Programs

An *extended disjunctive program* is a disjunctive program which contains *classical negation* as well as negation by failure in the program [GL91]. The definition of an extended disjunctive program is the same with that of a general disjunctive program except that each clause in a program has the following form:

$$L_1 \vee \ldots \vee L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m \wedge not\ L_{m+1} \wedge \ldots \wedge not\ L_n \quad (n \geq m \geq l \geq 0) \qquad (3)$$

where each L_i is a positive or negative literal. [2] Especially, if a program contains no disjunctive clause, it is just called an *extended logic program.*

The semantics of an extended disjunctive program is given by the notion of *answer sets.* Considering each negative literal in an extended disjunctive program as a newly introduced atom, an answer set of a program is defined in the same manner with a stable model given in the previous section. The collection of all answer sets of a program is denoted by $\mathcal{AS}_P$. Especially, if $\mathcal{AS}_P$ is empty, a program is called *incoherent*, while if an answer set S contains a pair of complementary literals A and $\neg A$, $S = \mathcal{L}$. A contradictory program has a unique answer set $\mathcal{L}$. Note here that in an extended disjunctive program, contradiction may be caused not only by negative clauses, but also negative heads of clauses in a program.[3]

The answer set semantics of an extended disjunctive program is a direct extension of the stable semantics, and the results presented in the previous section still hold here. The only extra requirement is that, for an extended disjunctive program P, we include a constraint

$$\leftarrow A \wedge \neg A$$

for each negative literal $\neg A$ in the transformed program P^κ.

Theorem 3.4 Let P be a consistent extended disjunctive program and $\mathcal{AS}_P$ be its answer sets. Then, $\mathcal{AS}_P = obj_c(min(\mathbf{T}_{P^\kappa} \uparrow \omega))$. □

Corollary 3.5 Let P be an extended disjunctive program. Then

(i) P is contradictory iff $\mathbf{T}_{P^\kappa} \uparrow \omega = \emptyset$.

[2] [GL91] uses the connective "|" instead of $\vee$ to distinguish its meaning from the classical first order logic. But we abuse the connective $\vee$ as far as no confusion arises.

[3] Note also that the meaning of clauses $\neg a \leftarrow$ and $\leftarrow a$ are different under the answer set semantics. In fact, an answer set of the first clause is $\{\neg a\}$, while the second is $\emptyset$. This difference is due to the fact that the connective $\leftarrow$ is non-contrapositive in a program.

(ii) P is incoherent iff $\mathbf{T}_{P^\kappa} \uparrow \omega \neq \emptyset$ and each element in $min(\mathbf{T}_{P^\kappa} \uparrow \omega)$ is not canonical. □

Corollary 3.6 Let P be a consistent extended logic program and $\mathcal{AS}_P$ its answer sets. Then, $\mathcal{AS}_P = obj_c(\mathbf{T}_{P^\kappa} \uparrow \omega)$. □

4 Comparison with Other Approaches

A fixpoint semantics for disjunctive programs has been studied by several researchers. An early approach to provide the fixpoint semantics for positive disjunctive programs was given by Minker and Rajasekar. In [MR90], they introduced an *extended Herbrand base* that is the set of all ground positive disjunctions from a program, and defined the notion of a *state* as a set of positive disjunctions from the extended base. Then they have developed a fixpoint operator which operates on states and shown that its least fixpoint contains positive clauses which are true in every minimal model of a program. Comparing their approach with ours, both of them characterize the minimal model semantics of a disjunctive program in terms of a fixpoint operator, but the approaches are basically different. Their fixpoint operator computes a minimal state at every stage of closure computation and finally computes minimal models from the model state of the least fixpoint. On the other hand, our fixpoint construction is based upon case-splitting of disjunctions and directly computes a set of models, and the minimality is checked only at the final stage of the computation. Further, our fixpoint construction is still based upon the manipulation of standard Herbrand interpretations and does not require any extension of the Herbrand base. The state based semantics has also been developed for stratified disjunctive programs in [RM89] and further extended to general disjunctive programs by Ross [Ro89], Baral et al [BLM90] and Przymusinski [Pr90b] in the context of the extended well-founded semantics.

Fernandez and Minker [FM91] have also presented a fixpoint semantics of stratified disjunctive programs using a fixpoint operator over sets of interpretations. According to [FM91], their fixpoint operator $T_P^M(\mathcal{I})$ is defined as follows.

$$
\begin{aligned}
T_P^M(\mathcal{I}) &= min(\bigcup_{I \in \mathcal{I}} models_I(state_P(I))), \quad \text{where} \\
state_P(I) &= \{\, A_1 \vee \ldots \vee A_l \mid \text{there is a ground clause from } P \text{ of the form (2)} \\
&\qquad \text{such that } \{B_1, \ldots, B_m\} \subseteq I \text{ and } \{B_{m+1}, \ldots, B_n\} \cap I = \emptyset \,\}, \text{ and} \\
models_I(S) &= \{\, M \subseteq \mathcal{HB} \mid M \text{ is a model of } S \cup I \,\}.
\end{aligned}
$$

With this fixpoint operator, they have defined ordinal powers as usual and shown that its iterative fixpoint characterizes the perfect models of a stratified disjunctive program. One important difference between their fixpoint operator and ours is that, as is observed in the above definition, their fixpoint operator is defined over sets of

minimal interpretations which are ordered under the *Hoare* ordering.[4] To this end, their fixpoint operator has to compute minimal sets of atoms at every stage of closure computation which is very expensive. On the other hand, our fixpoint operator are defined over sets of interpretations ordered by $\sqsubseteq$ and computes minimal sets only at the final stage of closure computation. This has the computational advantage that each interpretation can be treated in a different, independent process in closure computation so that split interpretations can be taken as the source for exploiting OR-parallelism. In this way, the *model generation theorem prover* MGTP [FH91] is implemented on a parallel inference machine for both testing refutability of a program (by Corollary 2.11) and generating the minimal models (by Theorem 2.10).

In [FLMS91, FM92], the result is further extended to non-stratified general disjunctive programs in which they have developed a method of computing stable models by transforming a general disjunctive program into a stratified disjunctive program with integrity constraints. Their *evidential transformation* transforms each general disjunctive clause (2): $A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m \wedge not\ B_{m+1} \wedge \ldots \wedge not\ B_n$ into the positive disjunctive clause

$$A_1 \vee \ldots \vee A_l \vee \mathcal{E}B_{m+1} \vee \ldots \vee \mathcal{E}B_n \leftarrow B_1 \wedge \ldots \wedge B_m\,.$$

They also have new rules $\mathcal{E}A \leftarrow A$ in a transformed program and integrity constraints $A \Leftarrow \mathcal{E}A$ for each atom $A \in \mathcal{HB}$. At a glance, their transformation is quite similar to ours, and the new rule and the integrity constraint respectively seem to correspond to the clause $\leftarrow A \wedge \neg \mathsf{K}A$ and the canonical condition in our approach. In fact, our transformation was independently developed for parallel computation of stable models using the MGTP in [IKH92]. However, from the computational aspect of a fixpoint, there is an essential difference between the two. As is presented in Section 3.2, in our transformation each disjunct A_i in the head has its prerequisite condition $\bigwedge_j \neg \mathsf{K}B_j$ $(m+1 \leq j \leq n)$ in an explicit way, while this is not the case in the evidential transformation. Let us show the effect of this difference using an example. Consider the following program P, its transformed program P' in our approach, and the program P'' obtained from P by their evidential transformation:

$$\begin{aligned} P &= \{\ a \leftarrow not\ a\ \}\,, \\ P' &= \{\ (a \wedge \neg \mathsf{K}a) \vee \mathsf{K}a \leftarrow\ ,\quad \leftarrow a \wedge \neg \mathsf{K}a\ \}\,, \\ P'' &= \{\ a \vee \mathcal{E}a \leftarrow\ ,\quad \mathcal{E}a \leftarrow a\ \}\,. \end{aligned}$$

Then in P' a negative clause blocks to expand the first disjunct of the first clause and we can immediately know that P' has no canonical model. In P'', on the other

[4]The "Hoare" ordering $\leq_h$ in [FM91] is defined as: $\mathbf{I} \leq_h \mathbf{J}$ if $\forall J \in \mathbf{J}$, $\exists I \in \mathbf{I}$ such that $I \subseteq J$. This ordering is, however, usually called the *Smyth* ordering. Note that $\leq_h$ is not a partial order.

hand, the first disjunct is expanded by the second clause and after computing minimal elements in the fixpoint, they know that the program has no stable model. That is, in our transformation an extra prerequisite condition works as a constraint to reduce the number of extensions, and negative clauses are effectively used to prune away improper extensions to avoid unnecessary expansions during the computation of a fixpoint. On the other hand, the evidential transformation does not use any such constraint nor negative clause, instead, it again depends heavily upon the computation of minimal interpretations which plays an essential part to compute the stable models of a program. Moreover, to compute stable models of general logic programs, Corollary 3.3 presents that our fixpoint operator does not need any computation of minimal models at all. These observation presents that our transformation has the advantage of computational efficiency in general and is easily realized in parallel. The usage of negative clauses is also useful to treat classical negation in extended disjunctive programs. For related work, interpretation based fixpoint semantics is also presented in [Fu91] for positive disjunctive programs.

Decker [Dec92] has also developed a fixpoint semantics of disjunctive programs. His fixpoint semantics is presented for only positive disjunctive programs. While his fixpoint operator maps a disjunction of interpretations into a disjunction of interpretations, each disjunction can also be interpreted as a set of atoms as our approach. The most important difference lies in the fact that while his operator computes the set of all *supported models* of a program,[5] ours does not compute all of them. Moreover, while Decker also permits negative clauses in a program, they are not used for pruning interpretations during computation and his fixpoint contains inconsistent interpretations in general. On the other hand, every interpretation contained in our disjunctive fixpoint is a supported model. As a result, our fixpoint construction avoids many unnecessary expansions for constructing minimal models. For example, suppose that a program contains the clause $a \vee b \vee c \leftarrow d$. Then, for $\mathbf{I} = \{\{c, d\}\}$, Decker's fixpoint operator maps $\mathbf{I}$ to $\{\{a, c, d\}, \{b, c, d\}, \{c, d\}\}$ and a further application adds $\{a, b, c, d\}$ to it, while ours never expands $\mathbf{I}$. This is because Decker's operator always expands an interpretation I by each disjunct from the head of a clause C whenever I satisfies the body of C, while our fixpoint operator $\mathbf{T}_P$ incorporates the test for *violatedness* of C in I (Definition 2.2). In fact, our fixpoint construction directly characterizes closure computation of model generation theorem provers with violatedness checking such as SATCHMO [MB88] and MGTP [FH91].

Reed et al [RLS91] have developed yet another fixpoint characterization of a positive disjunctive program. Their fixpoint construction is based upon the case-analysis

[5]Here, the notion of a supported model is different from that of [ABW88], but equivalent to the notion of a *possible model* in [Sa89]. For constructing minimal models, Decker also defines another fixpoint operator which is similar to that of [FM91] and again requires minimality checking at every stage of closure computation.

of disjunctions and, different from ours, two kinds of fixpoint operators, called case and join, are used to compute logical consequences of a disjunctive program. Ross and Topor [RT88] have also given a fixpoint construction for positive disjunctive programs to characterize the semantics of the inference rule called *disjunctive database rule* (DDR), but they concern about only negation in a program and do not discuss any connection to the model theoretical meaning of a program.

For general and extended disjunctive programs, the stable model semantics were extended to the answer set semantics [GL91] and the three-valued stable semantics [Pr90a]. However, as is the case for the stable model semantics, their fixpoint computation is not constructively given and is computationally expensive in general. On the other hands, our fixpoint computation is constructively defined and its computational complexity is the same as that of computing the minimal models of a positive disjunctive program. Further, our fixpoint computation is also different from the constructive approach by [SZ90, Fa90], because our fixpoint construction is performed in parallel based on case-splitting of disjunctions and does not need any selection strategies nor future backtracking during the computation of stable models.

5 Conclusion

We have presented a uniform framework of fixpoint characterization of disjunctive and general logic programs. First, we have developed a fixpoint operator which operates on a complete lattice consisting of sets of Herbrand interpretations, and shown that the minimal element of its fixpoint closure coincides with the set of minimal models of a positive disjunctive program. Next, we have provided a suitable program transformation of general disjunctive programs, showing that the stable model semantics is characterized by the fixpoint closure of the transformed program. The result is also directly applied to the answer set semantics of extended disjunctive programs.

The disjunctive fixpoint semantics presented in this paper is a direct extension of van Emden and Kowalski's fixpoint semantics for definite Horn programs. It provides a uniform framework for not only disjunctive programs but also general and extended logic programs. Compared with other approaches, our fixpoint theory is different from the state based semantics by Minker and Rajasekar and also has a computational advantage over Fernandez et al's fixpoint construction. For a procedural aspect of our fixpoint semantics, a bottom-up model generation theorem prover called MGTP is developed at ICOT which is sound and complete to compute stable models for range-restricted disjunctive programs [IKH92].

References

[ABW88] Apt, K.R., Blair, H.A. and Walker, A., Towards a theory of declarative knowledge, in *Foundations of Deductive Databases and Logic Programming* (J. Minker ed.), 89-148, Morgan Kaufmann, 1988.

[BLM90] Baral, C., Lobo, J. and Minker, J., Generalized disjunctive well-founded semantics for logic programs, Research Report CS-TR-2436, Dept. of Computer Science, Univ. of Maryland, 1990.

[Cla78] Clark, K.L., Negation as failure, in *Logic and Data Bases* (H. Gallaire and J. Minker eds.), 293-322, Plenum, 1978.

[Dec92] Decker, H., Foundations of first-order databases, Research Report, Siemens, 1992.

[Fa90] Fages, F., A new fixpoint semantics for general logic programs compared with the well-founded and the stable model semantics, *Proc. 7th Int. Conf. on Logic Programming*, 442-458, 1990.

[FM91] Fernandez, J.A. and Minker, J., Computing perfect models of disjunctive stratified databases, *Proc. ILPS Workshop on Disjunctive Logic Programs*, 1991.

[FM92] Fernandez, J.A. and Minker, J., Disjunctive deductive databases, *Proc. Logic Programming and Automated Reasoning*, Lecture Notes in Artificial Intelligence 624, Springer-Verlag, 1992.

[FLMS91] Fernandez, J.A., Lobo, J., Minker, J. and Subrahmanian, V.S., Disjunctive LP + integrity constraints = stable model semantics, *Proc. ILPS Workshop on Deductive Databases*, 110-117, 1991.

[FH91] Fujita, H. and Hasegawa, R. A model generation theorem prover in KL1 using a ramified-stack algorithm, *Proc. 8th Int. Conf. on Logic Programming*, 535–548, 1991.

[Fu91] Furbach, U., Computing answers for disjunctive logic programs, *Proc. ILPS Workshop on Disjunctive Logic Programs*, 1991.

[GL88] Gelfond, M. and Lifschitz, V., The stable model semantics for logic programming, *Proc. 5th Int. Conf. Symp. on Logic Programming*, 1070-1080, 1988.

[GL91] Gelfond, M. and Lifschitz, V., Classical negation in logic programs and disjunctive databases, *New Generation Computing* 9:365-385, 1991.

[IKH92] Inoue, K., Koshimura, M. and Hasegawa, R., Embedding negation as failure into a model generation theorem prover, *Proc. 11th Int. Conf. on Automated Deduction*, Lecture Notes in Artificial Intelligence 607, 400-415, Springer-Verlag, 1992.

[Li87] Lloyd, J.W., *Foundations of Logic Programming*, 2nd Edition, Springer-Verlag, 1987.

[MB88] Manthey, R. and Bry, F., SATCHMO: a theorem prover implemented in Prolog, *Proc. 9th Int. Conf. on Automated Deduction*, Lecture Notes in Computer Science 310, 415–434, Springer-Verlag, 1988.

[Mi82] Minker, J., On indefinite data bases and the closed world assumption, *Proc. 6th Int. Conf. on Automated Deduction*, Lecture Notes in Computer Science 138, 292-308, Springer-Verlag, 1982.

[MR90] Minker, J. and Rajasekar, A., A fixpoint semantics for disjunctive logic programs, *J. Logic Programming* 9:45-74, 1990.

[Pr88] Przymusinski, T.C., On the declarative semantics of deductive databases and logic programs, in *Foundations of Deductive Databases and Logic Programming* (J. Minker ed.), 193-216, Morgan Kaufmann, 1988.

[Pr89] Przymusinski, T.C., Every logic program has a natural stratification and an iterated least fixed point model, *Proc. 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 11-21, 1989.

[Pr90a] Przymusinski, T.C., Extended stable semantics for normal and disjunctive logic programs, *Proc. 7th Int. Conf. on Logic Programming*, 459-477, 1990.

[Pr90b] Przymusinski, T.C., Stationary semantics for disjunctive logic programs and deductive databases, *Proc. North American Conf. on Logic Programming*, 40-62, 1990.

[Re78] Reiter, R., On closed world databases, in *Logic and Data Bases* (H. Gallaire and J. Minker eds.), 55-76, Plenum, 1978.

[RLS91] Reed, D.W., Loveland, D.W. and Smith, B.T., An alternative characterization of disjunctive logic programs, *Proc. of Int. Logic Programming Symp.*, 54-68, 1991.

[Ro89] Ross, K., The well founded semantics for disjunctive logic programs, *Proc. 1st Int. Conf. on Deductive and Object-Oriented Databases*, 352-369, 1989.

[RM89] Rajasekar, A. and Minker, J., A stratification semantics for general disjunctive programs, *Proc. North American Conf. on Logic Programming*, 573-586, 1989.

[RT88] Ross, K.A. and Topor, R.W., Inferring negative information from disjunctive databases, *J. Automated Reasoning* 4:397-424, 1988.

[Sa89] Sakama, C., Possible model semantics for disjunctive databases, *Proc. 1st Int. Conf. on Deductive and Object-Oriented Databases*, 337-351, 1989.

[SZ90] Sacca, D. and Zaniolo, C., Stable models and non-determinism in logic programs with negation, *Proc. 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 205-229, 1990.

[VEK76] van Emden, M.H. and Kowalski, R.A., The semantics of predicate logic as a programming language, *J.ACM* 23(4):733-742, 1976.

[VG89] van Gelder, A., The alternating fixpoint of logic programs with negation, *Proc. 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1-10, 1989.

[VRS91] van Gelder, A., Ross, K. and Schlipf, J.S., The well-founded semantics for general logic programs, *J. ACM* 38(3):620-650, 1991.