Possible Model Semantics for Disjunctive Databases*

Chiaki Sakama

ASTEM Research Institute of Kyoto
17 Chudoji Minami-machi, Shimogyo, Kyoto 600, Japan
sakama@astem.or.jp

Abstract

This paper presents a novel approach to the semantics of deductive databases. The *possible model semantics* is introduced as an alternative approach to the classical minimal model semantics. The possible model semantics can distinguish both inclusive and exclusive disjunctions, and provide a flexible mechanism for inferring negation in disjunctive databases. The possible model semantics is characterized by a new fixpoint semantics of disjunctive databases. A proof procedure called the SLD_P*-resolution* is presented and shown to be sound and complete with respect to the possible model semantics.

1 Introduction

The declarative semantics of logic programming and deductive databases has been widely studied based on the minimal model semantics by incorporating an inference rule for negation. For definite Horn databases, the unique *least Herbrand model* gives a declarative meaning of a program [VK76], and the *closed world assumption (CWA)* [Rei78] provides negative information as the facts false in this model.

When a database contains some non-Horn clauses, the least Herbrand model does not exist in general and some *preferred* models are often introduced as the intended meaning of the database. For example, *supported models* [ABW88] or *perfect models* [Prz88] are the intended models for stratified databases, and the *iterated CWA (ICWA)* [GPP89] provides negative information as the complement of such a model. These results are extended to the *extended CWA (ECWA)* [GPP89], which is also relating to the *circumscription* [Mc80] in AI.

By contrast, the so-called *disjunctive databases* are indefinite databases which usually have multiple *minimal models*. For inferring negation from

*This is a revised version of the paper which is in the *Proceedings of the 1st International Conference on Deductive and Object-Oriented Databases (DOOD'89)*, North-Holland, pp. 369–383, 1989.

such databases, two alternative rules are known. The first approach considers the collection of minimal models and the facts which are not true in any minimal model are assumed to be false by the *generalized CWA (GCWA)* [Min82, YH85]. On the other hand, the other approach which is *not* based on the minimal model semantics is the *disjunctive database rule (DDR)* [RT88] or equivalently the *weak GCWA (WGCWA)* [RLM89].

Both the GCWA and the DDR provide simple and powerful mechanisms for inferring negation, however, each rule also has some drawbacks. For example, consider the sentence, "$S_1 : love(Scarlet, Charles) \vee love(Scarlet, Ashley)$" (*Scarlet loves Charles or Ashley*). If we know $love(Scarlet, Charles)$ is true, $\sim love(Scarlet, Ashley)$ is inferred by the GCWA. That is, the GCWA does not allow the possibility of Scarlet's loving both, which seems too strong. By contrast, the DDR still leaves the possibility of holding alternative conclusion and does not infer $\sim love(Scarlet, Ashley)$. On the other hand, consider another sentence, "$S_2 : marry(Scarlet, Charles) \vee marry(Scarlet, Ashley)$" (*Scarlet marries Charles or Ashley*). In this situation, it seems natural that knowing $marry(Scarlet, Charles)$ implies the negation of $marry(Scarlet, Ashley)$ by the GCWA.

Thus, the GCWA usually interprets disjunctions in an *exclusive* manner, while the DDR interprets disjunctions in an *inclusive* manner. Then the problem is that in the absence of a single uniform framework, one has to use these separate rules to treat both exclusive and inclusive disjunctions in the same program. (Consider a database containing both S_1 and S_2.) To distinguish both exclusive and inclusive disjunctions, negative clauses are useful to express exclusive disjunctions [Kow79][1]. In the above example, by adding the clause " $\sim marry(Scarlet, Charles) \vee \sim marry(Scarlet, Ashley)$", we get the intended meaning of the database.

This paper presents a semantics of disjunctive databases which can distinguish both inclusive and exclusive disjunctions in a database. In Section 2, we introduce the class of *possible models* which is an extension of minimal models, and characterize it by giving a new fixpoint semantics of disjunctive databases. We also introduce a rule, *GCWA for the possible model semantics ($GCWA_P$)* for inferring negation from databases. It is shown that the $GCWA_P$ can provide a flexible mechanism for inferring negation compared with the GCWA and the DDR. The results are extended to stratified disjunctive databases in Section 3. Section 4 provides a proof procedure, called SLD_P-resolution, which is a modification of SLD-resolution. It is shown that the procedure is sound and complete with respect to the possible model semantics.

2 Possible Model Semantics for Disjunctive Databases

2.1 Possible Models

First, we define databases considered in this section.

[1] This is also suggested in [RT88, Section 9].

A *positive disjunctive database* or simply a *database* is a finite set of clauses of the form:

$$A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n$$

where $m, n \geq 0$ and each A_i and B_j are atoms, and all variables are universally quantified at the front of the clause. $A_1 \vee \ldots \vee A_m$ is called the *head*, and $B_1 \wedge \ldots \wedge B_n$ is called the *body* of the clause. When $m > 1$ (resp. $m = 0$), the clause is called *disjunctive* (resp. *negative*). A *ground database* is a database whose variables are instantiated to the elements of the Herbrand universe in every possible way. A ground database is possibly infinite.

An *interpretation* of a program is a subset of the Herbrand base of the program. An interpretation I is called a *minimal model* of a database D if there is no smaller interpretation J satisfying the database. A database is *consistent* if it has a minimal model, otherwise it is *inconsistent*. The *minimal model semantics* [Min82] of a disjunctive database D is defined as the set of all minimal models of D. Note that a database possibly becomes inconsistent in the presence of negative clauses. In this paper, we assume a database to be consistent unless stated otherwise.

Next we introduce the notion of *split databases*.[2]

Definition 2.1 Let D be a database and C be a ground instance of a disjunctive clause in D of the form:

$$C : A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n.$$

Then C is *split* into $2^m - 1$ sets of clauses $C_1, \ldots, C_{2^m-1}$ where

$$C_j = \{\, A_i \leftarrow B_1 \wedge \ldots \wedge B_n \mid A_i \in S \ \text{ where } S \in 2^{\{A_1,\ldots,A_m\}} \setminus \{\emptyset\} \,\}.$$

A *split database* of D is a ground database in which each disjunctive clause C is replaced by some C_j $(1 \leq j \leq 2^m - 1)$. $\square$

By definition, a split database is a set of ground Horn clauses. If a ground database contains k disjunctive clauses where each has m_i atoms in the head, then the database is split into at most $\prod_{i=1}^{k}(2^{m_i}-1)$ different databases.

Example 2.1 Let $D = \{\ a \vee b \leftarrow c, \quad d \leftarrow b, \quad c \leftarrow, \quad \leftarrow a \wedge d\ \}$. The clause $a \vee b \leftarrow c$ is split into $C_1 = \{\, a \leftarrow c \,\}$, $C_2 = \{\, b \leftarrow c \,\}$ and $C_3 = \{\, a \leftarrow c, \quad b \leftarrow c \,\}$. Then D is split into the following three databases:

$$\begin{aligned} D_1 &= \{\ a \leftarrow c,\ \ d \leftarrow b,\ \ c \leftarrow,\ \ \leftarrow a \wedge d\ \}, \\ D_2 &= \{\ b \leftarrow c,\ \ d \leftarrow b,\ \ c \leftarrow,\ \ \leftarrow a \wedge d\ \}, \\ D_3 &= \{\ a \leftarrow c,\ \ b \leftarrow c,\ \ d \leftarrow b,\ \ c \leftarrow,\ \ \leftarrow a \wedge d\ \}. \quad \square \end{aligned}$$

Lemma 2.1 Let D be a database. Then M is a model of D iff there is a split database D' of D where M is a model of D'.

[2]The notion of split databases is also introduced in [Lov78] in a different context.

Proof: ($\Rightarrow$) Suppose that M is a model of D. Then, for every ground instance C of a clause in D, M satisfies C. Now let $C : A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n$. If $M \models A_1 \vee \ldots \vee A_m$, there is an A_i $(1 \leq i \leq m)$ such that $M \models A_i$. In this case, there exists a split database D' of D which contains $A_i \leftarrow B_1 \wedge \ldots \wedge B_n$, and M is a model of D'. Else if $M \not\models A_1 \vee \ldots \vee A_m$, then $M \not\models B_1 \wedge \ldots \wedge B_n$. In this case, M also satisfies a split clause $A_i \leftarrow B_1 \wedge \ldots \wedge B_n$ in D'. Hence, M is a model of D'.

($\Leftarrow$) Suppose that M is a model of D'. Since D' subsumes the ground database of D, the result immediately holds. □

Now we define the notion of possible models.

Definition 2.2 Let D be a database and M be a model of D. Then M is called a *possible model* of D if M is a minimal model of a split database of D. □

The following proposition says that a minimal model is a possible model.

Proposition 2.2 Let D be a database. If M is a minimal model of D, it is a possible model of D. Conversely, a minimal possible model is a minimal model of D.

Proof: By Lemma 2.1, a minimal model M of D is a model of some split database D' of D. Assume that there is a model M' of D' such that $M' \subseteq M$. By Lemma 2.1, M' is a model of D. Since M is minimal, $M' = M$. Conversely, for a minimal possible model M, assume that there is a minimal model M' of D such that $M' \subseteq M$. By the above proof, M' is a possible model, therefore $M' = M$. □

The possible models are different from the minimal models in general.

Example 2.2 Let $D = \{\ a \vee b \leftarrow c,\ \ c \leftarrow\ \}$. Then D has three possible models: $\{a, c\}$, $\{b, c\}$, and $\{\ a, b, c\ \}$, however, the last possible model is not a minimal model. □

Intuitively, a possible model represents a set of possible facts that might be true in the actual world. In the above example, D has three possible models which respectively represent exclusive and inclusive interpretations of disjunctions. Note that the minimal model semantics excludes the non-minimal inclusive interpretation of the database. In this sense, possible models can provide more flexible meaning of disjunctions than the minimal model semantics. In definite Horn databases, possible models clearly reduce to the least Herbrand model. Proposition 2.2 implies the following properties.

Proposition 2.3 A consistent database has a possible model. □

Proposition 2.4 Let D be a database and A be a ground atom. Then, $D \models A$ iff A is true in every possible model. □

2.2 Fixpoint Semantics

A fixpoint semantics for disjunctive databases has recently been proposed by Lobo, Minker and Rajasekar [LMR89]. They have developed a new fixpoint operator which operates on a set of positive disjunctions (called state). In this section, we give yet another fixpoint operator which operates on a set of interpretations. Then we present a new fixpoint semantics of a disjunctive database and characterize the possible model semantics introduced in the previous section. First, we define a mapping T_P from an Herbrand interpretation to a set of Herbrand interpretations. In the following, HB denotes the Herbrand base.

Definition 2.3 Let D be a database and I be an interpretation of D. Then a mapping $T_D : 2^{HB} \to 2^{2^{HB}}$ is defined as follows.

$$T_D(I) = \{\, J \mid \; J \text{ satisfies every negative clause in } D \text{ and for every non-negative ground clause } C_i : A_1 \vee \ldots \vee A_{m_i} \leftarrow B_1 \wedge \ldots \wedge B_{n_i} \text{ from } D \text{ such that } I \models B_1 \wedge \ldots \wedge B_{n_i},\; J = I \cup \bigcup_{C_i} \{A_j\}\; (1 \leq j \leq m_i) \,\} \quad \square$$

Example 2.3 Let $D = \{\, a \vee b \leftarrow c, \quad d \leftarrow c, \quad \leftarrow b \wedge d, \quad c \leftarrow \,\}$. Then, $T_D(\{c\}) = \{\{c, d, a\}, \{c, d, b\}\}$ and $T_D(\{b, c, d\}) = \emptyset$. $\square$

Next we define a mapping over the set of Herbrand interpretations.

Definition 2.4 Let D be a database and $\mathbf{I}$ be a set of interpretations. A mapping $\mathbf{T} : 2^{2^B} \to 2^{2^B}$ is defined as

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

In particular, $\mathbf{T}(\emptyset) = \emptyset$. $\square$

Example 2.4 Consider the database in the previous example. Then, $\mathbf{T}(\{\{c\}, \{\, b,\, d\,\}\}) = \{\{c, d, a\}, \{c, d, b\}\}$. $\square$

By definition, $\mathbf{I} \subseteq \mathbf{J}$ implies $\mathbf{T}(\mathbf{I}) \subseteq \mathbf{T}(\mathbf{J})$, therefore the mapping $\mathbf{T}$ is monotonic.On the other hand, T_D is not monotonic. For example, let $D = \{\, a \leftarrow b, \quad \leftarrow c \,\}$. Then, $T_D(\{b\}) = \{\{a, b\}\}$ and $T_D(\{b, c\}) = \emptyset$. Thus, $\{b\} \subseteq \{b, c\}$ does not imply $T_D(\{b\}) \subseteq T_D(\{b, c\})$.

Definition 2.5 Let D be a database. Then the ordinal powers of $\mathbf{T}$ are defined as follows:

$$\begin{array}{lcl} \mathbf{T} \uparrow 0 & = & \{\emptyset\}, \\ \mathbf{T} \uparrow n+1 & = & \mathbf{T}(\mathbf{T} \uparrow n), \\ \mathbf{T} \uparrow \omega & = & \bigcup_{\alpha<\omega} \bigcap_{\alpha \leq n < \omega} \mathbf{T} \uparrow n, \end{array}$$

where n is a successor ordinal and ω is a limit ordinal. $\square$

The above definition says that at the limit ordinal ω the closure retains interpretations which are persistent in the preceding iterations.

Theorem 2.5 Let D be a database. Then, $\mathbf{T} \uparrow \omega$ is a fixpoint.

Proof: When $I \in \mathbf{T} \uparrow \omega$, suppose that there is no interpretation J in $\mathbf{T} \uparrow \omega$ such that $I \in \mathbf{T}(\{J\})$. In this case, for any α there is some n ($\alpha \leq n < \omega$) such that J is not included in $\mathbf{T} \uparrow n$. Then $I \notin \mathbf{T} \uparrow n+1$. This contradicts the fact that $I \in \mathbf{T} \uparrow \omega$. Thus, $J \in \mathbf{T} \uparrow \omega$, so $I \in \mathbf{T}(\mathbf{T} \uparrow \omega)$. Conversely, if $I \in \mathbf{T}(\mathbf{T} \uparrow \omega)$, there is an interpretation J in $\mathbf{T} \uparrow \omega$ such that $I \in \mathbf{T}(\{J\})$. Then J is included in any $\mathbf{T} \uparrow n$ for $\alpha \leq n < \omega$ by definition. Thus $I \in \mathbf{T} \uparrow n$ for any $\alpha + 1 \leq n < \omega$. Hence, $I \in \mathbf{T} \uparrow \omega$. □

In definite Horn databases, the fixpoint $\mathbf{T} \uparrow \omega$ reduces to the least fixpoint of [VK76].

Definition 2.6 Let D be a database. Then the *selective-fixpoint* Φ_D is defined as:

$$\Phi_D = \{\ I \mid I \in \mathbf{T} \uparrow \omega \text{ and } I \in \mathbf{T}(\{I\})\ \}. \quad \square$$

Thus, Φ_D is the set of interpretations selected from the fixpoint so that each element in the set is kept by the mapping $\mathbf{T}$.

Lemma 2.6 Let D be a database and I be an interpretation of D. Then I is a model of D iff $I \in \mathbf{T}(\{I\})$.

Proof: I is a model of D
iff for each ground clause $A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n$ from D, $I \not\models B_1 \wedge \ldots \wedge B_n$ if $m = 0$; otherwise, $I \models B_1 \wedge \ldots \wedge B_n$ implies $I \models A_i$ for some A_i ($1 \leq i \leq m$)
iff $I \in T_D(I)$ (by the definition of T_D)
iff $I \in \mathbf{T}(\{I\})$. □

The following theorem provides a fixpoint characterization of the possible model semantics.

Theorem 2.7 Let D be a database. Then, I is in Φ_D iff I is a possible model of D.

Proof: $I \in \Phi_D$
iff $I \in \mathbf{T} \uparrow \omega$ and $I \in \mathbf{T}(\{I\})$
iff I is a model of D (by Lemma 2.6) and for each A in I, there is a ground clause: $A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n$ from D such that $A = A_i$ ($1 \leq i \leq m$) and $I \models B_1 \wedge \ldots \wedge B_n$
iff I is a model of some split database D' of D (by Lemma 2.1) and for each A in I, D' includes a split clause $A \leftarrow B_1 \wedge \ldots \wedge B_n$ and $I \models B_1 \wedge \ldots \wedge B_n$
iff I is the minimal model of D', hence a possible model of D. □

2.3 Negation under the Possible Model Semantics

For inferring negation from a disjunctive database, two alternative rules are known. The one is the *generalized closed world assumption (GCWA)* [Min82], and the other is the *disjunctive database rule (DDR)* [RT88].[3]

[3]The DDR is equivalent to the *weak generalized closed world assumption* of [RLM89]. Hereafter, we use the DDR as a representative.

Definition 2.7 [Min82] Let D be a database. Then, for any ground atom A,

$GCWA(D) \models\sim A$ iff A is false in any minimal model of D. $\square$

Definition 2.8 [RT88] Given a database D, let

$Horn(D) = \{A_i \leftarrow B_1 \wedge \ldots \wedge B_m \mid A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m \in D \text{ and } 1 \leq i \leq l\}$.

Then, for any ground atom A,

$WGCWA(D) \models\sim A$ iff A is false in the least Herbrand model of $Horn(D)$. $\square$

Note here that $Horn(D)$ is always consistent, since it does not contain negative clauses.

Under the possible model semantics, negation is defined as follows.

Definition 2.9 Let D be a database. Then, for any ground atom A,

$GCWA_P(D) \models\sim A$ iff A is false in any possible model of D . $\square$

Proposition 2.8 Let D be a database. If D is consistent, then $D \cup GCWA_P(D)$ is consistent.

Proof: A consistent database D has a possible model M. By the definition of $GCWA_P(D)$, M is also a model of $D \cup GCWA_P(D)$. $\square$

Proposition 2.9 Let D be a database and C be any ground positive clause. Then, $D \vdash C$ iff $D \cup GCWA_P(D) \vdash C$.

Proof: $D \vdash C$ iff C is true in every minimal model of D iff C is true in every possible model of D (by Proposition 2.2). Hence, the result follows. $\square$

The $GCWA_P(D)$ is not decreasing [RT88], that is, $D \subseteq D'$ does not imply $GCWA_P(D) \supseteq GCWA_P(D')$ as the following example shows.

Example 2.5 Let $D = \{\ a \vee b \leftarrow,\ \ a \leftarrow\ \}$ and $D' = \{\ a \vee b \leftarrow,\ \ a \leftarrow,$ $\leftarrow a \wedge b\ \}$. Then $GCWA_P(D) \not\models\sim b$, while $GCWA_P(D') \models\sim b$. $\square$

In definite Horn databases, the GCWA$_P$ reduces to the CWA. For disjunctive databases, negative inference from the GCWA$_P$ is weaker than the GCWA in general.

Proposition 2.10 Let D be a database and A be a ground atom. If $GCWA_P(D) \models\sim A$, then $GCWA(D) \models\sim A$, but not vice versa.

Proof: Since the set of minimal models is included by the set of possible models, the result follows from each definition. $\square$

Example 2.6 Let $D = \{\ a \vee b \leftarrow,\ \ c \leftarrow a \wedge b\ \}$. Then $GCWA(D) \models\sim c$, while $GCWA_P(D) \not\models\sim c$. $\square$

On the other hand, the GCWA$_P$ is stronger than the DDR.

Proposition 2.11 Let D be a database and A be a ground atom. If $DDR(D) \models\sim A$, then $GCWA_P(D) \models\sim A$, but not vice versa. In particular, $DDR(D) = GCWA_P(D)$ if D contains no negative clause.

Proof: Since the least Herbrand model of $Horn(D)$ is a superset of any possible model of D, the result holds. In particular, in the absence of negative clause, the least Herbrand model of $Horn(D)$ coincides with the maximal possible model. Hence, the equivalence relationship holds. □

Example 2.7 (cont. from Example 2.6) Let $D' = D \cup \{ \leftarrow a \wedge b \}$. Then, $DDR(D') \not\models\sim c$, while $GCWA_P(D') \models\sim c$. □

Thus, the GCWA$_P$ can infer negation correctly in both inclusive and exclusive disjunctive databases.

3 Stratified Disjunctive Databases

When a database contains negation, some minimal models which reflect the intended meaning of a database are often selected. In *stratified disjunctive databases*, an intended minimal model, called a *perfect model* is chosen based on the preference relation between predicates [Prz88]. In this section, we extend the possible model semantics to stratified disjunctive databases.

A *database* considered in this section is a finite set of clauses of the form:

$$A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \sim B_{m+1} \wedge \ldots \wedge \sim B_n$$

where $l \geq 0$ and $n \geq m \geq 0$, A_i and B_j are atoms, and all variables are universally quantified at the front of the clause. A database D is *stratified* if there is a partition

$$D = D_1 \cup \ldots \cup D_n$$

such that the following two conditions hold for $i = 1, \ldots, n$.

1. If a predicate occurs positively in a clause in D_i, then its definition (i.e, clauses which contain the predicate symbol in their heads) is contained within $\bigcup_{j \leq i} D_j$.
2. If a predicate occurs negatively in a clause in D_i, then its definition is contained within $\bigcup_{j < i} D_j$.

D_1 can be empty.

Note that [Prz88] gives a definition of a stratified disjunctive database without negative clauses, but our definition possibly includes negative clauses in a database.

Example 3.1 Let $D = \{ a \vee b \leftarrow\sim c, \ \ d \leftarrow, \ \ \leftarrow d \wedge \sim a \}$. Then D is stratified by $D = \{ a \vee b \leftarrow\sim c, \ \ d \leftarrow \} \cup \{ \leftarrow d \wedge \sim a \}$. □

It is easy to see that any split database of a given stratified disjunctive database is also a stratified database. Now we define the possible model semantics for a stratified disjunctive database.

Definition 3.1 Let D be a stratified disjunctive database and M be an interpretation. Then M is a possible model of D if M is a perfect model[4] of some split database D' of D. □

In the previous section, we defined the mapping T_D for databases containing no negation. However, the definition is directly extended to databases containing negation with the condition that $I \models\sim B$ iff $B \notin I$, and the mapping $\mathbf{T}$ is also defined accordingly.

Then we define an *iterative fixpoint semantics* for stratified disjunctive databases as follows.

Definition 3.2 Let D be a database stratified by $D_1 \cup \ldots \cup D_n$. Then the *iterative selective-fixpoint* of D, Ψ_D, is defined as follows.

$$\begin{aligned}
\Psi_{D_1} &= \{\ I_1 \mid I_1 \in \mathbf{T_1} \uparrow \omega(\{\emptyset\}) \ \text{ and } \ I_1 \in \mathbf{T_1}(\{I_1\})\ \}, \\
\Psi_{D_2} &= \{\ I_2 \mid I_2 \in \mathbf{T_2} \uparrow \omega(\Psi_{D_1}) \ \text{ and } I_2 \in \mathbf{T_2}(\{I_2\})\ \}, \\
&\ldots \\
\Psi_{D_n} &= \{\ I_n \mid I_n \in \mathbf{T_n} \uparrow \omega(\Psi_{D_{n-1}}) \ \text{ and } I_n \in \mathbf{T_n}(\{I_n\})\ \}, \\
\Psi_D &= \Psi_{D_n},
\end{aligned}$$

where $\mathbf{T_i}$ is a mapping such that $\mathbf{T_i}(\mathbf{I}) = \bigcup_{I \in \mathbf{I}} T_{D_i}(I)$ and T_{D_i} is the mapping defined for D_i like Definition 2.3. □

The following theorem characterizes the possible model semantics in terms of the iterative fixpoint semantics.

Theorem 3.1 Let D be a stratified disjunctive database. Then I is in Ψ_D iff I is a possible model of D.

Proof: We can get the result by iteratively applying the proof of Theorem 2.7 to each stratum. □

In a stratified disjunctive database D, negation $\sim A$ is inferred by the *iterative closed world assumption (ICWA)* [GPP89] iff A is false in any perfect model of D. The ICWA *for the possible model semantics* ICWA_P is defined in the same way as GCWA_P as follows.

Definition 3.3 Let D be a stratified disjunctive database and A be a ground atom. Then $\sim A$ is inferred from D by the $ICWA_P$ if A is false in any possible model of D. □

The relationship between the ICWA_P and the ICWA is given below.

Proposition 3.2 Let D be a stratified disjunctive database and A be a ground atom. If $ICWA_P(D) \models\sim A$, then $ICWA(D) \models\sim A$, but not vice versa. □

The above result is a generalization of Proposition 2.10.

Example 3.2 Let $D = \{\ a \vee b \leftarrow\sim\ c,\ \ d \leftarrow a \wedge b\ \}$. Then, $ICWA(D) \models\sim d$, while $ICWA_P(D) \not\models\sim d$. □

[4]For the definition of the perfect models, see [Prz88].

4 Proof Procedure

Proof procedures for disjunctive databases have been studied for the GCWA [YH85, HP88], the DDR [RT88] and the WGCWA [RLM89].

In this section, we consider a positive disjunctive database in which *every disjunctive clause is ground.*[5] Then, we present a proof procedure, called an SLD_P-resolution (*SLD-resolution for the possible model semantics*), and show its soundness and completeness with respect to the possible model semantics in such databases. We also give a rule, called the NAF_P (*negation as failure rule for the possible model semantics*) and show its soundness with respect to the $GCWA_P$.

Definition 4.1 Let D be a database and G be a goal $\leftarrow F$, where F is a conjunction of atoms. An *SLD_P-derivation* from D with a goal G consists of a (finite or infinite) sequence of pairs $< G_0, D_0 >=< G, D >, < G_1, D_1 >, \ldots$, such that for any $i \geq 0$, $< G_{i+1}, D_{i+1} >$ is obtained from $< G_i, D_i >$ as follows:

(i) G_i is a goal $\leftarrow L_1 \wedge .. \wedge L_l$ and $L_k (1 \leq k \leq l)$ is called a *selected* atom.

(ii) $C : A_1 \vee .. \vee A_m \leftarrow B_1 \wedge .. \wedge B_n$ is a clause in D_i such that $L_k\theta = A_j\theta$ $(1 \leq j \leq m)$ for some mgu θ.

(iii) G_{i+1} is a goal $\leftarrow (L_1 \wedge .. \wedge L_{k-1} \wedge B_1 \wedge .. \wedge B_n \wedge L_{k+1} \wedge .. \wedge L_l)\theta$.

(iv) D_{i+1} is a database such that C is replaced by a set of split clauses C_p of C, which contains a clause defining A_j (in particular, $D_{i+1} = D_i$ if $m = 1$.). $\square$

Definition 4.2 Let D be a database. An SLD$_P$-refutation from D with a goal G is a finite SLD$_P$-derivation of the pair $< \square, D_n >$ from $< G, D >$, where $\square$ is a null clause. A *successful* SLD$_P$-derivation is a refutation $< \square, D_n >$ where D_n is consistent. A *failed* SLD$_P$-derivation is either (i) one that ends in a pair $< G_n, D_n >$ where G_n is a non-empty goal whose selected atom does not unify with any atom in the head of any clause in D_n, or (ii) one that ends in a pair $< \square, D_n >$ and D_n is inconsistent. $\square$

In the above definition, consistency checking of D_n, which is generated as a pair of the null clause in an SLD$_P$-refutation, is achieved by taking each negative clause in D_n as a goal and examining whether it has an SLD-refutation or not. If any of these goals has an SLD-refutation, D_n is inconsistent. Note here that an SLD-refutation is enough for checking the consistency of D_n. This is because a database D is initially assumed to be consistent, thus inconsistency of D_n is caused by the split clauses in D_n.

An *SLD_P-tree* is defined in a similar way to the definition of an SLD-tree except that each node of the SLD$_P$-tree is a pair $< G_i, D_i >$. The SLD$_P$-tree is *finitely failed* if every branch of the tree is failed.

[5]This condition is needed for technical reasons.

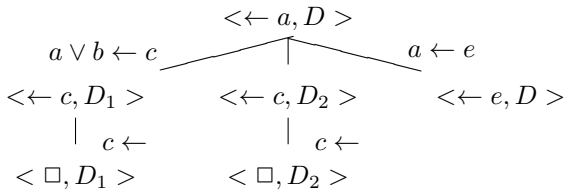

Figure 1: Example 4.1

Example 4.1 Let $D = \{\ a \vee b \leftarrow c, \quad a \leftarrow e, \quad c \leftarrow, \quad \leftarrow a \wedge b\ \}$. An SLD$_P$-tree for the goal $<\leftarrow a, D>$ is given in the Figure 1, where $D_1 = \{\ a \leftarrow c, \ a \leftarrow e, \ c \leftarrow, \ \leftarrow a \wedge b\ \}$ and $D_2 = \{\ a \leftarrow c, \ b \leftarrow c, \ a \leftarrow e, \ c \leftarrow, \ \leftarrow a \wedge b\ \}$. Then the derivation of $<\Box, D_1>$ is a successful derivation, and the derivations of $<\Box, D_2>$ and $<\leftarrow E, D>$ are failed derivations. $\Box$

The process of splitting becomes expensive as the number of disjunctive clauses increases during an SLD$_P$-derivation. For an efficient computation, separating disjunctive clauses from other part of a database is useful. Some search strategies for an SLD$_P$-derivation are also considered. For example, suppose the pairs $<G_i, D_i>$ and $<G_i, D_i'>$ such that $D_i \subseteq D_i'$. If $<G_i, D_i'>$ has no refutation, $<G_i, D_i>$ also has no refutation and we can omit the succeeding derivation from $<G_i, D_i>$. On the other hand, if $G_i = \Box$ and D_i is inconsistent, D_i' is also inconsistent and we know that $<G_i, D_i'>$ is a failed derivation. Further details of such optimisation techniques are not discussed here.

The following theorem presents the soundness and completeness of SLD$_P$-resolution.[6]

Theorem 4.1 Let D be a database and F be a conjunction of atoms. Then $\leftarrow F$ has a successful SLD$_P$-derivation in D with mgu's $\theta_1, \ldots, \theta_n$ iff there exists a possible model of D which satisfies $F\theta_1 \ldots \theta_n$.

Proof: ($\Rightarrow$) Suppose that $\leftarrow F$ has an SLD$_P$-refutation $<\Box, D_n>$. Then D_n is consistent, $F\theta_1..\theta_n$ is a logical consequence of D_n, that is, $F\theta_1..\theta_n$ is true in every minimal model of D_n. Since D_n is subsumed by some split database of D, the result follows from the definition of possible models.

($\Leftarrow$) Suppose that there exists a possible model of D which satisfies $F\theta_1 \ldots \theta_n$. Then there is a split database D_m of D in which $\leftarrow F\theta_1 \ldots \theta_n$ has an SLD-refutation. By the lifting lemma of [Llo87], $\leftarrow F$ has an SLD-refutation with mgu's $\theta_1', \ldots, \theta_n'$ such that $\theta_1, \ldots, \theta_n = \theta_1', \ldots, \theta_n'\gamma$ for some γ and also has an SLD$_P$-refutation $<\Box, D_n>$ for some consistent D_n where D_n is subsumed by D_m. $\Box$

For inferring negation, we give a rule for negation as failure under the possible model semantics.

[6]The theorem considers *credulous* reasoning.

Definition 4.3 Let D be a database and F be a conjunction of ground atoms. Then the *negation as failure rule for possible model semantics* (NAF_P) is that $\sim F$ is inferred if $D \cup \{\leftarrow F\}$ has a finitely failed SLD$_P$-tree. □

The following theorem shows the soundness of the NAF$_P$.

Theorem 4.2 Let D be a database and F be a conjunction of ground atom. If $D \cup \{\leftarrow F\}$ have a finitely failed SLD$_P$-tree, $\sim F$ is inferred from D by the GCWA$_P$.

Proof: Assume that the result does not hold, that is, there exists a possible model of D which satisfies F. Then, by Theorem 4.1, $\leftarrow F$ has a successful SLD$_P$-derivation in D, contradiction. □

To assure the completeness of the NAF$_P$ rule, some mechanism for detecting infinite derivation is needed.

5 Summary

This paper proposed a new semantics for disjunctive databases. The possible model semantics is an alternative approach to the classical minimal model semantics and provides a mechanism of treating both exclusive and inclusive disjunctions uniformly in a database. The possible model semantics was also characterized by the new fixpoint semantics of disjunctive databases. For inferring negation, GCWA$_P$ provides a flexible inference mechanism compared with the GCWA and the DDR. For the procedural part, we have introduced an SLD$_P$-resolution for the possible model semantics. We have also presented the NAF$_P$ rule as a sound procedure for computing the GCWA$_P$.

A disjunctive database is a database containing incomplete information and we consider the possible model semantics can provide natural meaning to such databases. Recently, Chan [Cha89] has independently proposed the *possible world semantics* for disjunctive databases, which turned *equivalent* to our possible model semantics in positive disjunctive databases. The result of this paper is directly extended to *non-stratifiable* general disjunctive databases using the techniques of [GL88, VRS88].

Acknowledgments

Most of this work is done when the author was at ICOT. I would like to thank Kazuhiro Fuchi and Shun-ichi Uchida for providing me with the opportunity to pursue this research. I also wish to thank Yutaka Ohno, director of ASTEM Research Institute of Kyoto, for his encouragement and Hirohisa Seki for helpful comments on an earlier draft of this paper.

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.), Morgan Kaufmann, 89–148, 1988.

[Cha89] Chan, E. P. F., A Possible World Semantics for Disjunctive Databases, Research Report CS-89-47, Dept. of Computer Science, Univ. of Waterloo, 1989.

[GL88] Gelfond, M. and Lifschitz, V., The Stable Model Semantics for Logic Programming, *Proc. 5th Int. Conf.& Symp. on Logic Programming*, 1070–1080, 1988.

[GPP89] Gelfond, M., Przymusinska, H. and Przymusinski, T. C., On the Relationship between Circumscription and Negation as Failure, *Artificial Intelligence* 38, 75–94, 1989.

[HP88] Henschen, L. J. and Park, H., Compiling the GCWA in Indefinite Deductive Databases, in *Foundations of Deductive Databases and Logic Programming* (J. Minker ed.), Morgan Kaufmann, 395–438, 1988.

[Kow79] Kowalski, R. A., *Logic for Problem Solving*, North Holland, New York, 1979.

[Llo87] Lloyd, J. W., *Foundations of Logic Programming*, 2nd edition, Springer-Verlag, Berlin, 1987.

[LMR89] Lobo, J., Minker, J. and Rajasekar, A., Extending the Semantics of Logic Programs to Disjunctive Logic Programs, *Proc. 6th Int. Conf. on Logic Programming*, 255–268, 1989.

[Lov78] Loveland, D. W., *Automated Theorem Proving: A Logical Basis*, North Holland, New York, 1978.

[Mc80] McCarthy, J., Circumscription – A Form of Nonmonotonic Reasoning, *Artificial Intelligence* 13, 27–39, 1980.

[Min82] Minker, J., On Indefinite Data Bases and The Closed World Assumption, *Proc. 6th Int. Conf. on Automated Deduction, Lecture Notes in Computer Science 138*, Springer-Verlag, 292–308, 1982.

[Prz88] Przymusinski, T. C., On the Declarative Semantics of Deductive Databases and Logic Programs, in *Foundations of Deductive Databases and Logic Programming* (J. Minker ed.), Morgan Kaufmann, 193–216, 1988.

[Rei78] Reiter, R., On Closed World Databases, in *Logic and Data Bases* (H. Gallaire and J. Minker eds.), Plenum, New York, 55–76, 1978.

[RLM89] Rajasekar, A., Lobo, J. and Minker, J., Weak Generalized Closed World Assumption, *J. Automated Reasoning* 5, 293–307, 1989.

[RT88] Ross, K. A. and Topor, R. W., Inferring Negative Information from Disjunctive Databases, *J. Automated Reasoning* 4, 397–424, 1988.

[VK76] van Emden, M. H. and Kowalski, R. A., The Semantics of Predicate Logic as a Programming Language, *J.ACM* 23, 733–742, 1976.

[VRS88] Van Gelder, A., Ross, K. and Schlipf, J. S., Unfounded Sets and Well-Founded Semantics for General Logic Programs, *Proc. 7th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, 221–230, 1988.

[YH85] Yahya, A. and Henschen, L. J., Deduction in Non-Horn Databases, *J. Automated Reasoning* 1, 141–160, 1985.