Journal of Automated Reasoning **13:** 145–172, 1994.

An Alternative Approach to the Semantics of Disjunctive Logic Programs and Deductive Databases*

CHIAKI SAKAMA
Advanced Software Technology and Mechatronics Research Institute of Kyoto, 17 Chudoji Minami-machi, Shimogyo, Kyoto 600 Japan. e-mail: sakama@astem.or.jp

and

KATSUMI INOUE
Department of Information and Computer Sciences, Toyohashi University of Technology, Tempaku-cho, Toyohashi 441 Japan. e-mail: inoue@tutics.tut.ac.jp

(Received August 2, 1993; revised March 28, 1994)

Abstract. In this paper, we study a new semantics of logic programming and deductive databases. The *possible model semantics* is introduced as a declarative semantics of disjunctive logic programs. The possible model semantics is an alternative theoretical framework to the classical minimal model semantics and provides a flexible inference mechanism for inferring negation in disjunctive logic programs. We also present a proof procedure for the possible model semantics and show that the possible model semantics has an advantage from the computational complexity point of view.

Key words: disjunctive logic programs, possible model semantics, closed world assumption, model generation procedure

1. Introduction

Traditionally, the declarative semantics of logic programming and deductive databases has been studied based on the notion of *minimal models.* For instance, the *least Herbrand model semantics* for Horn logic programs [40], the *perfect model semantics* for stratified logic programs [28], and the *stable model semantics* for normal logic programs [14] are all minimal models. The minimal models reflect the so-called Occam's razor such that "only those objects should be assumed to exist which are minimally required by the context." Such a *principle of minimality* plays a fundamental role in the area of not only logic programming but also *nonmonotonic reasoning* in artificial intelligence [23]. Therefore, it has been recognized that the principle of minimality is one of the most basic and indispensable criteria that each semantics for commonsense reasoning should obey [39].

This is also the case in the context of *disjunctive logic programs,* logic programs containing indefinite information. Namely, the *minimal model semantics* for positive disjunctive programs [24] and the *disjunctive stable model semantics* for normal disjunctive programs [29] are both minimal. However, such a minimalism is not always appropriate in a theory containing indefinite information. Ross and Topor [34] first noticed this problem in the context of inferring negation in disjunctive logic programs. They argue

* This is a revised and extended version of the paper [36] which was presented at the Tenth International Conference on Logic Programming, Budapest, 21–25 June 1993.

that when one infers negation from a disjunctive logic program, one should be cautious to interpret disjunctions *inclusively* rather than *exclusively*. In fact, the minimal model semantics minimizes truth extensions of predicates as much as possible; then it usually interprets disjunctions exclusively and maximizes negative information inferred from a program.

In logic programming and deductive databases, Reiter has introduced the *closed world assumption* (CWA) [31] as a default rule for inferring negation from a program. However, Reiter has also pointed out that the CWA works well only for Horn logic programs and causes an inconsistency in the presence of indefinite information in a program. In positive disjunctive programs, Minker [24] has extended Reiter's CWA to the *generalized closed world assumption* (GCWA). On the other hand, Ross and Topor [34] have proposed an alternative rule called the *disjunctive database rule* (DDR), which turns out to be equivalent to the *weak generalized closed world assumption* (WGCWA) that was independently proposed by Rajasekar *et al.* [32].

If one compares these two rules, one sees that the GCWA is based on the minimal model semantics and usually interprets disjunctions exclusively, while the DDR and the WGCWA are weaker than the GCWA and interpret disjunctions inclusively. Thus, both the GCWA and the WGCWA (or DDR) fairly extend the CWA, but the problem is that they are inherently *different* from each other. In fact, in the absence of a single uniform framework, one has to use these separate rules in order to treat both exclusive and inclusive disjunctions in the same program. Such a situation actually happens in our real-life situations. For instance, consider the program

$$land\text{-}animal \vee aquatic \leftarrow animal,$$

$$biped \vee quadruped \leftarrow land\text{-}animal,$$

$$amphibian \leftarrow land\text{-}animal \wedge aquatic,$$

in which the disjunction in the first clause is inclusive, while the disjunction in the second clause is exclusive. As another example, a disjunctive clause possibly contains *hybrid* disjunctions such as:

$$Sunday \vee national\text{-}holiday \vee weekday \leftarrow calendar\text{-}days,$$

in which *Sunday* ∨ *weekday* is exclusive, while *Sunday* ∨ *national-holiday* is inclusive.

To treat such kinds of programs, we need a single framework that can distinguish two kinds of disjunctions. Another point is that Ross and Topor and Rajasekar *et al.* have provided a rule for inferring negation in inclusive disjunctive programs; however, they concern only negative information in a program and do not provide any model theoretical meaning for inclusive disjunctive programs as a counterpart of the minimal model semantics.

In this paper, we present an alternative approach to the declarative semantics of disjunctive logic programs and deductive databases. We introduce a new semantics called the *possible model semantics* for disjunctive logic programs. In contrast to the classical minimal model semantics, the possible model semantics considers not only minimal models but also certain kinds of nonminimal models in a program. We show that the possible model semantics enjoys several interesting properties. Especially by treating both inclusive and exclusive disjunctions uniformly in a program, it can provide a flexible mechanism for inferring negation in a program. Next we develop an algorithm to compute the possible model semantics in normal disjunctive programs which is based on a bottom-up model generation proof procedure. We also discuss the computational complexity of the possible model semantics and show that the possible model semantics has a computational advantage compared with other minimal model based semantics.

The rest of this paper is organized as follows. In Section 2, we introduce the possible model semantics for positive disjunctive programs and present its properties. The possible model semantics is also extended to normal disjunctive programs in Section 3. In Section 4, we provide a proof procedure to compute the possible model semantics in normal disjunctive programs. In Section 5, we discuss the computational aspect of the possible model semantics. Section 6 presents detailed comparisons with related work, and Section 7 concludes this paper. Some proofs are contained in Appendix.

2. Possible Model Semantics for Positive Disjunctive Programs

In this section, we first consider positive disjunctive programs, that is, disjunctive programs containing no negation-as-failure formulas.

2.1. Positive disjunctive programs

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

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

where A_i's and B_j's are atoms. The left-hand side of the clause is called the *head*, while the right-hand side of the clause is called the *body*. A clause is called *disjunctive* (resp. *definite*) if its head contains more than one atom (resp. exactly one atom). A clause with the empty head and a nonempty body is called an *integrity constraint*. A program containing no disjunctive clauses is called a *Horn logic program*, and especially a Horn logic program containing no integrity constraint is called a *definite logic program*. A program is semantically identified with its *ground program*, which is the possibly infinite set of all ground clauses from the program.

An *interpretation* of a program P is a subset of the Herbrand base $\mathcal{HB}_P$ of the program. An interpretation I *satisfies* the clause (1) if $\{B_1, \ldots, B_m\} \subseteq I$ implies $A_i \in I$ for some i $(1 \leqslant i \leqslant l)$. In particular, I satisfies the integrity constraint (1) with $l = 0$ if $B_j \notin I$ for some j $(1 \leqslant j \leqslant m)$. An interpretation satisfying every clause in a program is a *model* of the program. A model I is called *supported* if for each atom A in I, there is a clause (1) such that $A = A_i$ $(1 \leqslant i \leqslant l)$ and $\{B_1, \ldots, B_m\} \subseteq I$.* A model I is a *minimal model* if there is no model smaller than I. If a program has a unique minimal model, it is called the *least Herbrand model*. A positive disjunctive program is *consistent* if it has a minimal model; otherwise it is *inconsistent*.

* This is a direct extension of the notion introduced in [1] for normal logic programs.

2.2. Negation in positive disjunctive programs

In logic programming, Reiter's CWA [31] is formally stated as follows: Given a program P,

$$CWA(P) = \{\neg A \mid A \in \mathcal{HB}_P \text{ and } P \not\models A\}.$$

However, Reiter's CWA causes an inconsistency in the presence of disjunctive information in a program; for instance, the program $P = \{a \vee b \leftarrow\}$ is inconsistent with $CWA(P) = \{\neg a, \neg b\}$. Then, for inferring negation in positive disjunctive programs, Reiter's CWA is mainly extended in two ways: one is Minker's *generalized closed world assumption* (GCWA) and the other is Ross and Topor's *disjunctive database rule* (DDR) or Rajasekar et al.'s *weak generalized closed world assumption* (WGCWA)*. We first review definitions and properties of those two frameworks.

Given a positive disjunctive program P, let us denote by $\mathcal{MM}_P$ the set of all minimal models of P. Then the GCWA is defined as follows.

DEFINITION 2.1. [24]. Let P be a consistent positive disjunctive program. Then $GCWA(P)$ is defined as

$$GCWA(P) = \{\neg A \mid A \in \mathcal{HB}_P \text{ and } A \notin I \text{ for any } I \in \mathcal{MM}_P\}.$$

On the other hand, the WGCWA provides a weaker form of closed world reasoning in positive disjunctive programs as follows.

The *Horn transformation* [34] of a positive disjunctive program P is defined as

$$Horn(P) = \{A_i \leftarrow B_1 \wedge \cdots \wedge B_m \mid A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \in P \text{ and } 1 \leqslant i \leqslant l,\ l \geqslant 1\}.$$

Note here that $Horn(P)$ is always consistent, since it does not contain integrity constraints.

DEFINITION 2.2. [34, 32]. Let P be a consistent positive disjunctive program and $Horn(P)$ be its Horn transformation. Let $M_{Horn(P)}$ be the least Herbrand model of $Horn(P)$. Then $WGCWA(P)$ is defined as

$$WGCWA(P) = \{\neg A \mid A \in \mathcal{HB}_P \text{ and } A \notin M_{Horn(P)}\}.$$

Properties of the GCWA and the WGCWA are as follows.

THEOREM 2.1. [24, 34, 32]. *Let P be a consistent positive disjunctive program and A be a ground atom. Then,*

(i) *$P \cup GCWA(P)$ is consistent.*

$P \cup WGCWA(P)$ is consistent.

* According to [32, 21], the DDR and the WGCWA are equivalent. Hence, we use the term WGCWA hereafter.

(ii) $P \models A$ *iff* $P \cup GCWA(P) \models A$.

$P \models A$ *iff* $P \cup WGCWA(P) \models A$.

(iii) $P \subseteq P'$ *does not imply* $GCWA(P') \subseteq GCWA(P)$.

$P \subseteq P'$ *implies* $WGCWA(P') \subseteq WGCWA(P)$.

(iv) $WGCWA(P) \subseteq GCWA(P)$.

(v) *For a definite logic program* P, $GCWA(P) = WGCWA(P) = CWA(P)$.

That is, (i) both $GCWA(P)$ and $WGCWA(P)$ are *consistent* with P, (ii) positive facts proven from P are *invariant*, (iii) the GCWA (resp. WGCWA) is *nondecreasing* (resp. *decreasing*), (iv) the GCWA is *stronger* than the WGCWA, and (v) for definite logic programs each rule *reduces* to the CWA.

EXAMPLE 2.1. Let P be the program

$$\{a \vee b \vee c \leftarrow,\ d \leftarrow a \wedge b,\ e \leftarrow b \wedge c,\ \leftarrow b \wedge c\},$$

where $\mathcal{MM}_P = \{\{a\},\{b\},\{c\}\}$ and $M_{Horn(P)} = \{a, b, c, d, e\}$. Then $GCWA(P) = \{\neg d, \neg e\}$, while $WGCWA(P) = \emptyset$.

In the above example, the GCWA interprets each disjunction *exclusively*, while the WGCWA interprets them *inclusively*. Then, $GCWA(P)$ excludes the inclusive interpretation of $a \vee b$ and infers $\neg d$ from the program. On the other hand, the integrity constraint $\leftarrow b \wedge c$ inhibits an inclusive interpretation of $b \vee c$; nevertheless, $WGCWA(P)$ cannot infer $\neg e$. This is because the WGCWA does not consider the model theoretical meaning of a given program, and ignores the effect of integrity constraints in a program. In fact, $M_{Horn(P)}$ is no longer a model of P. Generally speaking, the GCWA is too strong to interpret inclusive disjunctions, while the WGCWA is too weak to treat exclusive disjunctions. Then, to treat both types of disjunctions in a program, one has to use two different rules in the same program.

To improve such a situation, Sakama [35] has proposed a new semantics called the *possible model semantics*, which can distinguish both types of disjunctions uniformly in a program. In the next subsection, we present the possible model semantics and its properties.

2.3. Possible Model Semantics

A disjunctive program is considered to represent a set of possible facts that might have been true in the actual world. The possible model semantics is intended to formulate this situation.

Given a positive disjunctive program P, a *split program* is defined as a ground Horn logic program obtained from P by replacing each ground disjunctive clause of the form (1):

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

with the following ground definite clauses (called *split clauses*):

$$A_i \leftarrow B_1 \wedge \cdots \wedge B_m \quad \text{for every } A_i \in S,$$

where S is some nonempty subset of $\{A_1, \ldots, A_l\}$. Note that every ground Horn clause from P is included in any split program of P. Then, a *possible model* of P is defined as the least Herbrand model of any split program of P. The set of all possible models of P is denoted by $\mathcal{PM}_P$.

EXAMPLE 2.2. Let P be the program

$$\{a \vee b \leftarrow,\ b \vee c \leftarrow,\ \leftarrow b \wedge c\}.$$

Then the split programs of P are

$$\{a \leftarrow,\ b \leftarrow,\ \leftarrow b \wedge c\},$$

$$\{a \leftarrow,\ c \leftarrow,\ \leftarrow b \wedge c\},$$

$$\{b \leftarrow,\ \leftarrow b \wedge c\},$$

$$\{b \leftarrow,\ c \leftarrow,\ \leftarrow b \wedge c\},$$

$$\{a \leftarrow,\ b \leftarrow,\ c \leftarrow,\ \leftarrow b \wedge c\}.$$

Since the last two split programs are inconsistent, the set of all possible models of P is $\mathcal{PM}_P = \{\{a, b\}, \{a, c\}, \{b\}\}$.

Possible models have the following properties.

PROPOSITION 2.2. *A consistent positive disjunctive program has at least one possible model.*

Proof. Since a consistent positive disjunctive program has a consistent split program, the result immediately follows. □

PROPOSITION 2.3. *A possible model of a positive disjunctive program P is a model of P.*

Proof. Let M be a possible model of P such that M is the least Herbrand model of a consistent split program P_s. Then, for each clause C': $A_i \leftarrow B_1 \wedge \cdots \wedge B_m$ in P_s, there is a corresponding clause C: $A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m$ in P, where $1 \leqslant i \leqslant l$. Since M satisfies each C', it also satisfies C. Also each integrity constraint in P is included in P_s and is satisfied in M. Hence M is a model of P. □

The notion of possible models is different from minimal models. In fact, in Example 2.2, $\{a, b\}$ is a possible model, but not a minimal model. The intuitive meaning of the possible model is that each atom included in a possible model has its possible justification in a program. Thus, both inclusive and exclusive interpretations of disjunctions are considered whenever there is no integrity constraint to inhibit inclusive interpretations. The following property directly follows from the definition.

PROPOSITION 2.4. *A possible model is a supported model.*

The converse of the above proposition does not hold in general. For example, $\{a\}$ is a supported model of the program $\{a \vee b \leftarrow a\}$, but not a possible model. The following relationship holds between possible models and minimal models of a program.

PROPOSITION 2.5. *Let P be a consistent positive disjunctive program. Then the set of all minimal elements from $\mathcal{PM}_P$ coincides with $\mathcal{MM}_P$.*

Proof. Let M be a minimal model of P. Then, for each clause $A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m$ in P, $\{B_1, \ldots, B_m\} \subseteq M$ implies $A_i \in M$ for some i $(1 \leqslant i \leqslant l)$. In this case, there is a split program P_s of P that contains $A_i \leftarrow B_1 \wedge \cdots \wedge B_m$. Since M satisfies each integrity constraint in P, M is the least Herbrand model of the consistent Horn logic program P_s, hence a possible model of P. Thus, $\mathcal{MM}_P$ is included in $\mathcal{PM}_P$. On the other hand, since each possible model is a model of P by Proposition 2.3, the set of all minimal elements from $\mathcal{PM}_P$ coincides with $\mathcal{MM}_P$. □

Thus the set of minimal models are included by the set of possible models. In particular, a definite logic program has a unique possible model which is the least Herbrand model of the program. The above proposition implies that as for the inference of positive facts, the possible model semantics coincides with the minimal model semantics.

THEOREM 2.6. *Let P be a consistent positive disjunctive program. An atom A is true in P iff A is included in every possible model of P.*

Note that possible models depend on the syntax of a program. For instance, two programs $\{a \vee b \leftarrow, a \leftarrow\}$ and $\{a \leftarrow\}$ are equivalent under first-order logic, while the first program has the possible model $\{a, b\}$ which is not a possible model of the second program. This is because the first program is intended to specify some indefinite information about b, while it is not the case in the second program. Such a distinction is in fact meaningful in knowledge representation. Suppose a situation like that:

There is a visitor at my house whom I do not know. I am living with my parents so that I guess either my mother or farther must know him: $know(mother, visitor) \vee know(father, visitor)$. After a while, mother comes back and she actually knows him: $know(mother, visitor)$, and at this moment the possibility is open my father's knowing him too. However, if we replace the previous belief $know(mother, visitor) \vee know(father, visitor)$ with $know(mother, visitor)$, the negation $\neg know(father, visitor)$ is assumed under the closed world assumption.

Thus, logically equivalent sentences do not necessarily have the same meaning from the viewpoint of knowledge representation. Generally speaking, the syntax of a program plays an important role in logic programming and deductive databases to specify our intended knowledge, then it appears natural that the possible model semantics also shares such syntax-dependent properties.

Next we consider the inference of negative facts under the possible model semantics. Under the possible model semantics, negation is defined as follows.

DEFINITION 2.3. Let P be a consistent positive disjunctive program. Then the *possible world assumption* (PWA) of P is defined as:

$$PWA(P) = \{\neg A \mid A \in \mathcal{HB}_P \text{ and } A \notin I \text{ for any } I \in \mathcal{PM}_P\}. \qquad \square$$

THEOREM 2.7. *Let P be a consistent positive disjunctive program and A be a ground atom. Then,*

(i) $P \cup PWA(P)$ *is consistent.*
(ii) $P \models A$ *iff* $P \cup PWA(P) \models A$.
(iii) $P \subseteq P'$ *does not imply* $PWA(P') \subseteq PWA(P)$.
(iv) *For a definite logic program* P, $PWA(P) = CWA(P)$.

Proof. (i) Since P is consistent, it has a minimal model M which is also a possible model of P. Then, by definition, M is also a model of $P \cup PWA(P)$; hence the result follows. (ii) The result directly follows from Theorem 2.6. (iii) Let $P = \{a \vee b \leftarrow, c \leftarrow a \wedge b\}$ and $P' = P \cup \{\leftarrow a \wedge b\}$. Then $PWA(P) = \emptyset$, while $PWA(P') = \{\neg c\}$. (iv) For a definite logic program P, $\mathcal{PM}_P$ contains the unique least Herbrand model of P; hence the result follows. □

The next theorem presents that the PWA is stronger than the WGCWA and weaker than the GCWA.

THEOREM 2.8. *Let P be a consistent positive disjunctive program. Then the following relationship holds:*

$$WGCWA(P) \subseteq PWA(P) \subseteq GCWA(P).$$

In particular, $WGCWA(P) = PWA(P)$ *if* $P \cup Horn(P)$ *is consistent.*

Proof. The relationship $PWA(P) \subseteq GCWA(P)$ immediately follows from the fact that $\mathcal{MM}_P \subseteq \mathcal{PM}_P$. Since the least Herbrand model $M_{Horn(P)}$ of $Horn(P)$ is a superset of any possible model in $\mathcal{PM}_P$, the relationship $WGCWA(P) \subseteq PWA(P)$ also holds. In particular, $Horn(P)$ is the set of all definite clauses included in the maximal split program of P. Then, when $P \cup Horn(P)$ is consistent, $M_{Horn(P)}$ coincides with the maximal element in $\mathcal{PM}_P$; hence the result follows. □

EXAMPLE 2.3. (cont. from Example 2.1). Let P be the program

$$\{a \vee b \vee c \leftarrow,\ d \leftarrow a \wedge b,\ e \leftarrow b \wedge c,\ \leftarrow b \wedge c\},$$

where $\mathcal{PM}_P = \{\{a\}, \{b\}, \{c\}, \{a, b, d\}, \{a, c\}\}$. Then, $PWA(P)$ implies $\neg e$, but not $\neg d$.

Note that in the above example $P \cup Horn(P)$ is inconsistent and $WGCWA(P)$ fails to capture the intended meaning of P. By contrast, the possible model semantics can infer proper negation by distinguishing two kinds of disjunctions using integrity constraints.

The possible model semantics was independently discovered by Chan [4] under the name of the *possible world semantics*. In [4], both notions are proven to be equivalent in positive disjunctive programs.

3. Possible Model Semantics for Normal Disjunctive Programs

In this section, we extend the possible model semantics of positive disjunctive programs to normal disjunctive programs.

3.1. Normal disjunctive programs

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

$$A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \wedge not B_{m+1} \wedge \cdots \wedge not B_n \qquad (2)$$

$$(1 \geqslant 0,\ n \geqslant m \geqslant 0),$$

where A_i's and B_j's are atoms and *not* is the *negation-as-failure* operator [5]. A clause is called *disjunctive* (resp. *normal*) if its head contains more than one atom (resp. exactly one atom). A clause with the empty head and a nonempty body is called an *integrity constraint*. A program containing no disjunctive clause is called a *normal logic program*. A normal disjunctive program reduces to a positive disjunctive program when every clause contains no *not*. Every normal disjunctive program is also identified with its ground program.

An interpretation I *satisfies* the clause (2) if $\{B_1, \ldots, B_m\} \subseteq I$ and $\{B_{m+1}, \ldots, B_n\} \cap I = \emptyset$ implies $A_i \in I$ for some i $(1 \leqslant i \leqslant l)$. In particular, I satisfies the integrity constraint (2) with $l = 0$ if $B_j \notin I$ for some j $(1 \leqslant j \leqslant m)$ or $B_k \in I$ for some k $(m + 1 \leqslant k \leqslant n)$. An interpretation satisfying every clause in a program is a *model* of the program. A model I is called *supported* if for each atom A in I, there is a clause (2) such that $A = A_i$ $(1 \leqslant i \leqslant l)$, $\{B_1, \ldots, B_m\} \subseteq I$ and $\{B_{m+1}, \ldots, B_n\} \cap I = \emptyset$. A model I of a program P is called a *stable model* if it coincides with a minimal model of the positive disjunctive program P^I defined as

$$P^I = \{A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \mid \text{there is a ground clause of the form (2) from } P \text{ such that } \{B_{m+1}, \ldots, B_n\} \cap I = \emptyset\}.$$

The *disjunctive stable model semantics* [29] of a normal disjunctive program P is defined as the set of all stable models of P (denoted by $\mathcal{ST}_P$). In particular, the disjunctive stable model semantics coincides with the stable model semantics of [14] in normal logic programs, and the minimal model semantics in positive disjunctive programs. Stable models are minimal and supported models, but not vice versa [29, 25].

A normal disjunctive program is *consistent* if it has a model; otherwise it is *inconsistent*. A normal disjunctive program having at least one stable model is called *coherent*; otherwise it is called *incoherent*. Note that a consistent program is not always coherent. For instance, the program $\{a \leftarrow not\ a\}$ has a model $\{a\}$ that is not stable.

3.2. Possible model semantics and negation

We first introduce possible models in normal disjunctive programs. The notion of split programs is defined in the same manner as positive disjunctive programs.

Given a normal disjunctive program P, its *split program* is defined as a ground normal logic program obtained from P by replacing each ground disjunctive clause of the form (2):

$$A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \wedge not B_{m+1} \wedge \cdots \wedge not B_n$$

with the following ground normal clauses (called *split clauses*):

$$A_i \leftarrow B_1 \wedge \cdots \wedge B_m \wedge not B_{m+1} \wedge \cdots \wedge B_n \text{ for every } A_i \in S,$$

where S is some nonempty subset of $\{A_1, \ldots, A_l\}$. Then, a *possible model* of P is defined as a stable model of any split program of P. The set of all possible models of P is denoted by $\mathcal{PM}_P$. A normal disjunctive program having at least one possible model is called *p-coherent*, otherwise it is called *p-incoherent*.

The possible models defined above reduce to those presented in the previous section in a positive disjunctive program. Also, possible models coincide with stable models in normal logic programs. The following properties hold.

PROPOSITION 3.1. *A possible model of a normal disjunctive program P is a model of P.*

PROPOSITION 3.2. *A possible model is a supported model, but not vice versa.*

PROPOSITION 3.3. *Let P be a consistent normal disjunctive program. Then the set of all minimal elements from $\mathcal{PM}_P$ contains $\mathcal{ST}_P$.*

Proof. By definition, a stable model M of P is also a stable model of some split program of P. Then M is also a possible model of P. Since M is minimal, it is also a minimal element in $\mathcal{PM}_P$. □

The above proposition implies that a coherent normal disjunctive program is also p-coherent (but not vice versa; see Example 3.6). The converse of the above proposition does not hold in general. That is, minimal possible models are not always stable models.

EXAMPLE 3.1. Let P be the program

$$\{a \vee b \leftarrow,\ b \leftarrow a,\ c \leftarrow not\, a\}.$$

Then $\mathcal{PM}_P = \{\{a, b\}, \{b, c\}\}$ and $\{a, b\}$ is a minimal element in $\mathcal{PM}_P$ but not a stable model of P.

In the above example, $\{b, c\}$ is the unique stable model of P; hence c is true under the disjunctive stable model semantics. However, this is not the case under the possible model semantics, since there is an inclusive interpretation of the disjunction $\{a, b\}$ in which c is not true. Thus, in contrast to the case of positive disjunctive programs, positive facts true under the possible model semantics (viz. positive facts true in every possible model) differ from those ones under the disjunctive stable model semantics. At the end of this section, we will also show the case that the possible model semantics implies more positive facts than the disjunctive stable model semantics.

Now we consider negative inference in normal disjunctive programs. We first define an extension of the GCWA as negation under the disjunctive stable model semantics.

DEFINITION 3.1. Let P be a coherent normal disjunctive program. Then $GCWA^{\neg}(P)$ is defined as

$$GCWA^{\neg}(P) = \{\neg A \mid A \in \mathcal{HB}_P \text{ and } A \notin I \text{ for any } I \in \mathcal{ST}_P\}.$$

Next, to define a suitable extension of the WGCWA, we introduce a transformation from a normal disjunctive program to a normal logic program.

The *NLP-transformation* of a normal disjunctive program P is defined as

$$\begin{aligned} NLP(P) = \{ & A_i \leftarrow B_1 \wedge \cdots \wedge B_m \wedge not\, B_{m+1} \wedge \cdots \wedge not\, B_n \mid A_1 \vee \cdots \\ & \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \wedge not\, B_{m+1} \wedge \cdots \wedge not\, B_n \in P \text{ and} \\ & 1 \leqslant i \leqslant l,\ l \geqslant 1\}. \end{aligned}$$

The $NLP(P)$ is a direct extension of $Horn(P)$, and $NLP(P) = Horn(P)$ holds for a positive disjunctive program P. For a coherent normal disjunctive program P, $NLP(P)$ is not always coherent.

EXAMPLE 3.2. Let P be the program

$$\{a \vee b \leftarrow not\, a\}.$$

Then its NLP-transformation becomes

$$NLP(P) = \{a \leftarrow not\, a,\ b \leftarrow not\, a\}.$$

Here $\mathcal{ST}_P = \{\{b\}\}$, while $\mathcal{ST}_{NLP(P)} = \emptyset$.

Conversely, there is an incoherent program whose NLP-transformation has a stable model.

EXAMPLE 3.3. Let P be the program

$$\{a \vee b \leftarrow,\ b \leftarrow a,\ \leftarrow not\, a\}.$$

Then $\mathcal{ST}_P = \emptyset$, while $\mathcal{ST}_{NLP(P)} = \{\{a, b\}\}$.

We say that a normal disjunctive program P is *weakly coherent* if either P or $NLP(P)$ has a stable model. Clearly, a coherent program is also weakly coherent, but not vice versa.

DEFINITION 3.2. Let P be a weakly coherent normal disjunctive program. Then $WGCWA^{\neg}(P)$ is defined as

$$WGCWA^{\neg}(P) = \{\neg A \mid A \in \mathcal{HB}_P \text{ and } A \notin I \text{ for any } I \in \mathcal{ST}_P \cup \mathcal{ST}_{NLP(P)}\}.$$

This definition is natural in the sense that $WGCWA^{\neg}(P)$ restricts its negative inference like the WGCWA by taking into account the stable models of $NLP(P)$. A similar extension is also given in [8] in a different context. Note that in the above definition, one may consider that by analogy with the WGCWA, considering $\mathcal{ST}_{NLP(P)}$ is enough instead of $\mathcal{ST}_P \cup \mathcal{ST}_{NLP(P)}$. But this is not the case. For instance, let $P = \{a \vee b \leftarrow,$ $c \leftarrow not\, a,\ c \leftarrow not\, b\}$. Then $\mathcal{ST}_P = \{\{a, c\}, \{b, c\}\}$; hence $GCWA^{\neg}(P)$ does not imply $\neg c$. On the other hand, $NLP(P) = \{a \leftarrow,\ b \leftarrow,\ c \leftarrow not\, a,\ c \leftarrow not\, b\}$; then $\mathcal{ST}_{NLP(P)} = \{\{a, b\}\}$, which implies $\neg c$. Thus, without $\mathcal{ST}_P$, the $WGCWA^{\neg}$ is not weaker than the $GCWA^{\neg}$ anymore.

The PWA is also extended in normal disjunctive programs as follows.

DEFINITION 3.3. Let P be a p-coherent normal disjunctive program. Then $PWA^{\neg}(P)$ is defined as

$$PWA^{\neg}(P) = \{\neg A \mid A \in \mathcal{HB}_P \text{ and } A \notin I \text{ for any } I \in \mathcal{PM}_P\}.$$

Now we investigate the properties of each rule. In the following, we write $P \models_{ST} A$ (resp. $P \models_{PM} A$) if $A \in I$ for any $I \in \mathcal{ST}_P$ (resp. $I \in \mathcal{PM}_P$).

THEOREM 3.4. *Let P be a normal disjunctive program and A be a ground atom. Then the following properties hold.*

1. (i) *If P is coherent, $P \cup GCWA^{\neg}(P)$ is coherent.*
   (ii) *$P \models_{ST} A$ iff $P \cup GCWA^{\neg}(P) \models_{ST} A$.*
   (iii) *$P \subseteq P'$ does not imply $GCWA^{\neg}(P') \subseteq GCWA^{\neg}(P)$.*
   (iv) *For a positive disjunctive program P, $GCWA^{\neg}(P) = GCWA(P)$.*
2. (i) *If P is weakly coherent, $P \cup WGCWA^{\neg}(P)$ is weakly coherent.*
   (ii) *$P \models_{ST} A$ iff $P \cup WGCWA^{\neg}(P) \models_{ST} A$.*
   (iii) *$P \subseteq P'$ does not imply $WGCWA^{\neg}(P') \subseteq WGCWA^{\neg}(P)$.*
   (iv) *For a positive disjunctive program P, $WGCWA^{\neg}(P) = WGCWA(P)$.*
3. (i) *If P is p-coherent, $P \cup PWA^{\neg}(P)$ is p-coherent.*
   (ii) *$P \models_{PM} A$ iff $P \cup PWA^{\neg}(P) \models_{PM} A$.*
   (iii) *$P \subseteq P'$ does not imply $PWA^{\neg}(P') \subseteq PWA^{\neg}(P)$.*
   (iv) *For a positive disjunctive program P, $PWA^{\neg}(P) = PWA(P)$.*

Proof. 1. (i) Since P is coherent, it has at least one stable model and every negated atom in $GCWA^{\neg}(P)$ is not in any stable model of P; hence $P \cup GCWA^{\neg}(P)$ is coherent. (ii) Any negated fact included in $GCWA^{\neg}(P)$ is not included in any stable model. Then adding such negative facts to P does not affect the construction of stable models. Hence the result follows. (iii) The $GCWA^{\neg}$ is nondecreasing since the $GCWA^{\neg}$ includes the GCWA (by (iv)) which is nondecreasing. (iv) Since stable models reduce to minimal models in a positive disjunctive program, the result immediately follows.

2. (i) When P is weakly coherent, every negated atom in $WGCWA^{\neg}(P)$ is not included in any stable model of P and $NLP(P)$. Hence $P \cup WGCWA^{\neg}(P)$ is also weakly coherent. (ii) The result also follows from (i). (iii) For nondecreasing property of $WGCWA^{\neg}(P)$, see Example 3.4. (iv) Since $\mathcal{ST}_P \cup \mathcal{ST}_{NLP(P)}$ reduces to $\mathcal{MM}_P \cup \{M_{Horn(P)}\}$ in a positive disjunctive program P, and each minimal model in $\mathcal{MM}_P$ is a subset of $M_{Horn(P)}$, the result also holds.

3. (i) and (ii) follow directly from the definition. (iii) The $PWA^{\neg}$ is nondecreasing since the $PWA^{\neg}$ includes the PWA (by (iv)), which is nondecreasing. (iv) Since possible models in a normal disjunctive program reduce to those presented in the preceding section in a positive disjunctive program, the result immediately follows. □

Notice that in contrast to the WGCWA, the WGCWA$^{\neg}$ is nondecreasing.

EXAMPLE 3.4. Let P be the program

$$\{a \vee b \leftarrow not\, c,\ c \leftarrow d\}$$

and $P' = P \cup \{d \leftarrow\}$. Then, $\mathcal{ST}_P \cup \mathcal{ST}_{NLP(P)} = \{\{a\},\{b\},\{a,b\}\}$ and $\mathcal{ST}_{P'} \cup \mathcal{ST}_{NLP(P')} = \{\{c,d\}\}$. Hence, $WGCWA^{\neg}(P) = \{\neg c, \neg d\}$, while $WGCWA^{\neg}(P') = \{\neg a, \neg b\}$.

Thus, in the presence of negation as failure in a program, the monotonic decreasing property of the WGCWA does not hold anymore.

For coherent normal logic programs, the three rules coincide.

PROPOSITION 3.5. *Let P be a coherent normal logic program. Then, $GCWA^{\neg}(P) = WGCWA^{\neg}(P) = PWA^{\neg}(P)$.*

Proof. For a coherent normal logic program P, $\mathcal{ST}_P \cup \mathcal{ST}_{NLP(P)} = \mathcal{ST}_P$. Then the relation $GCWA^{\neg}(P) = WGCWA^{\neg}(P)$ holds by each definition. The relation $GCWA^{\neg}(P) = PWA^{\neg}(P)$ also holds since $\mathcal{ST}_P = \mathcal{PM}_P$ holds for a coherent normal logic program P. □

The next theorem presents the relationship among three rules in normal disjunctive programs.

THEOREM 3.6. *Let P be a coherent normal disjunctive program. Then,*

(i) $WGCWA^{\neg}(P) \subseteq GCWA^{\neg}(P)$.
(ii) $PWA^{\neg}(P) \subseteq GCWA^{\neg}(P)$.

Proof. Since $\mathcal{ST}_P \subseteq \mathcal{ST}_P \cup \mathcal{ST}_{NLP(P)}$, (i) follows from definitions. The part (ii) also follows from the fact that $\mathcal{ST}_P \subseteq \mathcal{PM}_P$. □

As for the WGCWA$^{\neg}$ and the PWA$^{\neg}$, there is no inclusion relationship.

EXAMPLE 3.5. Consider the program

$$P = \{a \vee b \vee c \leftarrow not\, d,\ e \leftarrow a \wedge b \wedge not\, c\}.$$

Then $\mathcal{ST}_P = \{\{a\},\{b\},\{c\}\}$, $\mathcal{ST}_{NLP(P)} = \{\{a,b,c\}\}$ and $WGCWA^{\neg}(P) = \{\neg d, \neg e\}$, while $\mathcal{PM}_P = \{\{a\},\{b\},\{c\},\{a,b,e\},\{b,c\},\{c,a\},\{a,b,c\}\}$ and $PWA^{\neg}(P) = \{\neg d\}$. Hence, $WGCWA^{\neg}(P) \not\subseteq PWA^{\neg}(P)$. The converse inclusion relation does not hold by Theorem 2.8 either, since each rule reduces to the WGCWA or the PWA in positive disjunctive programs.

In the above example, $WGCWA^{\neg}(P)$ treats the disjunction $a \vee b \vee c$ inclusively; then it infers $\neg e$. This is also the case for $GCWA^{\neg}(P)$, which treats it exclusively. On the other hand, there is the possible model $\{a,b,e\}$ in which a and b are inclusively true and c is exclusively false at the same time; then $\neg e$ is not inferred by $PWA^{\neg}(P)$. This example illustrates that the possible model semantics also properly treats both types of disjunctions in normal disjunctive programs and provides the most careful negative inference compared with the other two.

Moreover, the PWA$^{\neg}$ can often infer proper negation even in an incoherent program.

EXAMPLE 3.6. Let P be the program

$$\{a \vee b \leftarrow,\ b \leftarrow a,\ \leftarrow not\, a,\ c \leftarrow not\, b\}.$$

Then $\mathcal{ST}_P = \emptyset$, $\mathcal{ST}_{NLP(P)} = \{\{a, b\}\}$ and $\mathcal{PM}_P = \{\{a, b\}\}$; hence $GCWA^{\neg}(P)$ is not well defined, while $PWA^{\neg}(P)$ and $WGCWA^{\neg}(P)$ imply $\neg c$.

The above program is incoherent, but p-coherent, since $\{a, b\}$ is a possible model of P which is not a stable model. Observing the above program, the third clause asserts that a should be true, which possibly holds by the first disjunctive clause. Also the truth of a implies the truth of b in the second clause, then it seems natural to assert the falsity of c by the last clause.

As shown in the above example, the disjunctive stable model semantics often fails to capture the intended meaning of a program because of its minimal feature. By contrast, the possible model semantics is well defined whenever a stable model exists, and is often useful than the disjunctive stable model semantics thanks to its nonminimal nature.

4. Computing Possible Models

4.1. Bottom-up Model Generation Procedure

The algorithm we use to compute possible models in disjunctive programs is based on a bottom-up model generation proof procedure. We assume here and in the subsequent subsections that a program is function-free and range-restricted, that is, any variable in a clause has an occurrence in a positive atom in the body. Such conditions are usually imposed on a program in the context of deductive databases.

The following algorithm computes a set of interpretations of a positive disjunctive program from a given set of interpretations. Let P be a positive disjunctive program and $\mathcal{I}_P^i$ be a set of interpretations of P. Let $\mathcal{I}_P^0 = \{\emptyset\}$. For $i \geqslant 0$, do the following:

1. For any $I \in \mathcal{I}_P^i$, for every clause C_k in P of the form:

   $$C_k\colon\ A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \quad (l \geqslant 1)$$

   such that $I \models (B_1 \wedge \cdots \wedge B_m)\,\sigma$ for some ground substitution σ, put $I \cup \bigcup_{C_k} \{A_j \sigma\}$ into $\mathcal{I}_P^{i+1}$ for every $j = 1, \ldots, l$.

2. For any $I \in \mathcal{I}_P^{i+1}$, if there is an integrity constraint in P of the form:

   $$\leftarrow B_1 \wedge \cdots \wedge B_m$$

   such that $I \models (B_1 \wedge \cdots \wedge B_m)\,\sigma$ for some ground substitution σ, then remove I from $\mathcal{I}_P^{i+1}$.

3. Iterate the above two steps until it reaches the fixpoint $\mathcal{I}_P^{n+1} = \mathcal{I}_P^n$, which is closed under the above two operations.

In Step 1, the above procedure generates the new set of interpretations $\mathcal{I}_P^{i+1}$ from the given interpretations $\mathcal{I}_P^i$ by performing forward reasoning based on hyperresolution and case-splitting on nonunit derived clauses. In Step 2, interpretations which do not satisfy integrity constraints in the program are pruned. Note here that since a program is range-restricted, each disjunct $A_j\sigma$ generated in Step 1 is always ground. Hence, the soundness for unsatisfiability by case-splitting is guaranteed [22]. Moreover, since we consider a finite function-free program, the above procedure always terminates in a finite step.

EXAMPLE 4.1. Let P be the program

$$\{a \vee b \leftarrow c,\ d \leftarrow c,\ c \leftarrow,\ e \leftarrow b,\ \leftarrow b \wedge e\}.$$

Then,

$$\begin{aligned}
\mathcal{I}_P^0 &= \{\emptyset\},\\
\mathcal{I}_P^1 &= \{\{c\}\},\\
\mathcal{I}_P^2 &= \{\{c,d,a\},\{c,d,b\}\},\\
\mathcal{I}_P^3 &= \{\{c,d,a\},\{c,d,a,b\}\},\\
\mathcal{I}_P^4 &= \{\{c,d,a\},\{c,d,a,b\}\},
\end{aligned}$$

where $\mathcal{I}_P^4$ $(= \mathcal{I}_P^3)$ is the fixpoint.

In the above example, $\{c, d, a\}$ and $\{c, d, a, b\}$ in $\mathcal{I}_P^3$ and $\mathcal{I}_P^4$ are generated from $\{c, d, a\}$ by the first clause. On the other hand, $\{c, d, b, e\}$ is generated from $\{c, d, b\}$ in $\mathcal{I}_P^3$ but is pruned by the integrity constraint $\leftarrow b \wedge e$.

Now we characterize the possible model semantics using the algorithm presented above. Let $\mathcal{I}_P^\omega$ be the fixpoint closure obtained by the above procedure and $\mathcal{T}$ be the function such that $\mathcal{I}_P^{n+1} = \mathcal{T}(\mathcal{I}_P^n)$.

LEMMA 4.1. *Let P be a positive disjunctive program. Then I is a model of P iff $I \in \mathcal{T}(\{I\})$.*

Proof. I is a model of P iff for each disjunctive clause $A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m$ in P, $I \models (B_1 \wedge \cdots \wedge B_m)\sigma$ implies $I \models A_i\sigma$ for some A_i $(1 \leqslant i \leqslant l)$ and a ground substitution σ, and for any integrity constraint $\leftarrow B_1 \wedge \cdots \wedge B_m$ in P, $I \not\models (B_1 \wedge \cdots \wedge B_m)\sigma$ iff $I \in \mathcal{T}(\{I\})$. □

Now let $\mu(\mathcal{I}_P^\omega) = \{I \mid I \in \mathcal{I}_P^\omega \text{ and } I \in \mathcal{T}(\{I\})\}$.

THEOREM 4.2. *Let P be a positive disjunctive program. Then,*

$$\mathcal{PM}_P = \mu(\mathcal{I}_P^\omega).$$

Proof. I is in $\mu(\mathcal{I}_P^\omega)$

iff I is included in $\mathcal{I}_P^\omega$ and is a model of P (by Lemma 4.1) iff each A_i in I is obtained from a ground disjunctive clause $A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m$ $(1 \leqslant i \leqslant l)$ from P by hyperresolution and case-splitting, and I satisfies every integrity constraint in P

iff I is the least Herbrand model of a consistent split Horn logic program P_s of P and each A_i in I is derived by the split clause $A_i \leftarrow B_1 \wedge \cdots \wedge B_m$ in P_s

iff I is in $\mathcal{PM}_P$. □

COROLLARY 4.3. *A positive disjunctive program P is inconsistent iff $\mu(\mathcal{I}_P^\omega) = \emptyset$.*

EXAMPLE 4.2 (cont. from Example 4.1). $\mathcal{I}_P^\omega = \{\{c, d, a\}, \{c, d, a, b\}\}$ and $\mu(\mathcal{I}_P^\omega) = \{\{c, d, a\}\}$. Thus, $\mathcal{PM}_P = \{\{c, d, a\}\}$.

By Proposition 2.5, the following result directly follows.

COROLLARY 4.4. *Let P be a positive disjunctive program. Then,*

$$\mathcal{MM}_P = \min\left(\mu(\mathcal{I}_P^\omega)\right),$$

where $\min(\mu(\mathcal{I}_P^\omega)) = \{I \in \mu(\mathcal{I}_P^\omega) \mid \nexists J \in \mu(\mathcal{I}_P^\omega)$ *such that* $J \subset I\}$.

For each negative inference rule, the following results hold.

THEOREM 4.5. *For a consistent positive disjunctive program P and a ground atom A,*

(i) $GCWA(P) \models \neg A$ *iff* $A \notin I$ *for any* $I \in \min(\mu(\mathcal{I}_P^\omega))$.
(ii) $WGCWA(P) \models \neg A$ *iff* $A \notin I$ *for* $I \in \mathcal{I}_{Horn(P)}^\omega$.
(iii) $PWA(P) \models \neg A$ *iff* $A \notin I$ *for any* $I \in \mu(\mathcal{I}_P^\omega)$.

Proof. (i) and (iii) directly follow from each definition and Theorem 4.2 and Corollary 4.4. Since $\mathcal{I}_{Horn(P)}^\omega$ contains a unique element that is the least Herbrand model of $Horn(P)$, (ii) also follows from the definition of the WGCWA. □

4.2. Program Transformation

In this subsection, we present a method of computing possible models in normal disjunctive programs. To this end, we first introduce a program transformation that translates a normal disjunctive program into a semantically equivalent positive disjunctive program.*

DEFINITION 4.1. Let P be a normal disjunctive program. Then its *epistemic transformation* is defined as the positive disjunctive program P^κ obtained from P by transforming each clause of the form (2) in P containing *not*:

$$A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \wedge not\, B_{m+1} \wedge \cdots \wedge not\, B_n \quad (m \neq n)$$

* A similar transformation is also presented in [17, 18] in different contexts.

into the following *not*-free clauses in P^{κ}:

$$\lambda_1 \vee \cdots \vee \lambda_l \vee \mathrm{K}B_{m+1} \vee \cdots \vee \mathrm{K}B_n \leftarrow B_1 \wedge \cdots \wedge B_m, \tag{3}$$

$$A_i \leftarrow \lambda_i \quad \text{for } i = 1, \ldots, l, \tag{4}$$

$$\leftarrow \lambda_i \wedge B_j \quad \text{for } i = 1, \ldots, l \text{ and } j = m+1, \ldots, n, \tag{5}$$

$$\lambda_i \leftarrow A_i \wedge \lambda_k \quad \text{for } i = 1, \ldots, l \text{ and } k = 1, \ldots, l. \tag{6}$$

In particular, each integrity constraint containing *not* is transformed into

$$\mathrm{K}B_{m+1} \vee \cdots \vee \mathrm{K}B_n \leftarrow B_1 \wedge \cdots \wedge B_m.$$

Note here that each *not*-free clause in P is included in P^{κ} as it is.

In the epistemic transformation, the newly introduced atom $\mathrm{K}B_j$ means that B_j *is believed.* With this epistemic reading, each negation-as-failure formula *not* B_j in the body of a clause is rewritten in $\neg\mathrm{K}B_j$ and shifted to the head of the clause. In the transformed clause, λ_i is a newly introduced atom not appearing elsewhere in P and is uniquely associated with each ground instance of a clause (2) from P.*

An intuitive reading of the transformed clauses is that if $B_1, \ldots, B_m$ are true, then some A_i $(1 \leqslant i \leqslant l)$ becomes true via λ_i when $B_{m+1}, \ldots, B_n$ are not true; otherwise, some B_j $(m+1 \leqslant j \leqslant n)$ is believed. The clause (6) has an effect to associate λ_i with A_i whenever A_i is true and another disjunct A_k is derived from (3) via λ_k.** In this way, every normal disjunctive program P is transformed into a positive disjunctive program P^{κ}. Then its model generation procedure has already been defined.

Let I^{κ} be an interpretation of P^{κ}. Then I^{κ} is called *canonical* if $\mathrm{K}A \in I^{\kappa}$ implies $A \in I^{\kappa}$ for any atom A in $\mathcal{HB}_P$. That is, in a canonical interpretation each believed atom has a justification. Given a set of interpretations $\mathcal{I}_{P^{\kappa}}$, let

$$obj_c(\mathcal{I}_{P^{\kappa}}) = \{I^{\kappa} \cap \mathcal{HB}_P \mid I^{\kappa} \in \mathcal{I}_{P^{\kappa}} \text{ and } I^{\kappa} \text{ is canonical}\}.$$

For the disjunctive stable model semantics, the following results hold.

THEOREM 4.6. *Let P be a normal disjunctive program and P^{κ} be its epistemic transformation. Then,*

$$\mathcal{ST}_P = obj_c\big(\min\big(\mu(\mathcal{I}^{\omega}_{P^{\kappa}})\big)\big).$$

In particular, P is incoherent iff $obj_c(\min(\mu(\mathcal{I}^{\omega}_{P^{\kappa}}))) = \emptyset$.

Proof. See Appendix. □

COROLLARY 4.7. *Let P be a normal logic program. Then,*

$$\mathcal{ST}_P = obj_c\big(\mu(\mathcal{I}^{\omega}_{P^{\kappa}})\big).$$

* If a clause contains n distinct free variables $\mathbf{x} = x_1, \ldots, x_n$, then a new atom $\lambda_i(\mathbf{x})$ can be associated with each A_i, where λ_i is an n-ary predicate symbol appearing nowhere in P.

** In case of $l = 1$, the clause (6) becomes a tautological clause $\lambda \leftarrow A \wedge \lambda$, and hereafter we will omit such a clause in P^{κ}.

Proof. See Appendix. □

The next theorem states that the possible model semantics of a normal disjunctive program can be computed by the fixpoint closure of the transformed program.

THEOREM 4.8. *Let P be a normal disjunctive program. Then,*

$$\mathcal{PM}_P = obj_c(\mu(\mathcal{I}^{\omega}_{P^{\kappa}})).$$

In particular, P is p-incoherent iff $obj_c(\mu(\mathcal{I}^{\omega}_{P^{\kappa}})) = \emptyset$.

Proof. Let I be a possible model of P. Then I is a stable model of some coherent split normal logic program P_s of P. By Corollary 4.7, for the transformed program P_s^{κ} of P_s, I is in $obj_c(\mu(\mathcal{I}^{\omega}_{P_s^{\kappa}}))$. Since P_s is a program obtained by splitting each disjunctive clause in P, every interpretation included in $\mathcal{I}^{\omega}_{P_s^{\kappa}}$ is also obtained by case-splitting during the computation of $\mathcal{I}^{\omega}_{P^{\kappa}}$. Moreover, since I satisfies each integrity constraint in P_s^{κ}, it also satisfies the same integrity constraints in P^{κ}. Hence I is also in $obj_c(\mu(\mathcal{I}^{\omega}_{P_s^{\kappa}}))$. The converse is also shown in the same manner. □

For each negative inference rule, the following results hold.

THEOREM 4.9. *Let P be a normal disjunctive program and A be a ground atom.*

(i) *For a coherent program P, $GCWA^{\neg}(P) \models \neg A$ iff $A \notin I$ for any $I \in obj_c(\min(\mu(\mathcal{I}^{\omega}_{P^{\kappa}})))$.*

(ii) *For a weakly coherent program P, $WGCWA^{\neg}(P) \models \neg A$ iff $A \notin I$ for any $I \in obj_c(\min(\mu(\mathcal{I}^{\omega}_{P^{\kappa}}))) \cup obj_c(\mu(\mathcal{I}^{\omega}_{NLP(P)^{\kappa}}))$.*

(iii) *For a p-coherent program P, $PWA^{\neg}(P) \models \neg A$ iff $A \notin I$ for any $I \in obj_c(\mu(\mathcal{I}^{\omega}_{P^{\kappa}}))$.*

Proof. (i) and (iii) directly follow from Theorems 4.6 and 4.8. (ii) also follows from Corollary 4.7 and the definition of the WGCWA$^{\neg}$. □

EXAMPLE 4.3. (cont. from Example 3.1) The program

$$P = \{a \vee b \leftarrow,\ b \leftarrow a,\ c \leftarrow not\ a\}$$

is transformed into the epistemic form

$$P^{\kappa} = \{a \vee b \leftarrow,\ b \leftarrow a,\ \lambda \vee \mathrm{K}a \leftarrow,\ c \leftarrow \lambda,\ \leftarrow \lambda \wedge a\}.$$

Then it becomes

$$\mu(\mathcal{I}^{\omega}_{P^{\kappa}}) = \{\{a, b, \mathrm{K}a\}, \{b, \mathrm{K}a\}, \{b, c, \lambda\}, \{b, c, \lambda, \mathrm{K}a\}\}.$$

Thus,

$$obj_c(\mu(\mathcal{I}^{\omega}_{P^{\kappa}})) = \{\{a, b\}, \{b, c\}\},$$

which contains the possible models of P. On the other hand,

$$\min(\mu(\mathcal{I}^{\omega}_{P^{\kappa}})) = \{\{b, \mathrm{K}a\}, \{b, c, \lambda\}\};$$

hence,

$$obj_c(\min(\mu(\mathcal{I}^{\omega}_{P^{\kappa}}))) = \{\{b, c\}\},$$

which coincides with the stable model of P.

As noticed in the preceding section, stable models are not necessarily minimal possible models. However, the above example shows that we can compute stable models exactly by computing the minimal set of the closure $\mu(\mathcal{I}^{\omega}_{P^{\kappa}})$ before applying the operation $obj_c(\cdot)$.

4.3. Query Answering

In this subsection, we address an application of the previously presented algorithm to query answering under the possible model semantics in normal disjunctive programs.

A *query* we consider here is the following form:

$$Q(\mathbf{x}) \leftarrow B_1 \wedge \cdots \wedge B_m \wedge not\, B_{m+1} \wedge \cdots \wedge not\, B_n, \tag{7}$$

where (7) is a function-free range-restricted normal clause and $\mathbf{x}$ represents variables appearing in the body of the clause. An *answer* to the query is a ground substitution σ for variables in $Q(\mathbf{x})$. In particular, if Q contains no variable, σ is the empty substitution.

For a given normal disjunctive program P, let P_Q be a program obtained from P by adding a query of the form (7). Then, the query is *true* in P under the possible model semantics if for every possible model I of P_Q there is an answer σ such that $Q(\mathbf{x})\sigma$ is included in I. Else if for some possible model I of P_Q there is an answer σ such that $Q(\mathbf{x})\sigma$ is included in I, a query is *possibly true*. Otherwise, if there is no such answer, a query is *false*. By Theorem 4.8, the following results hold.

THEOREM 4.10. *Let P be a normal disjunctive program and Q be a query. Then,*

(i) *Q is true in P iff for any $I \in obj_c(\mu(\mathcal{I}^{\omega}_{P^{\kappa}_Q}))$, $Q(\mathbf{x})\sigma \in I$ for some σ.*

(ii) *Q is possibly true in P iff for some $I \in obj_c(\mu(\mathcal{I}^{\omega}_{P^{\kappa}_Q}))$, $Q(\mathbf{x})\sigma \in I$ for some σ.*

(iii) *Q is false in P iff for any $I \in obj_c(\mu(\mathcal{I}^{\omega}_{P^{\kappa}_Q}))$, $Q(\mathbf{x})\sigma \notin I$ for any σ.*

EXAMPLE 4.4. Let P be the program

$$\{p(a) \vee p(b) \leftarrow\}.$$

Then, $q_1(X) \leftarrow p(X)$ is true, $q_2 \leftarrow p(a)$ is possibly true, and $q_3 \leftarrow p(c)$ is false.

By using Theorem 4.6 instead, the above result is also applicable to query answering under the disjunctive stable model semantics.

5. Computational Complexity

In a propositional positive disjunctive program, a minimal model exists whenever the program is satisfiable. Then the complexity of deciding the existence of a minimal model is NP-complete. Also, an atom is a logical consequence of a propositional positive disjunctive program whenever its negation is unsatisfiable in the program, so deciding whether a given atom is included in every minimal model is coNP-complete. Since the possible model semantics coincides with the minimal model semantics for positive inference, those complexity results also hold for the possible model semantics in propositional positive disjunctive programs. On the other hand, it is known that the complexity of deciding whether a given atom is included in some minimal model is Σ_2^P-complete, and thus inferring negation under the GCWA is Π_2^P-complete [10]. By contrast, Chan [4] has shown that, in a propositional positive disjunctive program, inferring negation under the WGCWA or the PWA is still coNP-complete. In particular, in the absence of integrity constraints both the WGCWA and the PWA are tractable.

These observations tell us that the possible model semantics has the computational advantage over the minimal model semantics for inferring negation, since it does not increase the complexity more than the classical propositional entailment. Moreover, as shown in Section 2, since the PWA is more intuitive than the WGCWA, it is concluded that the possible model semantics is the best choice among others from both the reasoning and computational points of view.

In this section, we prove that the complexity results for the possible model semantics is still within (co)NP, even in normal disjunctive programs. We show this fact by transforming possible models in a normal disjunctive program into stable models in a normal logic program.

DEFINITION 5.1. Let P be a normal disjunctive program. Then the *pm-transformation* transforms P into the normal logic program $\wp(P)$ which is obtained from P by replacing each disjunctive clause

$$A_1 \vee \cdots \vee A_l \leftarrow \Gamma \tag{8}$$

in P with the following normal clauses and an integrity constraint:

$$A_i \leftarrow \Gamma \wedge not\, A_i' \quad \text{for} \quad i = 1, \ldots, l, \tag{9}$$

$$A_i' \leftarrow \Gamma \wedge not\, A_i \quad \text{for} \quad i = 1, \ldots, l, \tag{10}$$

$$\leftarrow \Gamma \wedge A_1' \wedge \cdots \wedge A_l', \tag{11}$$

in $\wp(P)$, where Γ denotes a conjunction in the body of the clause and each A_i' is a new atom not appearing in P and is uniquely introduced for each A_i in $\mathcal{HB}_P$.

The intuitive meaning of the *pm*-transformation is that when A_i becomes true by the disjunctive clause (8), we can make it true also by the corresponding normal clause (9) in $\wp(P)$ by assuming that its complementary atom A_i' is not true. Else when A_i does not become true by (8), we will make A_i' true by assuming that A_i is not true in the corresponding normal clause (10) in $\wp(P)$. The condition (11) states that when Γ is true, every A_i' cannot become true at the same time, that is, at least one A_i should be true. Thus, the transformed clauses represent every possible selection of disjuncts from the disjunctive head of the clause, which exactly characterizes every set of split clauses of (8).

Now we show that there is a one-to-one correspondence between the possible models of P and the stable models of $\wp(P)$. We first present a preliminary lemma.

LEMMA 5.1. *Let P be a normal disjunctive program. Then M is a possible model of P iff M is a possible model of P^M.*

Proof. M is a possible model of P

iff M is a stable model of some split normal logic program P_s of P
iff M is the least Herbrand model of P_s^M

iff M is the least Herbrand model of some split Horn logic program P_s^M of P^M
iff M is a possible model of P^M. □

THEOREM 5.2. *Let P be a normal disjunctive program and $\wp(P)$ be its pm-transformation. Then $\mathcal{PM}_P = \mathcal{ST}_{\wp(P)} \cap \mathcal{HB}_P$ holds, where $\mathcal{ST}_{\wp(P)} \cap \mathcal{HB}_P = \{I \cap \mathcal{HB}_P \mid I \in \mathcal{ST}_{\wp(P)}\}$.*

Proof. (i) First we show that $\mathcal{PM}_P = \mathcal{ST}_{\wp(P)} \cap \mathcal{HB}_P$ holds for a positive disjunctive program P. Let M be a possible model of a positive disjunctive program P. Then M is the least Herbrand model of a split program P_s of P. In this case, M is also the least Herbrand model of a program in which each disjunctive clause (8) in P is replaced with its split clauses $\{A_i \leftarrow \Gamma \mid A_i \in M \cap \{A_1, \ldots, A_l\}\}$. Now let us consider a Horn logic program P_s' which is obtained from P by replacing each disjunctive clause (8) with clauses of $\{A_i \leftarrow \Gamma \mid A_i \in M \cap \{A_1, \ldots, A_l\}\} \cup \{A_j' \leftarrow \Gamma \mid A_j \in \{A_1, \ldots, A_l\} \setminus M\}$. Let M' be the least Herbrand model of P_s'. Then clearly $M = M' \cap \mathcal{HB}_P$ holds. Here such a program P_s' together with the integrity constraint (11) coincides with the program $\wp(P)^{M'}$. Since $M \models \Gamma$ implies at least one $A_i \in M$, M' satisfies the condition (11). Then M' is also the least Herbrand model of $\wp(P)^{M'}$, hence a stable model of $\wp(P)$.

Conversely, let M be a stable model of $\wp(P)$. Since M satisfies the condition (11), $M \models \Gamma$ implies that at least one of the clauses (9) becomes $A_i \leftarrow \Gamma$ in $\wp(P)^M$, and for each such clause $M \models \Gamma$ implies $A_i \in M$. In this case, there is a split program P_s of P in which each disjunctive clause (8) is replaced with its split clauses $\{A_i \leftarrow \Gamma \mid A_i \in M \cap \{A_1, \ldots, A_l\}\}$. Since M is the least Herbrand model of $\wp(P)^M$, $M \cap \mathcal{HB}_P$ is also the least Herbrand model of P_s, hence a possible model of P.

(ii) Next we show that $\mathcal{PM}_P = \mathcal{ST}_{\wp(P)} \cap \mathcal{HB}_P$ holds for a normal disjunctive program P. Let M be a possible model of a normal disjunctive program P. By Lemma 5.1, M is also a possible model of a positive disjunctive program P^M. Then, by (i), there is a stable model M' of $\wp(P^M)$ such that $M = M' \cap \mathcal{HB}_P$, which is also the least Herbrand model of $\wp(P^M)^{M'}$. Since $\wp(P^M)^{M'} = \wp(P^{M'})^{M'} = \wp(P)^{M'}$, M' is also a stable model of $\wp(P)$.

Conversely, let M be a stable model of $\wp(P)$. Then M is the least Herbrand model of $\wp(P)^M$. Since $\wp(P)^M = \wp(P^M)^M$, M is also the least Herbrand model of $\wp(P^M)^M$, and a stable model of $\wp(P^M)$. Then, by (i), $M \cap \mathcal{HB}_P$ is a possible model of P^M. Since $P^M = P^{M \cap \mathcal{HB}_P}$, $M \cap \mathcal{HB}_P$ is a possible model of P by Lemma 5.1. □

The above theorem presents that the possible models of any normal disjunctive program are expressed by the stable models of the corresponding transformed normal logic program.

For the stable model semantics in propositional normal logic programs, it is known that the problem of existence of a stable model and the problem of membership of an atom in some stable model are both NP-complete, while the problem of membership of an atom in every stable model is coNP-complete [26, 27].* We use this fact to show the computational complexity of the possible model semantics.

* In [26, 27], integrity constraints are not included in a program. However, a program containing integrity constraints is easily reducible to the one without them by rewriting each integrity constraint $\leftarrow G$ by the normal clause $false \leftarrow G$.

TABLE I. Complexity results for disjunctive programs

Program	Semantics	Complexity
Positive DLP	minimal model (GCWA)	Π_2^P-complete
	WGCWA	coNP-complete
	possible model (PWA)	coNP-complete
Normal DLP	disjunctive stable model (GCWA$^\neg$)	Π_2^P-complete
	WGCWA$^\neg$	Π_2^P-complete
	possible model (PWA$^\neg$)	coNP-complete

THEOREM 5.3. *Let P be a propositional normal disjunctive program. Then,*

(i) *Deciding the existence of a possible model of P is NP-complete.*
(ii) *Deciding whether an atom is true in some possible model of P is NP-complete.*
(iii) *Deciding whether an atom is true in every possible model of P is coNP-complete.*

Proof. Since possible models coincide with stable models in normal logic programs, each decision problem under the possible models semantics is (co)NP-hard. To see that it is in (co)NP, note that the *pm*-transformation efficiently translates each decision problem for possible models into the corresponding problem for stable models, which is in (co)NP; then the membership in (co)NP follows. □

COROLLARY 5.4. *Inferring negation under the* PWA$^\neg$ *is coNP-complete.*

It is known that the decision problems for the disjunctive stable model semantics is Σ_2^P-complete for the existence problem and the membership problem for some stable model, and Π_2^P-complete for the membership problem for every stable model [10]. Then the following result also follows from the definition.

COROLLARY 5.5. *Inferring negation under the $GCWA^\neg$ or the $WGCWA^\neg$ is both Π_2^P-complete.*

The complexity results for disjunctive programs are summarized in Table I.

These results show that the frameworks based on the minimal/disjunctive stable model semantics introduce an additional source of complexity for minimality-checking, while this is not the case for computation of possible models due to its nonminimal feature.

We have already seen in the preceding sections that the possible model semantics can provide flexible reasoning mechanisms compared with the minimal/disjunctive stable model semantics thanks to its nonminimal nature. The results of this section present that this unique feature of the possible model semantics also brings a computational advantage over those minimal model-based semantics.

6. Related Work

6.1. Declarative Semantics

The minimal model semantics of positive disjunctive programs was first introduced by Minker [24] and extended by Przymusinski [28] to the perfect model semantics for (locally) stratified disjunctive programs. Further extensions to normal disjunctive programs have been done in the context of the stable model semantics [29] and the well-founded semantics [33, 30, 2]. As nonminimal model approaches, Ross and Topor [34], and Rajasekar *et al.* [32] have proposed the DDR and the WGCWA as a counterpart of the GCWA. However, they present only negative inference in inclusive disjunctive programs and do not provide any model theoretical meaning for such programs. Ross and Topor also suggest in their paper the usage of integrity constraints to distinguish exclusive disjunctions from inclusive ones, but they give no semantics for such programs. To characterize the meaning of inclusive disjunctive programs, Dix [8] presents the *weak* perfect/stationary model semantics for normal disjunctive programs without integrity constraints. However, these weak semantics have some drawbacks compared with the possible model semantics. First, the weak semantics cannot represent exclusive disjunctive programs. Second, the weak semantics do not work well in the presence of integrity constraints. For example, in Example 2.1 the weak semantics of the program is given by $\mathcal{MM}_P \cup \{M_{Horn(P)}\} = \{\{a\},\{b\},\{c\},\{a,b,c,d,e\}\}$, but as noted there, $\{a,b,c,d,e\}$ is not a model of P. Then, if we choose models satisfying the constraint and give the meaning of P by $\{\{a\},\{b\},\{c\}\}$, it cannot represent the inclusive disjunction $a \vee b$ anymore.

To treat both exclusive and inclusive disjunctions, Ross [33] has proposed the *optimal* well-founded semantics, which can distinguish two types of disjunctions in normal disjunctive programs. However, his semantics requires each rule to be *clarified* whether it is exclusive or inclusive, and it cannot treat a disjunctive clause containing hybrid disjunctions as presented in the introductory example. Dung [9] has also presented a *completion* theory of negation, which can distinguish two types of disjunctions in a program. However, it is defined for only positive disjunctive programs and also cannot treat hybrid disjunctions in a program. Przymusinski [30] suggests that his *stationary semantics* can also treat two types of disjunctions by altering the GCWA and the WGCWA during the construction of completions of disjunctions, while it seems impossible to treat disjunctive clauses containing hybrid disjunctions. Gelfond [16] has developed an epistemic theory for disjunctive programs and provided a flexible mechanism for inferring closed world negation. However, his approach is based on modal logic and is different from our object-level approach.

Recently, Eiter *et al.* [11] have introduced a circumscriptive approach for inclusive disjunctions in a first-order theory. Their *good models* provide a model theoretical counterpart of inclusive interpretation of disjunctions. However, in contrast to our approach, their *Curb* theory is defined for a first-order theory and is classical in its nature. For instance, $\{a,b\}$ is a possible model of the program $\{a \vee b \leftarrow, a \leftarrow\}$ as presented in Section 2, while they identify the above program with $\{a \leftarrow\}$ and $\{a\}$ is the unique good model. In this sense, their approach is syntax-independent and different from our syntax-dependent logic programming approach. Moreover, their Curb theory is defined for a first-order theory and its application to logic programming is limited to positive disjunctive programs.

The possible model semantics was also independently discovered by Chan [4] under the name of the possible world semantics. It was also discovered by Decker [6] under the name of the *sustained model semantics*. Decker and Casamayor [7] have also shown that their sustained world assumption, which corresponds to the PWA, satisfies the properties such as *cautious monotonicity, cumulativity* and *rationality* in the sense of [20]. These works have characterized the possible model semantics from different viewpoints, while they consider only positive disjunctive programs; extensions to normal disjunctive programs are not studied in the literature.

To distinguish two types of disjunctions, one may consider that instead of inserting integrity constraints, inserting *cyclic* clauses under the usual minimal model semantics is enough to interpret inclusive disjunctions. But this is not the case. Consider interpreting the disjunction $a \vee b$ inclusively by adding cyclic clauses $a \leftarrow b$ and $b \leftarrow a$ to it. The resultant program now implies the equivalence $a \Longleftrightarrow b$. Applying it to the introductory example, we obtain *land-animal* $\Longleftrightarrow$ *aquatic*, which is of course not our intention.

We have used integrity constraints to distinguish exclusive disjunctions from inclusive ones. Then if one wishes to simulate the GCWA under the PWA, it is enough to insert integrity constraints for each exclusive disjunction. Such a simulation is discussed in [4].

6.2. Proof Procedure

Fernandez *et al.* [13] develop a model generation procedure for computing minimal and stable models of dicjunctive logic programs using a similar program transformation to ours. Compared with their approach, our algorithm is designed for computing not only minimal/stable models but also possible models and is easily realizable in a nondeterministic or-parallel environment like [17]. The model generation procedure presented in this paper might be considered as a variant of SATCHMO [22] or MGTP [17], but these procedures are designed to compute minimal/stable models and are different from ours. For computing possible models, Chan [4] presents a different procedure that, given a positive disjunctive program P and its model M, finds a subset of M that is also a possible model of P.

We have also presented a method of using a bottom-up procedure to evaluate queries under the possible model semantics. However, this method is somewhat *naive* in the sense of [3] and might need some compilation technique in the presence of huge databases. As an alternative approach, in the preceding section we have presented that possible models of a normal disjunctive program can be expressed in terms of stable models of a normal logic program by using the *pm*-transformation. This means that, using the *pm*-transformation, a top-down proof procedure for the stable model semantics of normal logic programs can also be used as a procedure for the possible model semantics of normal disjunctive programs. For instance, Eshghi and Kowalski's *abductive proof procedure* [12] is known to be correct with respect to *call-consistent* normal logic programs.* Since the *pm*-transformation preserves the call-consistency, the abductive procedure is also used as a proof procedure for the possible model semantics. For positive disjunctive programs, yet other top-down procedures are developed in [35, 6, 7].

7. Conclusion

In this paper, we have introduced the possible model semantics for positive and normal disjunctive programs, which is an alternative nonminimal model approach to the

* Informally speaking, a normal logic program is call-consistent if it contains no self-recursive predicate through an odd number of negation. For a more precise definition, see [38] for instance.

declarative semantics of logic programming and deductive databases. The possible model semantics gives a uniform framework to treat both inclusive and exclusive disjunctions in a program, and provides a flexible negative inference mechanism compared with the previously proposed closed world assumptions.

For computing possible models, we have presented a bottom-up model generation proof procedure for positive and normal disjunctive programs. The procedure is sound and complete with respect to the possible model semantics as well as the minimal/stable model semantics in function-free range-restricted programs. We have also shown that the possible model semantics has a computational advantage over the minimal/stable model semantics.

In normal disjunctive programs, we have defined the possible model semantics based on the stable model semantics. However, since its definition is given through the set of split normal logic programs, it is easy to construct another version of the possible model semantics based on any semantics of normal logic programs other than the stable model semantics. In this sense, the possible model semantics presented in this paper provides a fairly general framework independent of any specific semantics. In other words, it establishes the principle of *possibilism* as a semantical counterpart of the traditional minimalism, which contributes to enriching our perspectives for commonsense reasoning in logic programming and artificial intelligence.

The possible model semantics presented in this paper is also directly extensible to disjunctive logic programs with classical negation [15]. Moreover, recent studies have revealed that the possible model semantics is also useful for abductive reasoning in logic programming [37] and has a close relation to autoepistemic logic [19].

Appendix

Here we present the proofs of Theorem 4.6 and Corollary 4.7.

THEOREM 4.6. *Let P be a normal disjunctive program and P^{κ} be its epistemic transformation. Then,*

$$\mathcal{ST}_P = obj_c\big(\min\big(\mu(\mathcal{I}^{\omega}_{P^{\kappa}})\big)\big).$$

In particular, P is incoherent iff $obj_c(\min(\mu(\mathcal{I}^{\omega}_{P^{\kappa}}))) = \emptyset$.

Proof. Suppose that I is in $obj_c(\min(\mu(\mathcal{I}^{\omega}_{P^{\kappa}})))$. Let I^{κ} be a canonical interpretation in $\min(\mu(\mathcal{I}^{\omega}_{P^{\kappa}}))$ such that $I^{\kappa} \cap \mathcal{HB}_P = I$. Then, for each ground clause of the form (2) from P, $\{B_1, \dots, B_m\} \subseteq I^{\kappa}$ implies either (i) $\exists\lambda_i \in I^{\kappa}$ $(1 \leqslant i \leqslant l)$, $A_i \in I^{\kappa}$, and $\{B_{m+1}, \dots, B_n\} \cap I^{\kappa} = \emptyset$ or (ii) $\exists \mathrm{K} B_j \in I^{\kappa}$ $(m+1 \leqslant j \leqslant n)$ by (3), (4), and (5).*

In case of (i), $\{B_{m+1}, \dots, B_n\} \cap I^{\kappa} = \emptyset$ implies $\{B_{m+1}, \dots B_n\} \cap I = \emptyset$. Then there is a clause of the form

$$A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m \tag{$*$}$$

in P^I. Since $\{B_1, \dots, B_m\} \subseteq I^{\kappa}$ and $A_i \in I^{\kappa}$ implies $\{B_1, \dots, B_m\} \subseteq I$ and $A_i \in I$, I satisfies the clause $(*)$ in P^I. In case of (ii), since I^{κ} is canonical, $\exists \mathrm{K} B_j \in I^{\kappa}$ implies $B_j \in I^{\kappa}$, and thus $B_j \in I$. In this case, the clause $(*)$ is not included in P^I. Thus, in both cases, I satisfies every clause in P^I.

* When (2) contains no *not*, $\{B_1, \dots, B_m\} \subseteq I^{\kappa}$ implies $A_i \in I^{\kappa}$ $(1 \leqslant i \leqslant l)$ as a special case of (i).

Suppose that there is an interpretation J such that (a) $J \subset I$ and (b) J satisfies each clause from P^I. Then, two conditions (a) and (b) are satisfied only if there is a clause $(*)$ such that $\{B_1, \ldots, B_m\} \subseteq J$ and for some two atoms A_{i_1} and A_{i_2} $(1 \leqslant i_1, i_2 \leqslant l;\ i_1 \neq i_2)$, $A_{i_1} \in J$ but $A_{i_2} \in I \setminus J$. Without loss of generality, we can assume that just one such clause exists in P^I. Since I does not contain atoms $B_{m+1}, \ldots, B_n$, the corresponding canonical interpretation I^κ does not contain $\mathrm{K}B_{m+1}, \ldots, \mathrm{K}B_n$ either. Thus, $\{B_1, \ldots, B_m\} \subseteq I$ implies $\exists \lambda_k \in I^\kappa$ for some $1 \leqslant k \leqslant l$. Since $A_{i_1}, A_{i_2} \in I$ implies $A_{i_1}, A_{i_2} \in I^\kappa$, $\lambda_k \in I^\kappa$ implies $\lambda_{i_1}, \lambda_{i_2} \in I^\kappa$ by (6). Let $J^\kappa = I^\kappa \setminus \{A_{i_2}, \lambda_{i_2}\}$. Then, the interpretation J^κ satisfies all the clauses (3), (4), (5), (6) in P^κ. This contradicts the fact that I^κ is a minimal model of P^κ. Then I is also a minimal model of P^I, hence a stable model of P.

Conversely, suppose that I is a stable model of P. Let $I = I^\kappa \cap \mathcal{HB}_P$ for some interpretation I^κ of P^κ. Then, for each atom A_i in I, there is a ground clause of the form $(*)$ in P^I such that $1 \leqslant i \leqslant l$ and $\{B_1, \ldots, B_m\} \subseteq I^\kappa$. In this case, there are corresponding clauses (3), (4), and (5) in P^κ such that $\lambda_i \in I^\kappa$, $A_i \in I^\kappa$, and $\{B_{m+1}, \ldots, B_n\} \cap I^\kappa = \emptyset$. Hence $I^\kappa \in \mu(\mathcal{I}^\omega_{P^\kappa})$. Also we can choose I^κ to be a minimal set such that $\{\mathrm{K}B_{m+1}, \ldots, \mathrm{K}B_n\} \cap I^\kappa = \emptyset$, then $I^\kappa \in \min(\mu(\mathcal{I}^\omega_{P^\kappa}))$ and I^κ is canonical with respect to $B_{m+1}, \ldots, B_n$. On the other hand, for such I^κ assume that $\mathrm{K}B_j \in I^\kappa$ for some j $(m + 1 \leqslant j \leqslant n)$. Then there exists a clause of the form (3) in P^κ such that $\{B_1, \ldots, B_m\} \subseteq I^\kappa$. Since I^κ is minimal, $\lambda_i \notin I^\kappa$ for any i $(1 \leqslant i \leqslant l)$. Then A_i is not derived from (3), so the clause $(*)$ is not in P^I. In this case, there exists some k $(m + 1 \leqslant k \leqslant n)$ such that $B_k \in I$, and we can choose k as $k = j$ so that B_j is in I^κ. Therefore, I^κ is canonical, and thus $I \in obj_c(\min(\mu(\mathcal{I}^\omega_{P^\kappa})))$. □

COROLLARY 4.7. *Let P be a normal logic program. Then,*

$$\mathcal{ST}_P = obj_c(\mu(\mathcal{I}^\omega_{P^\kappa})).$$

Proof. By Theorem 4.6, $obj_c(\min(\mu(\mathcal{I}^\omega_{P^\kappa})))$ is the set of all stable models of P. Since $I \in obj_c(\min(\mu(\mathcal{I}^\omega_{P^\kappa})))$ implies $I \in obj_c(\mu(\mathcal{I}^\omega_{P^\kappa}))$, we show that the converse is also true. Assume that the converse does not hold. That is, there is a nonminimal set $I \in obj_c(\mu(\mathcal{I}^\omega_{P^\kappa}))$ and $\exists J \in obj_c(\min(\mu(\mathcal{I}^\omega_{P^\kappa})))$ such that $J \subset I$. In this case, there exists an atom A such that $A \in I$ and $A \notin J$. Let $I = I^\kappa \cap \mathcal{HB}_P$ and $J = J^\kappa \cap \mathcal{HB}_P$ for some canonical interpretations I^κ and J^κ. Then, corresponding to (3), (4), and (5), there exist clauses

$$\lambda \vee \mathrm{K}B_{m+1} \vee \cdots \vee \mathrm{K}B_n \leftarrow B_1 \wedge \cdots \wedge B_m,$$

$$A \leftarrow \lambda,$$

$$\leftarrow \lambda \wedge B_j \quad (j = m + 1, \ldots, n)$$

in P^κ, where $I^\kappa, J^\kappa \models B_1, \ldots, B_n$, $\lambda \in I^\kappa$, and $\exists \mathrm{K}B_j \in J^\kappa$ $(m + 1 \leqslant j \leqslant n)$. Note here that the clause (6) becomes $\lambda \leftarrow A \wedge \lambda$ and is neglected. Since J^κ is canonical, $B_j \in J^\kappa$, hence $B_j \in I^\kappa$. But this is impossible from the third clause above. □

Acknowledgement

We are grateful to anonymous referees for useful comments on the previous draft of this paper.

References

1. Apt, K. R., Blair, H. A. and Walker, A.: Towards a theory of declarative knowledge, in J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, 1988, pp. 89–148.
2. Baral, C., Lobo, J. and Minker, J.: Generalized disjunctive well-founded semantics for logic programs, *Ann. Mathematics and Artificial Intelligence* 5 (1992), 89–132.
3. Bancilhon, F. and Ramakrishnan, R.: Performance evaluation of data intensive logic programs, in J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, 1988, pp. 439–517.
4. Chan, E. P. F.: A possible world semantics for disjunctive databases, *IEEE Trans. on Knowledge and Data Engineering* 5(2) (1993), 282–292. Preliminary version in: Research Report CS-89-47, Dept. of Computer Science, Univ. of Waterloo, 1989.
5. Clark, K. L.: Negation as failure, in H. Gallaire and J. Minker (eds.), *Logic and Data Bases*. II, Plenum, New York, 1978, pp. 293–322.
6. Decker, H.: Foundations of first-order databases, Research Report, Siemens, 1992. Preliminary version in *Proc. 2nd Int. Workshop on the Deductive Approach to Information Systems and Databases*, Universitat Politecnica de Catalunya, Report de Recerca LSI/91/30, 1991, pp. 149–173.
7. Decker, H. and Casamayor, J. C.: Sustained models and sustained answers in first order databases, *Proc. 4th Int. Workshop on the Deductive Approach to Information Systems and Databases*, 1993.
8. Dix, J.: Classifying semantics of disjunctive logic programs, *Proc. Joint Int. Conf. and Symp. on Logic Programming*, MIT Press, 1992, pp. 798–812.
9. Dung, P. M.: Negation as failure for disjunctive logic programming, *Proc. ILPS'91 Post-Conference Workshop on Disjunctive Logic Programs*, 1991.
10. Eiter, T. and Gottlob, G.: Complexity aspects of various semantics for disjunctive databases, *Proc. 12th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, 1993, pp. 158–167.
11. Eiter, T., Gottlob, G. and Gurevich, Y.: Curb Your Theory!: A circumscriptive approach for inclusive interpretation of disjunctive information, *Proc. IJCAI-93*, Morgan Kaufmann, 1993, pp. 634–639.
12. Eshghi, K. and Kowalski, R. A.: Abduction compared with negation by failure, *Proc. 6th Int. Conf. on Logic Programming*, MIT Press, 1989, pp. 234–254.
13. Fernandez, J. A., Lobo, J., Minker, J. and Subrahmanian, V. S.: Disjunctive LP + integrity constraints = stable model semantics, *Ann. Mathematics and Artificial Intelligence* 8(3&4) (1993), 449–474.
14. Gelfond, M. and Lifschitz, V.: The Stable model semantics for logic programming, *Proc. 5th Int. Conf. and Symp. on Logic Programming*, MIT Press, 1988, pp. 1070–1080.
15. Gelfond, M. and Lifschitz, V.: Classical negation in logic programs and disjunctive databases, *New Generation Computing* 9(3&4) (1991), 365–385.
16. Gelfond, M.: Strong introspection, *Proc. AAAI-91*, MIT Press, 1991, pp. 386–391.
17. 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, Springer-Verlag, 1992, 400–415.
18. Inoue, K. and Sakama, C.: Transforming abductive logic programs to disjunctive programs, *Proc. 10th Int. Conf. on Logic Programming*, MIT Press, 1993, pp. 335–353.
19. Inoue, K. and Sakama, C.: On positive occurrences of negation as failure, *Proc. 4th Int. Conf. on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1994, pp. 293–304.
20. Kraus, S., Lehmann, D. and Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics, *Artificial Intelligence* 44(1) (1990), 167–207.
21. Lobo, J., Minker, J. and Rajasekar, A.: *Foundations of Disjunctive Logic Programming*, MIT Press, 1992.
22. 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, Springer-Verlag, 1988, pp. 415–434.
23. McCarthy, J.: Circumscription – a form of nonmonotonic reasoning, *Artificial Intelligence* 13(1&2) (1980), 27–39.
24. 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, 1982, pp. 292–308.

25. Marek, W. and Subrahmanian, V. S.: The relationship between stable, supported, default and autoepistemic semantics for general logic programs, *Theoretical Computer Science* **103** (1992), 365–386.
26. Marek, W. and Truszczynski, M.: Autoepistemic logic, *J. ACM* **38**(3) (1991), 588–619.
27. Marek, W. and Truszczynski, M.: Computing intersection of autoepistemic expansions, *Proc. 1st Int. Workshop on Logic Programming and Nonmonotonic Reasoning*, MIT Press, 1991, 37–50.
28. Przymusinski, T. C.: On the declarative semantics of deductive databases and logic programs, in J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, 1988, pp. 193–216.
29. Przymusinski, T. C.: Stable semantics for disjunctive programs, *New Generation Computing* **9**(3&4) (1991), 401–424.
30. Przymusinski, T. C.: Semantics of disjunctive logic programs and deductive databases, *Proc. 2nd Int. Conf. on Deductive and Object-Oriented Databases*, Lecture Notes in Computer Science 566, Springer-Verlag, 1991, pp. 85–107.
31. Reiter, R.: On closed world databases, in H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, II, Plenum, New York, 1978, pp. 55–76.
32. Rajasekar, A., Lobo, J. and Minker, J.: Weak generalized closed world assumption, *J. Automated Reasoning* **5** (1989), 293–307.
33. Ross, K.: The well founded semantics for disjunctive logic programs, *Proc. 1st Int. Conf. on Deductive and Object-Oriented Databases*, North-Holland, 1989, pp. 385–401.
34. Ross, K. A. and Topor, R. W.: Inferring negative information from disjunctive databases, *J. Automated Reasoning* **4**(2) (1988), 397–424.
35. Sakama, C.: Possible model semantics for disjunctive databases, *Proc. 1st Int. Conf. on Deductive and Object-Oriented Databases*, North-Holland, 1989, pp. 369–383.
36. Sakama, C. and Inoue, K.: Negation in disjunctive logic programs, *Proc. 10th Int. Conf. on Logic Programming*, MIT Press, 1993, pp. 703–719.
37. Sakama, C. and Inoue, K.: On the equivalence between disjunctive and abductive logic programs, *Proc. 11th Int. Conf. on Logic Programming*, MIT Press, 1994, pp. 489–503.
38. Sato, T.: Completed logic programs and their consistency, *J. Logic Programming* **9**(1), 1990, pp. 33–44.
39. Schlipf, J. S.: Formalizing a logic for logic programming. *Ann. Mathematics and Artificial Intelligence* **5** (1992), 279–302.
40. Van Emden, M. H. and Kowalski, R. A.: The semantics of predicate logic as a programming language, *J. ACM* **23**(4) (1976), 733–742.