Paraconsistent Stable Semantics for Extended Disjunctive Programs

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

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

Abstract

This paper presents declarative semantics of possibly inconsistent disjunctive logic programs. We introduce the paraconsistent minimal and stable model semantics for extended disjunctive programs, which can distinguish inconsistent information from other information in a program. These semantics are based on lattice-structured multi-valued logics, and are characterized by a new fixpoint semantics of extended disjunctive programs. Applications of the paraconsistent semantics for reasoning in inconsistent programs are also presented.

Keywords: Extended disjunctive programs, inconsistency, multi-valued logic, paraconsistent stable model semantics.

1 Introduction

Representing and reasoning with incomplete information in a program is one of the central issues in recent studies of logic programming. *Extended disjunctive programs* introduced by Gelfond and Lifschitz [15] provide a fairly general framework for that purpose. An extended disjunctive program can specify incomplete information by using classical negation as well as disjunctions in a program. In the presence of such explicit negation in a program, however, an extended disjunctive program possibly becomes inconsistent, since negative consequences are allowed in the program. In [15], a declarative semantics of extended disjunctive programs is given by the notion of *answer sets*, which is a generalization of stable models of normal disjunctive programs. However, the problem of the answer set semantics is that the answer set becomes trivial in an inconsistent program and implies every formula from the program. This is also the case for most of the traditional logic programming semantics in which local inconsistency might spoil the whole program. Practically speaking, when we build a large-scale knowledge base in a logic programming framework, inconsistent information as well as incomplete information is likely to happen in the knowledge base. In such a knowledge base, a piece of contradictory information would make the whole program inconsistent, but still the program may contain meaningful information which is not affected by the local inconsistency.

The *paraconsistent logics* are logics which are not destructive in the presence of inconsistent information. In these logics, the contradictory statement $A \wedge \neg A$ does not deduce an arbitrary formula, hence would not trivialize the whole theory. In this regard, paraconsistent logics can localize inconsistent information in a theory and serve as useful inference tools in artificial

intelligence. Historically, paraconsistent logics have been developed in the area of philosophical logic [1], and a formal framework for inconsistent theories was given by da Costa [7]. Applications of paraconsistent logics to logic programming have also been investigated by several researchers. Blair and Subrahmanian [5] firstly introduced a framework of *paraconsistent logic programming*. They extended Fitting's three-valued semantics of logic programming [11] and developed a theory for possibly inconsistent logic programs using Belnap's *four-valued logic* [4]. The result was generalized by Subrahmanian [32] to programs possibly containing disjunctive information. Recently, the paraconsistent logic programming framework was further extended to treat default negation along with explicit negation in a program [28, 35, 21, 17]. However, in the context of extended disjunctive programs, a suitable paraconsistent extension of the answer set semantics has not been studied in the literature.

In this paper, we present declarative semantics of possibly inconsistent extended disjunctive programs. We introduce the paraconsistent minimal and stable model semantics for extended disjunctive programs, which can distinguish inconsistent information from other information in a program. The proposed semantics are based on lattice-structured multi-valued logics, and are characterized by a new fixpoint semantics of extended disjunctive programs. We also present applications of the paraconsistent semantics for reasoning in inconsistent programs.

The rest of this paper is organized as follows. In Section 2, we first present the paraconsistent minimal model semantics for positive extended disjunctive programs. We introduce a new fixpoint semantics and characterize the paraconsistent minimal model semantics of positive extended disjunctive programs. The result is extended in Section 3 to the paraconsistent stable model semantics for extended disjunctive programs. A fixpoint characterization of the paraconsistent stable model semantics is also presented. In Section 4, we address applications of the paraconsistent stable model semantics for reasoning with inconsistency. The notions of preferred stable models, suspicious stable models, and semi-stable models are introduced as variants of the paraconsistent stable model semantics. Section 5 discusses comparison with related work, and Section 6 concludes this paper.

2 Paraconsistent semantics for positive extended disjunctive programs

2.1 *Positive extended disjunctive programs*

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

$$L_1 \vee \ldots \vee L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m \quad (m \geq l \geq 0) \tag{2.1}$$

where L_is are positive or negative literals. When L is a positive (resp. negative) literal, $\neg L$ denotes its complementary negative (resp. positive) literal and $L = \neg\neg L$ holds as usual. The left-hand side of the clause is called the *head* and the right-hand side of the clause is called the *body*. A clause is called *disjunctive* if its head contains more than one literal. A clause is called an *integrity constraint* if it has an empty head and a non-empty body. A positive extended disjunctive program containing no disjunctive clause is called a *positive extended logic program*. A positive extended disjunctive program is called a *positive disjunctive program* if all L_is are atoms.

In this section, a program means a positive extended disjunctive program unless stated otherwise. As usual, we semantically identify a program with its *ground program*, which is the possibly infinite set of all ground clauses from the program. In positive extended disjunctive programs, negative literals have the same status as positive literals, then we consider the *Herbrand base* of a program P as the set of all ground literals $\mathcal{L}_P$ from the language of P.

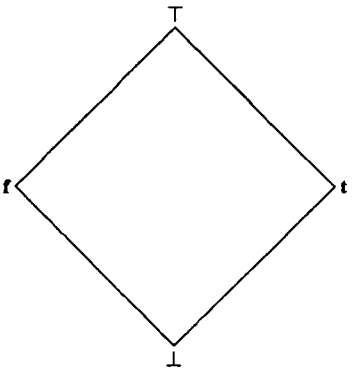

FIG. 1. Four-valued lattice IV

The set of truth values of our *four-valued logic* is defined as $IV = \{\mathbf{t}, \mathbf{f}, \top, \bot\}$, in which $\mathbf{t}$, $\mathbf{f}$, $\top$, $\bot$ are propositions in the language of a program and respectively denote *true*, *false*, *contradictory*, and *undefined*. For simplicity, we assume that programs do not contain these reserved propositions and $IV \not\subset \mathcal{L}_P$.[1] The set of truth values IV makes a complete lattice under the ordering $\preceq$ such that $\bot \preceq \mathbf{x} \preceq \top$ for $\mathbf{x} \in \{\mathbf{t}, \mathbf{f}\}$ (Figure 1). Such a lattice is also known as Belnap's four-valued logic [4].[2]

Let I be a subset of $\mathcal{L}_P$. An *interpretation* of a program P is defined as a function $I : \mathcal{L}_P \to IV$ such that for each literal $L \in \mathcal{L}_P$,

$$I(L) = \begin{cases} \mathbf{t} & \text{if } L \in I \text{ and } \neg L \notin I, \\ \mathbf{f} & \text{if } \neg L \in I \text{ and } L \notin I, \\ \top & \text{if both } L \in I \text{ and } \neg L \in I, \\ \bot & \text{otherwise.} \end{cases}$$

Note that $I(L) = \mathbf{t}$ iff $I(\neg L) = \mathbf{f}$, $I(L) = \top$ iff $I(\neg L) = \top$, and $I(L) = \bot$ iff $I(\neg L) = \bot$.

In this paper, when no confusion arises, we identify a set of literals I with its interpretation $I(L)$ for each $L \in I$. For instance, we identify $I = \{L\}$ with $I(L) = \mathbf{t}$; $I = \{L, \neg L\}$ with $I(L) = \top$; $I = \emptyset$ with $I(L) = \bot$ for any $L \in \mathcal{L}_P$, and so on.

Satisfaction (denoted by $\models$) of each clause in a program P is inductively defined as follows.

DEFINITION 2.1
Let P be a positive extended disjunctive program and I be an interpretation. Then:

1. For any literal $L \in \mathcal{L}_P$,
   (a) $I \models L$ iff $\mathbf{t} \preceq I(L)$,
   (b) $I \models \neg L$ iff $\mathbf{f} \preceq I(L)$.
2. For any disjunction of ground literals $F = L_1 \vee \ldots \vee L_n$,
   $I \models F$ iff $I \models L_i$ for some i $(1 \leq i \leq n)$.

[1] This assumption is not essential and can be removed, but this issue will not be discussed in this paper.

[2] Note that the order considering here is the so-called *knowledge ordering*, while there is the alternative *truth ordering* in the context of *bilattices* [16, 12].

3. For any conjunction of ground literals $G = L_1 \wedge \ldots \wedge L_n$,
   $I \models G$ iff $I \models L_i$ for every i $(1 \leq i \leq n)$.
4. For any ground clause $C = F \leftarrow G$, $I \models C$ iff $I \models F$ or $I \not\models G$.
   In particular, $I \models \leftarrow G$ iff $I \not\models G$, and $I \models F \leftarrow$ iff $I \models F$.

An interpretation I is a *model* of a program P if I satisfies every ground clause from P. The ordering $\preceq$ on truth values is also defined between interpretations. For interpretations I and J, $I \preceq J$ iff $I(L) \preceq J(L)$ for any $L \in \mathcal{L}_P$. The orderings $\succeq$, $\prec$, $\succ$ are defined in the usual way. Note that when we identify a set of literals with its interpretation, $I \preceq J$ iff $I \subseteq J$. A model I is *minimal* if there is no model J such that $J \prec I$. A model I is *least* if $I \preceq J$ for every model J. To distinguish terms from standard logic programming, a minimal/least model is also called a *paraconsistent minimal/least model* (shortly, *p-minimal/p-least model*). A *consistent model* is a model I such that $I(L) \neq \top$ for any $L \in \mathcal{L}_P$, otherwise I is an *inconsistent model*. A program is *consistent* if it has a consistent model, otherwise it is *inconsistent*.

PROPOSITION 2.2
If a positive extended disjunctive program has a model, it has at least one p-minimal model.

PROOF. Let us consider a decreasing sequence of models $I_1 \supseteq I_2 \supseteq \ldots$ and their greatest lower bound $I = \bigcap_{i \geq 1} I_i$. Then, for each ground clause $F \leftarrow G$ from a program P, if $I \models G$, $I_i \models G$ for any $i \geq 1$. In this case, since each I_i is a model of P, F is not empty and $I_i \models F$ holds for any $i \geq 1$. Thus $I \models F$. Hence, I is also a model of P, and by definition it is a p-minimal model. ∎

PROPOSITION 2.3
A consistent positive extended disjunctive program has a consistent p-minimal model.

PROOF. When a program P is consistent, it has a consistent model I by definition. If I is not minimal, there is a p-minimal model J such that $J \prec I$ by Proposition 2.2. Then, $I(L) \neq \top$ for any $L \in \mathcal{L}_P$ implies $J(L) \neq \top$ for any $L \in \mathcal{L}_P$. ∎

COROLLARY 2.4
If a positive extended logic program has a model, it has the unique p-least model. In particular, a consistent positive extended logic program has the consistent p-least model.

EXAMPLE 2.5
Let P be the program:

$$\{\, a \vee b \leftarrow,\quad \neg a \leftarrow,\quad \neg b \leftarrow,\quad c \leftarrow \,\}.$$

Then P has two p-minimal models $\{a, \neg a, \neg b, c\}$ and $\{b, \neg a, \neg b, c\}$.

Note that the above program is inconsistent and the classical minimal model semantics makes the program trivial, while the p-minimal models retain truth information about c that is not affected by the inconsistency.

REMARK 2.6
1. In extended disjunctive programs, the meanings of the clauses $\leftarrow L$ and $\neg L \leftarrow$ are different. This is also the case in our multi-valued setting. In fact, $I(L) = \bot$ is a model of the first clause, while it is not a model of the second clause. Such a difference is due to the fact that the connective $\leftarrow$ is non-contrapositive in extended disjunctive programs [15].
2. Corresponding to the above fact, the program $\{L \leftarrow,\ \neg L \leftarrow\}$ has a model $I(L) = \top$, while $\{L \leftarrow,\ \leftarrow L\}$ has no model. That is, we consider any interpretation meaningless if it does not satisfy integrity constraints. However, it is easy to construct a paraconsistent theory for integrity violation if desired.

2.2 *Fixpoint semantics*

In this section, we introduce a new fixpoint semantics of positive extended disjunctive programs to characterize the paraconsistent minimal model semantics presented in the previous section. In contrast to logic programs containing only definite information, a positive extended disjunctive program has multiple p-minimal models in general. In order to characterize such non-deterministic behaviour of disjunctive programs, we first introduce a closure operator which acts over the lattice of sets of Herbrand interpretations $2^{2^{\mathcal{L}_P}}$.

DEFINITION 2.7
Let P be a positive extended disjunctive program and $\mathcal{I}$ be a set of interpretations. Then a mapping $\mathcal{T}_P : 2^{2^{\mathcal{L}_P}} \rightarrow 2^{2^{\mathcal{L}_P}}$ is defined as

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

where the mapping $T_P : 2^{\mathcal{L}_P} \rightarrow 2^{2^{\mathcal{L}_P}}$ is defined as follows:

$$T_P(I) = \begin{cases} \emptyset, & \text{if } \{L_1, \ldots, L_m\} \subseteq I \text{ for some ground integrity constraint} \\ & \quad \leftarrow L_1 \wedge \ldots \wedge L_m \text{ from } P; \\ \{\ J \mid & \text{for each ground clause } C_i : \ L_1 \vee \ldots \vee L_{l_i} \leftarrow L_{l_i+1} \wedge \ldots \wedge L_{m_i} \\ & \quad \text{from } P \text{ such that } \{L_{l_i+1}, \ldots, L_{m_i}\} \subseteq I, \\ & \quad J = I \cup \bigcup_{C_i} \{L_j\} \ \ (1 \leq j \leq l_i)\ \}, \quad \text{otherwise}. \end{cases}$$

Thus, $T_P(I)$ is the set of interpretations Js such that for each clause C_i whose body is satisfied by I, I is expanded into J by adding one disjunct L_j from the heads of every such C_i. In particular, if I does not satisfy an integrity constraint from P, I is removed in $T_P(I)$.

EXAMPLE 2.8
Let P be the program:

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

Then, $T_P(\{c\}) = \{\{c, \neg d, a\}, \{c, \neg d, b\}\}$ and $\mathcal{T}_P(\{\{c, \neg d, a\}, \{c, \neg d, b\}\}) = \{\{c, \neg d, a\}, \{c, \neg d, b\}, \{c, \neg d, a, b\}\}$.

DEFINITION 2.9
The ordinal powers of $\mathcal{T}_P$ are defined as follows:

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

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

The above definition means that at the limit ordinal ω the closure retains interpretations which are persistent in the preceding iterations. That is, for any interpretation I in $\mathcal{T}_P \uparrow \omega$, there is an ordinal α smaller than ω such that, for every n $(\alpha \leq n < \omega)$, I is included in $\mathcal{T}_P \uparrow n$. Such a closure definition is also used in [9, 33] for computing stable models of normal logic programs.

THEOREM 2.10
$\mathcal{T}_P \uparrow \omega$ is a fixpoint.

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

EXAMPLE 2.11 (continued from Example 2.8)
Given the program P, it becomes

$$\mathcal{T}_P \uparrow 1 = \{\{c\}\},$$
$$\mathcal{T}_P \uparrow 2 = \{\{c, \neg d, a\}, \{c, \neg d, b\}\},$$
$$\mathcal{T}_P \uparrow 3 = \{\{c, \neg d, a\}, \{c, \neg d, b\}, \{c, \neg d, a, b\}\},$$
$$\mathcal{T}_P \uparrow 4 = \{\{c, \neg d, a\}, \{c, \neg d, b\}, \{c, \neg d, a, b\}\},$$

where $\mathcal{T}_P \uparrow \omega = \mathcal{T}_P \uparrow 3$.

In the above example, the interpretation $\{c, \neg d, a, b\}$ in $\mathcal{T}_P \uparrow 3$ is pruned in $\mathcal{T}_P \uparrow 4$ by the integrity constraint $\leftarrow a \wedge b$, while the same interpretation is also generated from $\{c, \neg d, a\}$ and $\{c, \neg d, b\}$ in $\mathcal{T}_P \uparrow 3$, hence $\{c, \neg d, a, b\}$ remains in $\mathcal{T}_P \uparrow 4$.

By definition, the fixpoint closure presented above exists for any program and is uniquely determined. Intuitively, the fixpoint characterizes a set of interpretations which are 'generated' in a program by starting from the empty interpretation. Next we show that the fixpoint closure in fact contains what we want, i.e. the set of all p-minimal models.

LEMMA 2.12
Let P be a positive extended disjunctive program. Then I is a model of P iff $I \in \mathcal{T}_P(\{I\})$.

PROOF. I is a model of P
iff it satisfies integrity constraints and for each clause $L_1 \vee \ldots \vee L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m$ in P, $\{L_{l+1}, \ldots, L_m\} \subseteq I$ implies $L_i \in I$ for some L_i $(1 \leq i \leq l)$
iff $I \in \mathcal{T}_P(\{I\})$. ∎

LEMMA 2.13
If I is a p-minimal model of P, then, for each literal L in I, there is a ground clause $L_1 \vee \ldots \vee L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m$ from P such that $\{L_{l+1}, \ldots, L_m\} \subseteq I$ and $L = L_i$ for some i $(1 \leq i \leq l)$.

PROOF. If no such clause exists, there is a model J of P such that $J(L) = \bot$ and $J(L') = I(L')$ for every literal $L' \in \mathcal{L}_P$ other than L. Then $J \prec I$, which contradicts the assumption that I is minimal. ∎

Let $\mu(\mathcal{T}_P \uparrow \omega) = \{I \mid I \in \mathcal{T}_P \uparrow \omega \text{ and } I \in \mathcal{T}_P(\{I\})\}$. Then, by Lemma 2.12, $\mu(\mathcal{T}_P \uparrow \omega)$ represents the set of models of P which are included in the fixpoint closure. Also, let $min(\mathcal{I}) = \{ I \in \mathcal{I} \mid \nexists J \in \mathcal{I} \text{ such that } J \subset I \}$. Then the following result holds.

THEOREM 2.14
Let P be a positive extended disjunctive program and $\mathcal{PMM}_P$ be the set of all p-minimal models of P. Then,

$$\mathcal{PMM}_P = min(\mu(\mathcal{T}_P \uparrow \omega)).$$

PROOF. By definition and Lemma 2.12, each element in $\mu(\mathcal{T}_P \uparrow \omega)$ is a model of P. Thus, each element in $min(\mu(\mathcal{T}_P \uparrow \omega))$ is a p-minimal model of P. On the other hand, let I be a p-minimal model of P. Then, for each literal L in I, there is a ground clause $L_1 \vee \ldots \vee L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m$ from P such that $\{L_{l+1}, \ldots, L_m\} \subseteq I$ and $L = L_i$ for some i $(1 \leq i \leq l)$ (by Lemma 2.13). Then, by the definition of the fixpoint construction, I is included in $\mathcal{T}_P \uparrow \omega$. Since each element in $\mu(\mathcal{T}_P \uparrow \omega)$ is a model of P, I is also included in $min(\mu(\mathcal{T}_P \uparrow \omega))$. ∎

EXAMPLE 2.15 (continued from Example 2.11)
$min(\mu(\mathcal{T}_P \uparrow \omega)) = \{\{c, \neg d, a\}, \{c, \neg d, b\}\}$ which contains the p-minimal models of P.

For positive extended logic programs, the following result holds.

COROLLARY 2.16
Let P be a positive extended logic program. Then $\mathcal{T}_P \uparrow \omega$ contains the unique p-least model of P.

The above corollary corresponds to Blair and Subrahmanian's fixpoint semantics of paraconsistent logic programs [5], and also reduces to van Emden and Kowalski's fixpoint semantics in definite logic programs [34].

For positive disjunctive programs, our fixpoint construction characterizes Sakama's *possible model semantics* [29][3] and Minker's minimal model semantics [24]. Let $\mathcal{PM}_P$ (resp. $\mathcal{MM}_P$) be the set of all possible models (resp. minimal models) of a positive disjunctive program P.

THEOREM 2.17
Let P be a positive disjunctive program. Then,

(i) $\mathcal{PM}_P = \mu(\mathcal{T}_P \uparrow \omega)$,
(ii) $\mathcal{MM}_P = min(\mu(\mathcal{T}_P \uparrow \omega))$.

The above (i) also indicates that for a positive extended disjunctive program P, the fixpoint closure $\mu(\mathcal{T}_P \uparrow \omega)$ characterizes the paraconsistent extension of the possible model semantics. We do not discuss here the detailed definition and properties of the possible model semantics and refer the reader to the literature [29, 31].

3 Paraconsistent semantics for extended disjunctive programs

This section extends the results presented in the previous section to extended disjunctive programs in general.

3.1 Paraconsistent stable models

An *extended disjunctive program* is a finite set of clauses of the form:

$$L_1 \vee \ldots \vee L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m \wedge not\ L_{m+1} \wedge \ldots \wedge not\ L_n \quad (n \geq m \geq l \geq 0) \qquad (3.1)$$

where L_is are literals and *not* is a connective representing *default negation*. The notions of heads, bodies, disjunctive clauses, and integrity constraints are defined in the same way as those of positive extended disjunctive programs. An extended disjunctive program containing no disjunctive clause is called an *extended logic program*. An extended disjunctive program

[3]The possible model semantics is equivalent to the *possible world semantics* in [6].

is called a *normal disjunctive program* if all L_is are atoms, and an extended logic program is called a *normal logic program* if all L_is are atoms. For an extended disjunctive program, its ground program and the Herbrand base $\mathcal{L}_P$ are defined as before. Hereafter, a program means an extended disjunctive program unless stated otherwise.

In an extended disjunctive program, the notion of interpretations and satisfaction of literals and clauses are defined in the same manner as in Definition 2.1 except that for each formula $not\ L$, we include the additional statements:

- $I \models not\ L$ iff $I(L) \preceq \mathbf{f}$,
- $I \models not\ \neg L$ iff $I(L) \preceq \mathbf{t}$.

The first condition indicates that if L is false in I, its default negation $not\ L$ holds in I; else if L is undefined in I, $not\ L$ holds in I as negation as failure to prove; otherwise $I \not\models not\ L$. The second condition gives the counterpart statement.

The notions of (p-minimal) models can be defined in the same way as in Section 2.1.

The *paraconsistent stable model semantics* of an extended disjunctive program is defined as follows.

DEFINITION 3.1
Let P be an extended disjunctive program and I be a subset of $\mathcal{L}_P$. The *reduct* of P with respect to I is the positive extended disjunctive program P^I such that a clause

$$L_1 \vee \ldots \vee L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m \tag{3.2}$$

is in P^I iff there is a ground clause of the form (3.1)

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

from P such that $\{L_{m+1}, \ldots, L_n\} \cap I = \emptyset$. Then I is called a *paraconsistent stable model* (shortly, *p-stable model*) of P if I is a p-minimal model of P^I.

EXAMPLE 3.2
Let P be the program:

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

Then P has two p-stable models $\{a, \neg a, \neg b, c\}$ and $\{b, \neg a, \neg b, c\}$.

A p-stable model I is *consistent* if $I(L) \neq \top$ for any $L \in \mathcal{L}_P$, otherwise I is *inconsistent*. There is a program which has no p-stable model.

EXAMPLE 3.3
The program

$$P = \{\ a \leftarrow not\ a,\quad b \leftarrow\ \}$$

has no p-stable model.

A program which has at least one p-stable model is called *coherent*, while a program having no p-stable model is called *incoherent*. By definition, the notion of p-stable models reduces to that of p-minimal models in positive extended disjunctive programs. In extended disjunctive programs, every p-stable model is minimal.

PROPOSITION 3.4
A p-stable model is a p-minimal model.

PROOF. Let I be a p-stable model of a program P. Assume that there is a p-minimal model J of P such that $J \preceq I$. Since J is a model of P and satisfies each clause (3.1) in P, and $\{L_{m+1}, \ldots, L_n\} \cap I = \emptyset$ and $J \preceq I$ imply $\{L_{m+1}, \ldots, L_n\} \cap J = \emptyset$, J also satisfies each clause (3.2) in P^I. Thus J is a model of P^I, and since I is a p-minimal model of P^I, $J \preceq I$ implies $J = I$. ∎

The converse of the above proposition does not hold in general. For instance, in Example 3.3, $\{a, b\}$ is the p-minimal model of P, but not p-stable.

3.2 *Fixpoint semantics of extended disjunctive programs*

In this section, we characterize the p-stable models of extended disjunctive programs using the fixpoint semantics presented in the previous section. To this end, we first introduce a program transformation which translates an extended disjunctive program into a semantically equivalent positive extended disjunctive program.[4]

DEFINITION 3.5
Let P be an extended disjunctive program. Then its *epistemic transformation* is defined as the positive extended disjunctive program P^κ obtained from P by replacing each clause of the form (3.1) in P containing default negation:

$$L_1 \vee \ldots \vee L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m \wedge not\, L_{m+1} \wedge \ldots \wedge not\, L_n \quad (m \neq n)$$

with the following *not*-free clauses in P^κ:

$$\lambda_1 \vee \ldots \vee \lambda_l \vee \mathsf{K}L_{m+1} \vee \ldots \vee \mathsf{K}L_n \leftarrow L_{l+1} \wedge \ldots \wedge L_m\,, \tag{3.3}$$

$$L_i \leftarrow \lambda_i \quad \text{for } i = 1, \ldots, l\,, \tag{3.4}$$

$$\leftarrow \lambda_i \wedge L_j \quad \text{for } i = 1, \ldots, l \text{ and } j = m+1, \ldots, n\,, \tag{3.5}$$

$$\lambda_i \leftarrow L_i \wedge \lambda_k \quad \text{for } i = 1, \ldots, l \text{ and } k = 1, \ldots, l\,. \tag{3.6}$$

In particular, each integrity constraint containing default negation is transformed into

$$\mathsf{K}L_{m+1} \vee \ldots \vee \mathsf{K}L_n \leftarrow L_{l+1} \wedge \ldots \wedge L_m\,.$$

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 $\mathsf{K}L_j$ means that L_j *is believed*. With this epistemic reading, each default negation $not\, L_j$ in the body of a clause is rewritten in $\neg\mathsf{K}L_j$ 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 (3.1) from P.[5]

An intuitive reading of the transformed clauses is that if $L_{l+1}, \ldots, L_m$ are true, then some L_i $(1 \leq i \leq l)$ becomes true via λ_i when $L_{m+1}, \ldots, L_n$ are not true; otherwise, some L_j $(m+1 \leq j \leq n)$ is believed. The clause (3.6) has an effect to associate λ_i with L_i whenever L_i is true and another disjunct L_k is derived from (3.3) via λ_k.[6] In this way, every extended

[4] The transformation is originally introduced in [19] in a different form.

[5] 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 L_i, where λ_i is an n-ary predicate symbol appearing nowhere in P.

[6] In the case of $l = 1$, the clause (3.6) becomes a tautological clause $\lambda \leftarrow L \wedge \lambda$ and hereafter we will omit such a clause in P^κ.

disjunctive program P is transformed into a positive extended disjunctive program P^κ. Then we can construct the fixpoint of P^κ as presented in the previous section.

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

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

Then the next theorem provides the fixpoint characterization of p-stable models in extended disjunctive programs.

THEOREM 3.6
Let P be an extended disjunctive program and $\mathcal{PST}_P$ be the set of all p-stable models of P. Then,

$$\mathcal{PST}_P = obj_c(min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))).$$

PROOF. Suppose that I is in $obj_c(min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega)))$. Let I^κ be a canonical interpretation in $min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))$ such that $I^\kappa \cap \mathcal{L}_P = I$. Then, for each ground clause of the form (3.1) from P, $\{L_{l+1}, \ldots, L_m\} \subseteq I^\kappa$ implies either (i) $\exists \lambda_i \in I^\kappa$ $(1 \leq i \leq l)$, $L_i \in I^\kappa$, and $\{L_{m+1}, \ldots, L_n\} \cap I^\kappa = \emptyset$, or (ii) $\exists \mathsf{K}L_j \in I^\kappa$ $(m+1 \leq j \leq n)$ by (3.3), (3.4), and (3.5).[7] In case of (i), $\{L_{m+1}, \ldots, L_n\} \cap I^\kappa = \emptyset$ implies $\{L_{m+1}, \ldots, L_n\} \cap I = \emptyset$. Then there is a clause of the form (3.2) in P^I. Since $\{L_{l+1}, \ldots, L_m\} \subseteq I^\kappa$ and $L_i \in I^\kappa$ implies $\{L_{L+1}, \ldots, L_m\} \subseteq I$ and $l_i \in I$, I satisfies the clause (3.2) in P^I. In case (ii), since I^κ is canonical, $\mathsf{K}L_j \in I^\kappa$ implies $L_j \in I^\kappa$, and thus $L_j \in I$. In this case, the clause (3.2) is not included in P^I. Thus, in both cases, I satisfies every clause in P^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 (3.2) such that $\{L_{l+1}, \ldots, L_m\} \subseteq J$ and for some two literals L_{i_1} and L_{i_2} $(1 \leq i_1, i_2 \leq l;\ i_1 \neq i_2)$, $L_{i_1} \in J$ but $L_{i_2} \in I \setminus J$. Without loss of generality, we can assume that just one such clause exists in P^I. Since I does not contain literals $L_{m+1}, \ldots, L_n$, the corresponding canonical interpretation I^κ does not contain $\mathsf{K}L_{m+1}, \ldots, \mathsf{K}L_n$ either. Thus, $\{L_{l+1}, \ldots, L_m\} \subseteq I$ implies $\exists \lambda_k \in I^\kappa$ for some $1 \leq k \leq l$. Since $L_{i_1}, L_{i_2} \in I$ implies $L_{i_1}, L_{i_2} \in I^\kappa$, $\lambda_k \in I^\kappa$ implies $\lambda_{i_1}, \lambda_{i_2} \in I^\kappa$ by (3.6). Let $J^\kappa = I^\kappa \setminus \{L_{i_2}, \lambda_{i_2}\}$. Then the interpretation J^κ satisfies all the clauses (3.3), (3.4), (3.5), (3.6) in P^κ. This contradicts the fact that I^κ is a minimal set satisfying each clause in P^κ. Then I is also a p-minimal model of P^I, hence a p-stable model of P.

Conversely, suppose that I is a p-stable model of P. Then, for each clause C of the form (3.1) from P, let $I_\lambda = \bigcup_C \{\lambda_i \mid \{L_{l+1}, \ldots, L_m\} \subseteq I, \{L_{m+1}, \ldots, L_n\} \cap I = \emptyset$, and $L_i \in I$ $(1 \leq i \leq l)\}$ and $I_K = \bigcup_C \{\mathsf{K}L_j \mid \{L_{l+1}, \ldots, L_m\} \subseteq I$ and $L_j \in I$ $(m+1 \leq j \leq n)\}$. Let $I^{\kappa\prime} = I \cup I_\lambda \cup I_K$. Then, $I^{\kappa\prime}$ satisfies each clause (3.3), (3.4), (3.5) and (3.6) from P^κ, and by the construction of $I^{\kappa\prime}$, $I^{\kappa\prime} \in \mu(\mathcal{T}_{P^\kappa} \uparrow \omega)$. Now let us define $I^\kappa = I \cup S$ where S is a minimal subset of $I_\lambda \cup I_K$ such that each λ_i or $\mathsf{K}L_j$ is chosen in a way that I^κ satisfies every clause in P^κ. Note that for each literal $\mathsf{K}L$ in I^κ, $L \in I$ by definition, so $L \in I^\kappa$. Hence I^κ is canonical. Next assume that there exists $J^\kappa \in \mu(\mathcal{T}_{P^\kappa} \uparrow \omega)$ such that $J^\kappa \subset I^\kappa$. Since we have defined I^κ as a minimal set with respect to the atoms from $I_\lambda \cup I_K$, the inclusion relation implies $J^\kappa \cap \mathcal{L}_P \subset I^\kappa \cap \mathcal{L}_P$. Then $\exists L_i \in I^\kappa \setminus J^\kappa$. In this case, there is a clause (3.3) in P^κ such that $\{L_{l+1}, \ldots, L_m\} \subseteq J^\kappa$, $\lambda_i \in I^\kappa \setminus J^\kappa$, $\mathsf{K}L_j \in J^\kappa$ for some $1 \leq i \leq l$ and $m+1 \leq j \leq n$. Since $J^\kappa \subset I^\kappa$, $\mathsf{K}L_j \in I^\kappa$. As I^κ is canonical, $\mathsf{K}L_j \in I^\kappa$ implies $L_j \in I^\kappa$.

[7] When a clause (3.1) contains no *not*, $\{L_{l+1}, \ldots, L_m\} \subseteq I^\kappa$ implies $L_i \in I^\kappa (1 \leq i \leq l)$ as a special case of (i).

But this is impossible from the condition (3.5). Thus, there is no J^κ which is smaller than I^κ, hence $I^\kappa \in min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))$. Since I^κ is canonical, $I \in obj_c(min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega)))$. ∎

COROLLARY 3.7
Let P be an extended logic program. Then,

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

PROOF. By Theorem 3.6, $obj_c(min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega)))$ is the set of all p-stable models of P. Since $I \in obj_c(min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega)))$ implies $I \in obj_c(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))$, we show that the converse is also true. Assume that the converse does not hold. That is, there is a non-minimal set $I \in obj_c(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))$ and $\exists J \in obj_c(min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega)))$ such that $J \subset I$. In this case, there exists a literal L such that $L \in I \setminus J$. Let $I = I^\kappa \cap \mathcal{L}_P$ and $J = J^\kappa \cap \mathcal{L}_P$ for some canonical interpretations I^κ and J^κ. Then, corresponding to (3.3), (3.4), and (3.5), there exist clauses:

$$\lambda \vee \mathsf{K}L_{m+1} \vee \ldots \vee \mathsf{K}L_n \leftarrow L_{l+1} \wedge \ldots \wedge L_m,$$

$$L \leftarrow \lambda,$$

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

in P^κ, where $\{L_{l+1}, \ldots, L_m\} \subseteq I^\kappa$, $\{L_{l+1}, \ldots, L_m\} \subseteq J^\kappa$, $\lambda \in I^\kappa$, and $\exists \mathsf{K}L_j \in J^\kappa$ $(m+1 \leq j \leq n)$. Note here that the clause (3.6) becomes $\lambda \leftarrow L \wedge \lambda$ and is neglected. Since J^κ is canonical, $L_j \in J^\kappa$. Then $J \subset I$ implies $L_j \in I^\kappa$. But this is impossible from the third clause above. ∎

EXAMPLE 3.8
Let P be the program:

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

Then its epistemic transformation becomes

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

Hence,

$$min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega)) = \{\{\neg a,\ a,\ \neg c,\ \lambda\}, \{b,\ a,\ \neg c,\ \lambda\}, \{\neg a,\ \mathsf{K}c\}, \{b,\ \mathsf{K}c\}\},$$

and thus,

$$obj_c(min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))) = \{\{\neg a,\ a,\ \neg c\}, \{b,\ a,\ \neg c\}\}$$

is the set of p-stable models of P.

For normal disjunctive programs, our fixpoint construction also characterizes the possible model semantics of [31][8] and the *disjunctive stable model semantics* of [27]. Let $\mathcal{PM}_P$ (resp. $\mathcal{ST}_P$) be the set of all possible models (resp. stable models) of P.[9] Since the definition of the p-stable models coincides with that of stable models in normal disjunctive programs, the next results follow from Theorem 3.6, Corollary 3.7, and the result presented in [31].

THEOREM 3.9
Let P be a normal disjunctive program. Then,

[8] It was called the possible world semantics in [31].

[9] Here, stable models mean *total* stable models, that is, any atom in a model has a truth value t or f.

(i) $\mathcal{PM}_P = obj_c(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))$,
(ii) $\mathcal{ST}_P = obj_c(min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega)))$.

In particular, for a normal logic program P, $\mathcal{PM}_P = \mathcal{ST}_P = obj_c(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))$.

The above results are extensions of the results in Theorem 2.17.

3.3 Connection to the answer set semantics

For extended disjunctive programs, Gelfond and Lifschitz have introduced the *answer set semantics* in [15]. The answer sets are defined in the same manner as p-stable models in Definition 3.1 except that the definition of p-minimal models of a positive extended disjunctive program P^I is changed in a way that $I = \mathcal{L}_P$ if a model I contains a pair of complementary literals L and $\neg L$. For instance, in Example 3.2, P has two inconsistent p-stable models, while it has the unique answer set $\mathcal{L}_P$. Thus p-stable models are paraconsistent, while answer sets are not.

In this section, we present a connection between the p-stable model semantics and the answer set semantics in extended disjunctive programs. As presented above, the essential difference between the two semantics is the treatment of inconsistency. Then we relate p-stable models to answer sets by trivializing inconsistent p-stable models.

Let us consider a program P_{tr} obtained from P by incorporating the *trivialization rule*:

$$N \leftarrow L \wedge \neg L \tag{3.7}$$

for all literals L and N from $\mathcal{L}_P$. Then the relationship between answer sets and p-stable models of extended disjunctive programs is as follows.

THEOREM 3.10
Let P be an extended disjunctive program. Then I is an answer set of P iff I is a p-stable model of P_{tr}.

PROOF. Since consistent answer sets coincide with consistent p-stable models, the result follows when I is a consistent answer set. Otherwise, suppose the case that P has the contradictory answer set $\mathcal{L}_P$. Then, by the definition of answer sets, the positive extended disjunctive program $P^{\mathcal{L}_P}$ has the answer set $\mathcal{L}_P$. In this case, $P^{\mathcal{L}_P}$ has no consistent p-minimal model, but an inconsistent p-minimal model. Thus, in the presence of the trivialization rule (3.7), $P_{tr}^{\mathcal{L}_P}$ has the p-minimal model containing every literal N from $\mathcal{L}_P$. Hence, $\mathcal{L}_P$ is the unique p-stable model of P_{tr}. On the other hand, if I is an inconsistent p-stable model of P_{tr}, it contains every literal N from $\mathcal{L}_P$ by (3.7). In this case, $\mathcal{L}_P$ is the p-minimal model of $P_{tr}^{\mathcal{L}_P}$, and thus each p-minimal model of $P^{\mathcal{L}_P}$ contains a pair of complementary literals. Hence, $\mathcal{L}_P$ is the unique answer set of P. ∎

COROLLARY 3.11
Let P be an extended disjunctive program and $\mathcal{AS}_P$ be the set of all answer sets of P. Then,

$$\mathcal{AS}_P = obj_c(min(\mu(\mathcal{T}_{P_{tr}^\kappa} \uparrow \omega))),$$

where P_{tr}^κ is the epistemic transformation of P_{tr}. In particular, when P is an extended logic program, $\mathcal{AS}_P = obj_c(\mu(\mathcal{T}_{P_{tr}^\kappa} \uparrow \omega))$.

The above theorem indicates that we can easily simulate the 'classical' meaning of logic programming by a simple program transformation. Note that without the trivialization rule, there is no one-to-one correspondence between inconsistent p-stable models and the answer set $\mathcal{L}_P$ in general.

EXAMPLE 3.12
The program

$$\{ \neg a \leftarrow, \quad a \leftarrow not\, b \}$$

has no answer set, while it has an inconsistent p-stable model $\{\neg a, a\}$. On the other hand, the program

$$\{ a \leftarrow, \quad \neg a \leftarrow, \quad b \leftarrow not\, b \}$$

has the answer set $\mathcal{L}_P$, while it has no p-stable model.

The example illustrates that a program possibly has an inconsistent p-stable model even when there is no answer set of the program. When a program has no p-stable model, on the other hand, the program has either no answer set or the trivial answer set $\mathcal{L}_P$. Since the answer set semantics brings no useful information, the absence of p-stable models in this case is not a serious drawback. As a result, we can conclude that the paraconsistent stable model semantics is more useful than the answer set semantics.

P-stable models of an extended disjunctive program are also characterized by stable models of the *positive form* of the program. A positive form of an extended disjunctive program P is defined as a normal disjunctive program P^+ which is obtained by replacing each negative literal $\neg L$ in P with a corresponding newly introduced atom L' in P^+. Let I^+ be a model of such P^+. Then the following relation holds by definition.

PROPOSITION 3.13
Let P be an extended disjunctive program and P^+ be its positive form. Then I is a p-stable model of P iff I^+ is a stable model of P^+.

Note that in case of the answer set semantics the above relation holds only for consistent answer sets [15].

4 Reasoning with inconsistency

This section presents applications of paraconsistent semantics for reasoning with inconsistent information.

4.1 Preferred stable models

In the previous section, we have defined the paraconsistent stable model semantics by the collection of all p-stable models. However, when a program has consistent models as well as inconsistent ones, a rational reasoner may prefer consistent models to inconsistent ones and consider truth values only in consistent models.

EXAMPLE 4.1
Recall the program P in Example 3.8:

$$\{ \neg a \vee b \leftarrow, \quad a \leftarrow \neg c, \quad \neg c \leftarrow not\, c \},$$

which has two p-stable models $\{a, \neg a, \neg c\}$ and $\{a, b, \neg c\}$. In this case, however, it seems natural to prefer the consistent model $\{a, b, \neg c\}$ and conclude the truth of a and b.

When an extended disjunctive program has consistent p-stable models, we distinguish these consistent p-stable models as *preferred p-stable models*. Thus preferred p-stable models characterize 'consistent' meaning of a program. In fact, preferred p-stable models coincide with consistent answer sets of extended disjunctive programs.

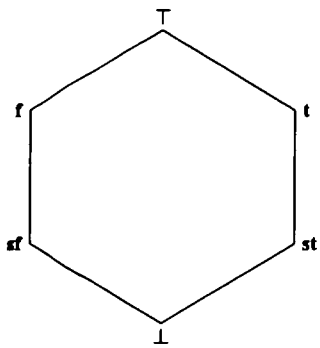

FIG. 2. Six-valued lattice VI

THEOREM 4.2
Let P be an extended disjunctive program. Then I is a preferred p-stable model of P iff I is a consistent answer set of P.

PROOF. Since consistent answer sets coincide with consistent p-stable models, the result follows. ∎

4.2 *Suspicious stable models*

When a program contains inconsistent information, it is useful to distinguish facts affected by such information from others in a program.

EXAMPLE 4.3
Let P be the program:

$$\{\, a \leftarrow b \wedge not\, c, \quad b \leftarrow, \quad \neg b \leftarrow, \quad d \leftarrow \,\}.$$

Then P has the p-stable model $\{a, b, \neg b, d\}$. However, the truth of a is less credible than the truth of d, since a is derived through the contradictory fact b.

In order to distinguish such suspicious facts from others, we present suspicious reasoning under the paraconsistent stable model semantics. To this end, we first introduce two new truth values $\mathbf{st}$ and $\mathbf{sf}$, which, respectively, denote *suspiciously true* and *suspiciously false*. These newly introduced values together with the values in IV constitute a lattice of six-valued logic VI such that $\bot \preceq \mathbf{sx} \preceq \mathbf{x} \preceq \top$ for $\mathbf{x} \in \{\mathbf{t}, \mathbf{f}\}$ (Figure 2).

Let $\mathcal{L}_P^s = \mathcal{L}_P \cup \{L^s \mid L \in \mathcal{L}_P\}$ and I^s be a subset of $\mathcal{L}_P^s$, where each adorned literal L^s denotes a *suspicious literal*. Then an interpretation under the logic VI is defined as a function $I^s : \mathcal{L}_P^s \to VI$ such that for each literal $L \in \mathcal{L}_P$:

$$\begin{aligned} I^s(L) \;=\; lub\{x \mid\; & x = \mathbf{t} \quad \text{if } L \in I^s, \\ & x = \mathbf{f} \quad \text{if } \neg L \in I^s, \\ & x = \mathbf{st} \quad \text{if } L^s \in I^s, \\ & x = \mathbf{sf} \quad \text{if } \neg L^s \in I^s, \\ & x = \bot \quad \text{otherwise}\,\}. \end{aligned}$$

That is, the truth value of each literal $I^s(L)$ is defined as the least upper bound of each value x which is determined by the literal occurrences in I^s. Thus, $I^s(L) = \top$ iff either $L \in I^s$ and $\neg L \in I^s$, or $L^s \in I^s$ and $\neg L^s \in I^s$, or $L \in I^s$ and $\neg L^s \in I^s$, or $\neg L \in I^s$ and $L^s \in I^s$. Note that $I^s(L) = \mathbf{st}$ iff $I^s(\neg L) = \mathbf{sf}$. Under the logic VI, satisfaction of literals and default negation is defined as follows.

- $I^s \models L$ iff $\mathbf{st} \preceq I^s(L)$,
- $I^s \models \neg L$ iff $\mathbf{sf} \preceq I^s(L)$,
- $I^s \models not\ L$ iff $I^s(L) \preceq \mathbf{f}$,
- $I^s \models not \neg L$ iff $I^s(L) \preceq \mathbf{t}$.

Satisfaction of clauses is the same as before.

Next, for a positive extended disjunctive program P and an interpretation I^s, let T_P^s be a mapping which is defined in the same way as in Definition 2.7 except that we consider the mapping T_P^s instead of T_P as follows: if $I^s \models L_1 \wedge \ldots \wedge L_m$ for some ground integrity constraint $\leftarrow L_1 \wedge \ldots \wedge L_m$ from P, then $T_P^s(I^s) = \emptyset$; otherwise,

$$
\begin{aligned}
T_P^s(I^s) \;=\; \{\, J^s \mid\ & \text{for each ground clause } C_i :\ L_1 \vee \ldots \vee L_{l_i} \leftarrow L_{l_i+1} \wedge \ldots \wedge L_{m_i} \\
& \text{from } P \text{ such that } I^s \models L_{l_i+1} \wedge \ldots \wedge L_{m_i}, \\
& J^s = I^s \cup \bigcup_{C_i} \{L_j{}'\}\ \ (1 \le j \le l_i) \quad \text{where} \\
& L_j{}' = L_j \quad \text{if } L_k \in I^s \text{ and } I^s \not\models \neg L_k \text{ for each } L_k\ (l_i + 1 \le k \le m_i); \\
& L_j{}' = L_j^s, \quad \text{otherwise}\ \}.
\end{aligned}
$$

The intuitive meaning of T_P^s is that when the body of a clause C_i is satisfied by I^s, each derived disjunct $L_j{}' = L_j$ is added to I^s if any literal L_k in the body is derived without suspicion and its negative counterpart $\neg L_k$ or $\neg L_k^s$ is not included in I^s. Otherwise, the derived disjunct is suspicious $L_j{}' = L_j^s$, since it is derived through inconsistent information in a program.

Given an extended disjunctive program P and its epistemic transformation P^κ, let us consider the fixpoint closure $\mathcal{SPST}_P = obj_c(min(\mu(\mathcal{T}_{P^\kappa}^s \uparrow \omega)))$. We call $\mathcal{SPST}_P$ the *suspicious p-stable models* of P.

THEOREM 4.4
Let P be an extended disjunctive program. Then a suspicious p-stable model is a model of P.

PROOF. In a suspicious p-stable model, the truth value of each literal possibly becomes $\mathbf{st}$ or $\mathbf{sf}$ when its truth value is, respectively, $\mathbf{t}$ or $\mathbf{f}$ in its corresponding p-stable model. Let I^s be a suspicious p-stable model and I be its corresponding p-stable model in which each literal L^s is identified with L. By definition, $I^s \models L$ iff $I \models L$, and $I^s \models not\ L$ iff $I \models not\ L$. Thus I^s satisfies each clause of P whenever I is a p-stable model of P. Hence the result follows. ∎

COROLLARY 4.5
For each suspicious p-stable model I^s of P, $I = \{L \mid I^s \models L\}$ is a p-stable model of P. Conversely, for each p-stable model J of P, there is a suspicious p-stable model J^s of P such that $J = \{L \mid J^s \models L\}$.

Thus suspicious p-stable models can distinguish information derived through contradictory facts from other information. Clearly, suspicious p-stable models reduce to p-stable models in the absence of suspicious information.

Note that a proved fact is considered to be suspicious if every proof of the fact includes inconsistent information. Then if an interpretation includes both L and L^s, it means that there is a proof of L which depends on no inconsistent information. In this case, by taking the least upper bound of $\mathbf{t}$ and $\mathbf{st}$, the truth value of L becomes $\mathbf{t}$.

EXAMPLE 4.6 (continued from Example 4.3)
P^κ becomes

$$\{\ \lambda \vee \mathsf{K}c \leftarrow b,\quad a \leftarrow \lambda,\quad \leftarrow \lambda \wedge c,\quad b \leftarrow,\quad \neg b \leftarrow,\quad d \leftarrow\ \},$$

and

$$obj_c(min(\mu(\mathcal{T}^s_{P^\kappa} \uparrow \omega))) = \{\{a^s,\ b,\ \neg b,\ d\}\}.$$

Then, $I^s(a) = \mathbf{st}$, $I^s(b) = \top$, $I^s(c) = \bot$, and $I^s(d) = \mathbf{t}$.

4.3 *Semi-stable models*

There is an extended disjunctive program which has no p-stable model but still contains useful information. For instance, in Example 3.3, P has no p-stable model but it seems reasonable to conclude the truth of b. Roughly speaking, incoherency arises when a literal is implied by its default negation in a program. Since incoherency is viewed as a kind of inconsistency, it is desirable to provide a framework which is paraconsistent for such incoherency. In this section, we introduce the notion of semi-stable models which is paraconsistent for incoherent programs.

To present incoherent facts, we first introduce five extra truth values $\mathbf{bt}$, $\mathbf{bf}$, $\mathbf{b}\top$, $\mathbf{tcb}$, and $\mathbf{fcb}$ which, respectively, denote *believed true*, *believed false*, *believed contradictory*, *true with contradictory belief* and *false with contradictory belief*. These values together with the values in IV constitute a lattice of *nine-valued logic* IX such that $\bot \preceq \mathbf{bx} \preceq \mathbf{x} \preceq \mathbf{xcb} \preceq \top$ and $\mathbf{bx} \preceq \mathbf{b}\top \preceq \mathbf{xcb}$ for $\mathbf{x} \in \{\mathbf{t}, \mathbf{f}\}$ (Figure 3).

Let $\mathcal{L}^\kappa_P = \mathcal{L}_P \cup \{\mathsf{K}L \mid L \in \mathcal{L}_P\}$ and I^κ be a subset of $\mathcal{L}^\kappa_P$. Then, an interpretation under the logic IX is defined as a function $I^\kappa : \mathcal{L}^\kappa_P \to IX$ such that for each literal $L \in \mathcal{L}_P$:

$$\begin{aligned} I^\kappa(L) \quad = \quad lub\{x \mid\ & x = \mathbf{t} \quad \text{if } L \in I^\kappa, \\ & x = \mathbf{f} \quad \text{if } \neg L \in I^\kappa, \\ & x = \mathbf{bt} \quad \text{if } \mathsf{K}L \in I^\kappa, \\ & x = \mathbf{bf} \quad \text{if } \mathsf{K}\neg L \in I^\kappa, \\ & x = \bot \quad \text{otherwise}\ \}. \end{aligned}$$

Thus, $I^\kappa(L) = \mathbf{b}\top$ iff both $\mathsf{K}L \in I^\kappa$ and $\mathsf{K}\neg L \in I^\kappa$; $I^\kappa(L) = \mathbf{fcb}$ iff both $\mathsf{K}L \in I^\kappa$ and $\neg L \in I^\kappa$; $I^\kappa(L) = \mathbf{tcb}$ iff both $\mathsf{K}\neg L \in I^\kappa$ and $L \in I^\kappa$, and so on. Note that $I^\kappa(L) = \mathbf{bt}$ iff $I^\kappa(\neg L) = \mathbf{bf}$, $I^\kappa(L) = \mathbf{b}\top$ iff $I^\kappa(\neg L) = \mathbf{b}\top$, and $I^\kappa(L) = \mathbf{tcb}$ iff $I^\kappa(\neg L) = \mathbf{fcb}$.

The intuitive reading of each newly introduced truth value is that if $I^\kappa(L) = \mathbf{bt}$, I^κ contains a belief $\mathsf{K}L$ without its justification L. On the other hand, if $I^\kappa(L) = \mathbf{tcb}$, I^κ contains a fact L with its opposite belief $\mathsf{K}\neg L$.

Under this logic, satisfaction of literals and default negation is defined in the same way as Section 3, i.e. $I \models L$ iff $\mathbf{t} \preceq I(L)$; $I \models \neg L$ iff $\mathbf{f} \preceq I(L)$; $I \models not\ L$ iff $I(L) \preceq \mathbf{f}$; and $I \models not \neg L$ iff $I(L) \preceq \mathbf{t}$. Satisfaction of clauses is also defined as before.

According to the above definition, when $I(L) = \mathbf{bt}$ or $I(L) = \mathbf{b}\top$, it holds that $I \not\models L$ and $I \not\models not\ L$. This means when L is believed true, it is too weak to conclude the truth of L, but

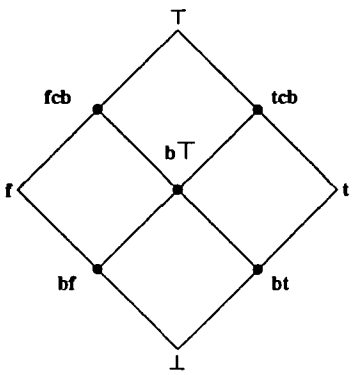

FIG. 3. Nine-valued lattice IX

enough to reject *not* L.[10] Else when $I(L) = \mathbf{tcb}$, $I \models L$ while $I \not\models not \neg L$. This means when L is true with contradictory belief, I concludes the truth of L but rejects *not* $\neg L$ in the presence of its opposite belief $\mathsf{K}\neg L$.

Next let $\mathcal{I}_{P^\kappa}$ be a set of interpretations of a program P^κ obtained by the epistemic transformation of an extended disjunctive program P. Then an interpretation $I^\kappa \in \mathcal{I}_{P^\kappa}$ is said *maximally canonical* if there is no interpretation $J^\kappa \in \mathcal{I}_{P^\kappa}$ such that $\{\mathsf{K}L \mid \mathsf{K}L \in J^\kappa \text{ and } L \notin J^\kappa\} \subset \{\mathsf{K}L \mid \mathsf{K}L \in I^\kappa \text{ and } L \notin I^\kappa\}$. That is, a maximally canonical interpretation is an interpretation such that the canonical condition is satisfied as much as possible. In particular, if $\mathcal{I}_{P^\kappa}$ contains an interpretation I^κ which is canonical, it is also maximally canonical. Now let

$$obj^\kappa_{mc}(\mathcal{I}_{P^\kappa}) = \{I^\kappa \cap \mathcal{L}^\kappa_P \mid I^\kappa \in \mathcal{I}_{P^\kappa} \text{ and } I^\kappa \text{ is maximally canonical }\}.$$

THEOREM 4.7
Let P be an extended disjunctive program. Then, any interpretation included in $\mathcal{SST}_P = obj^\kappa_{mc}(min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega)))$ is a model of P.

PROOF. By definition, each maximally canonical interpretation I^κ included in $min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))$ is a model of P^κ. Then, for each transformed clauses (3.3) and (3.4), $\{L_{l+1}, \ldots, L_m\} \subseteq I^\kappa$ implies either $L_i \in I^\kappa$ $(1 \leq i \leq l)$ or $\mathsf{K}L_j \in I^\kappa$ $(m+1 \leq j \leq n)$. In the case $L_i \in I^\kappa$, I^κ satisfies the corresponding clause (3.1) in P. In the case $\mathsf{K}L_j \in I^\kappa$, when $L_j \in I^\kappa$, I^κ satisfies the clause (3.1) in P. Else when $L_j \notin I^\kappa$, (i) if $\neg L_j \notin I^\kappa$, the truth value of L_j is $\mathbf{bt}$ or $\mathbf{b}\top$, then $I^\kappa \not\models not\ L_j$. (ii) Else if $\neg L_j \in I^\kappa$, the truth value of L_j becomes $\mathbf{fcb}$, then $I^\kappa \not\models not\ L_j$. In either case, I^κ satisfies the clause (3.1) in P. Therefore, I^κ satisfies each clause in P. Hence, $I^\kappa \cap \mathcal{L}^\kappa_P$, which is obtained from I^κ by removing every λ_i, is also a model of P. ∎

We call models $\mathcal{SST}_P$ the *semi-stable models* of P.
The notion of semi-stable models reduces to p-stable models in coherent programs.

COROLLARY 4.8
Let P be a coherent program. Then its semi-stable models coincide with the p-stable models.

PROOF. When $min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))$ contains canonical interpretations, they are also maximally canonical. Hence the result follows by definition. ∎

[10] Recall that *not* L corresponds to $\neg\mathsf{K}L$.

The existence of semi-stable models is guaranteed for any program which has models.

THEOREM 4.9
When a program has a model, it has a semi-stable model.

PROOF. When a program P has a model, it is easy to see that P^κ also has a model. Then the closure $min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))$ contains models which are maximally canonical, hence $\mathcal{SST}_P$ is not empty. ∎

Thus incoherent programs obtain their meaning by considering semi-stable models.

EXAMPLE 4.10 (Barbar's Paradox)
Let P be the program:

$$\{\ shave(Noel, x) \leftarrow not\, shave(x, x), \quad mayor(Casanova) \leftarrow\ \}.$$

Then its epistemic transformation P^κ becomes

$$\begin{array}{l} \{\ \lambda(x) \vee \mathsf{K}shave(x, x) \leftarrow, \\ shave(Noel, x) \leftarrow \lambda(x), \\ \leftarrow \lambda(x) \wedge shave(x, x), \\ mayor(Casanova) \leftarrow\ \}. \end{array}$$

Thus,

$$\begin{array}{rcl} min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega)) & = & \{\{\mathsf{K}shave(N, N),\ \mathsf{K}shave(C, C),\ mayor(C)\}, \\ & & \{\mathsf{K}shave(N, N),\ \lambda(C),\ shave(N, C),\ mayor(C)\}\}. \end{array}$$

In the above closure, the second interpretation is maximally canonical, hence

$$obj^\kappa_{mc}(min(\mu(\mathcal{T}_{P^\kappa} \uparrow \omega))) = \{\{\ \mathsf{K}shave(N, N),\ shave(N, C),\ mayor(C)\ \}\},$$

which contains the unique semi-stable model of P such that $shave(N, C)$ and $mayor(C)$ are true, while $shave(N, N)$ is believed true.

Note that the above program has neither standard two-valued stable models nor answer sets.

In the incoherent program $\{\ a \leftarrow not\, a\ \}$, it is known that interpretations 'oscillate' between $\emptyset$ and $\{a\}$ under the *stable class semantics* [3]. Then it is interesting to observe that the truth value $I(a) = \mathbf{bt}$ in its semi-stable model correspondingly lies between $\bot$ and $\mathbf{t}$.

5 Related work

A framework of paraconsistent logic programming is firstly developed by Blair and Subrahmanian [5] in the context of annotated logic programs. They employ Belnap's four-valued logic as a theoretical basis, but their framework does not treat default negation in a program. Fitting [12] provides a general framework for logic programming in terms of bilattices, but he does not discuss programs containing two kinds of negation. Kifer and Lozinskii [21] extend Blair and Subrahmanian's annotated logic programming framework to a theory possibly containing default negation, and Wagner [35] also develops a theory of inconsistent logic programs with two kinds of negation. Compared with our approach, they do not treat disjunctions in a program and the underlying logics presented by them are different from our stable model semantics.

Subrahmanian [32] has extended the framework of [5] to programs containing disjunctive information. However, he does not treat default negation in a program. He also provides a fixpoint semantics of paraconsistent disjunctive programs based on Minker and Rajasekar's model state fixpoint semantics [25]. By contrast, our fixpoint semantics is based on the manipulation of Herbrand interpretations and directly computes the paraconsistent minimal/stable models. Lu and Henschen [23] consider specifying the closed world assumption in paraconsistent definite and disjunctive logic programs. However, they neither consider programs containing two kinds of negation nor develop any fixpoint theory for disjunctive programs. A fixpoint semantics of disjunctive programs is also developed by Fernandez *et al.* [10]. Their approach is close to ours but different in the following points. First, their fixpoint closure computes stable models of normal disjunctive programs, while ours computes p-stable models as well as answer sets of extended disjunctive programs. Second, our fixpoint closure computes not only stable models but also possible models of disjunctive programs. Inoue and Sakama [20] also present yet another fixpoint semantics which characterizes the answer set semantics of extended disjunctive programs, while they do not treat paraconsistent semantics nor the possible model semantics.

Paraconsistent stable model semantics is also proposed by several researchers. Pimentel and Rodi [28], Grant and Subrahmanian [17], and Wagner [36] study paraconsistent stable model semantics from different viewpoints. The differences between these approaches and ours are as follows. First, their paraconsistent stable model semantics are defined for extended logic programs and do not treat disjunctive information in a program. Second, they do not provide any mechanism to compute their stable models, while our fixpoint computation realizes constructive computation of paraconsistent stable models. Third, we have introduced the notion of semi-stable models which are paraconsistent for incoherent programs, while they do not discuss the issue of handling incoherency. Recently, Fitting [13] provided a framework of stable model semantics in terms of bilattices, but it did not treat disjunctive programs. Sakama [30] has also developed a paraconsistent well-founded semantics for extended logic programs and disjunctive programs, which is different from the stable model approach presented in this paper.

The paraconsistent semantics presented in this paper is intended to localize inconsistent information in a program. There is an alternative approach which tries to detect the source of inconsistency and recover the consistency of a program. Generally speaking, however, it is a hard task to automatically resolve inconsistency in a program. When inconsistency arises from default assumptions, Pereira *et al.* [26] and Dung *et al.* [8] present methods of removing the inconsistency by preferring a fact that does not depend on any default assumption. However, their approaches are of no use in a program where inconsistency is derived without default assumptions. Kowalski and Sadri [22] resolve contradiction by giving a higher priority to one of the conflicting conclusions as an exception, but such an approach generally requires one to specify a preference for each individual rule. Inoue [18] and Baral *et al.* [2] consider the meaning of an inconsistent program as a collection of maximally consistent subsets of the program, but such a collection grows exponentially according to the increase of inconsistent information.

6 Conclusion

This paper has presented declarative semantics of extended disjunctive programs. We have introduced the paraconsistent minimal and stable model semantics for extended disjunctive programs based on lattice-structured multi-valued logics. The paraconsistent semantics are characterized by a new fixpoint semantics of extended disjunctive programs. We have also discussed applications of the paraconsistent semantics for reasoning with inconsistency.

The paraconsistent minimal/stable model semantics are natural extensions of the usual minimal/stable model semantics for disjunctive programs, and compared with Gelfond and Lifschitz's answer set semantics, the proposed semantics do not trivialize a program in the presence of inconsistent information. The paraconsistent semantics presented in this paper generalize previous studies of paraconsistent logic programming and provide a uniform framework of logic programming possibly containing inconsistent information, disjunctive information, integrity constraints, and both explicit and default negation in a program. The fixpoint semantics presented in this paper characterizes operational aspects of such programs and is also implemented using bottom-up model generation techniques as presented in [19].

References

[1] A. I. Arruda. A survey of paraconsistent logic. In *Mathematics Logic in Latin America*, A. I. Arruda, R. Chuaqui and N. C. A. da Costa, eds, pp. 1–41. North-Holland, 1980.

[2] C. Baral, S. Kraus, J. Minker and V. S. Subrahmanian. Combining knowledge bases consisting of first-order theories. *Computational Intelligence*, **8**, 45–71, 1992.

[3] C. Baral and V. S. Subrahmanian. Stable and extension class theory for logic programs and default logics. *Journal of Automated Reasoning*, **8**, 345–366, 1992.

[4] N. D. Belnap. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logic*, J. M. Dunn and G. Epstein, eds, pp. 8–37. Reidel Publishing, 1975.

[5] H. A. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computer Science*, **68**, 135–154, 1989.

[6] E. P. F. Chan. A possible world semantics for disjunctive databases. *IEEE Transactions on Knowledge and Data Engineering*, **5**, 282–292, 1993.

[7] N. C. A. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, **15**, 497–510, 1974.

[8] P. M. Dung and P. Ruamviboonsuk. Well-founded reasoning with classical negation. In *Proceedings of the First International Workshop on Logic Programming and Nonmonotonic Reasoning*, pp. 120–132. MIT Press, 1991.

[9] F. Fages. A new fixpoint semantics for general logic programs compared with the well-founded and the stable model semantics. *New Generation Computing*, **9**, 425–443, 1991.

[10] J. A. Fernandez, J. Lobo, J. Minker and V. S. Subrahmanian. Disjunctive LP + integrity constraints = stable model semantics. *Annals of Mathematics and Artificial Intelligence*, **8**, 449–474, 1993.

[11] M. Fitting. A Kripke–Kleene semantics for logic programming. *Journal of Logic Programming*, **4**, 295–312, 1985.

[12] M. Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming*, **11**, 91–116, 1991.

[13] M. Fitting. The family of stable models. *Journal of Logic Programming*, **17**, 197–225, 1993.

[14] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth Joint International Conference and Symposium on Logic Programming*, pp. 1070–1080. MIT Press, 1988.

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

[16] M. L. Ginsberg. Multivalued logics. *Computational Intelligence*, **4**, 265–316, 1988.

[17] J. Grant and V. S. Subrahmanian. Reasoning in inconsistent knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, in press.

[18] K. Inoue. Extended logic programs with default assumptions. In *Proceedings of the Eighth International Conference on Logic Programming*, pp. 490–504. MIT Press, 1991.

[19] K. Inoue, M. Koshimura and R. Hasegawa. Embedding negation as failure into a model generation theorem prover, In *Lecture Notes in Artificial Intelligence 607, Proceedings of the Eleventh International Conference on Automated Deduction*, pp. 400–415. Springer-Verlag, 1992.

[20] K. Inoue and C. Sakama. Transforming abductive logic programs to disjunctive programs. In *Proceedings of the Tenth International Conference on Logic Programming*, pp. 335–353. MIT Press, 1993.

[21] M. Kifer and E. L. Lozinskii. A logic for reasoning with inconsistency. *Journal of Automated Reasoning*, **9**, 179–215, 1992.

[22] R. A. Kowalski and F. Sadri. Logic programs with exception. In *Proceedings of the Seventh International Conference on Logic Programming*, pp. 598–613. MIT Press, 1990.

[23] J. J. Lu and L. J. Henschen. The closed world assumption in paraconsistent deductive databases. Draft manuscript, Northwestern University, 1992.

[24] J. Minker. On indefinite data bases and the closed world assumption. In *Lecture Notes in Computer Science 138, Proceedings of the Sixth International Conference on Automated Deduction*, pp. 292–308. Springer-Verlag, 1982.

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

[26] L. M. Pereira, J. J. Alferes and N. Aparicio. Contradiction removal within well-founded semantics. In *Proceedings of the First International Workshop on Logic Programming and Nonmonotonic Reasoning*, pp. 105–119. MIT Press, 1991.

[27] T. C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, **9**, 401–424, 1991.

[28] S. G. Pimentel and W. L. Rodi. Belief revision and paraconsistency in a logic programming framework. In *Proceedings of the First International Workshop on Logic Programming and Nonmonotonic Reasoning*, pp. 228–242. MIT Press, 1991.

[29] C. Sakama. Possible model semantics for disjunctive databases. In *Proceedings of the First International Conference on Deductive and Object-Oriented Databases*, pp. 369–383. North-Holland, 1989.

[30] C. Sakama. Extended well-founded semantics for paraconsistent logic programs. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pp. 592–599. ICOT, 1992.

[31] C. Sakama and K. Inoue. Negation in disjunctive logic programs. In *Proceedings of the Tenth International Conference on Logic Programming*, pp. 703–719. MIT Press, 1993.

[32] V. S. Subrahmanian. Paraconsistent disjunctive deductive databases. *Theoretical Computer Science*, **93**, 115–141, 1992.

[33] F. Teusink. A characterization of stable models using a non-monotonic operator. In *Proceedings of the Second International Workshop on Logic Programming and Nonmonotonic Reasoning*, pp. 206–222. MIT Press, 1993.

[34] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, **23**, 733–742, 1976.

[35] G. Wagner. A database needs two kinds of negation. In *Lecture Notes in Computer Science 495, Proceedings of the Third Symposium on Mathematical Fundamentals of Database and Knowledge Base Systems*, pp. 357–371. Springer-Verlag, 1991.

[36] G. Wagner. Reasoning with inconsistency in extended deductive databases. In *Proceedings of the Second International Workshop on Logic Programming and Nonmonotonic Reasoning*, pp. 300–315. MIT Press, 1993.

Received 5 November 1993