プログラミングー言語・基礎・実践ー 19－4
（1994．11．4）

矛盾による否定と二重否定問題

山崎 進* 足立 亘**

* 岡山大学工学部
** NTT 中国支社

本稿では、失敗による否定を包含した、矛盾による否定を実現するデータベースを扱う。二重否定問題を考察し、二重否定の規則が成立するデータベースの１つのクラスを与える。一般論理プログラムを正の情報とし、その安定モデルに属さないアトムの集合を負の情報とするデータベースにおいては、二重否定の規則が成立することを示す。このデータベースのクラスは、Gabbay らが失敗による否定との関係を論じたデータベースのクラスと異なる。

Negation as Inconsistency and Double Negation Problem

Susumu Yamasaki * and Wataru Adachi **

* Faculty of Engineering, Okayama University
** NTT, Chugoku Branch

In this paper, we deal with Gabbay et al's database, which is proposed to realize negation as inconsistency, containing negation as failure, from the point of double negation problem. We give a class of databases in which the double negation law may hold. The database in the class consist of a general logic program as a positive declaration, and of a set, as a negative declaration, each of which is not a member of a stable model of the general logic program. It is shown the class is different from the class of databases in which the negation as inconsistency is regarded as negation as failure

1 Introduction

Since Reiter proposed the closed world assumption[11], the extended or generalized closed world assumption has been considered [4, 7]. In logic programming, negation as failure (NAF, for short) has been combined with SLD resolutions, and examined in refined details [6, 12, 13, 14, 15]. On the one hand, NAI may be made use of in abduction frameworks. The model theory for general logic programs is developed [3, 13] and made clear in relation with default theory and abduction applied to the general logic programs [8].

In [2], negation as inconsistence (NAI, for short) is presented to be realized in a database so that it might involve NAF. The database for NAI consists of a logic program and a set of goals. Given a database with a query as a goal, it is checked whether the query is successful or finitely fails for the logic program by means of resolution deductions. If some in the given set of goals is successful as a query at some point in deductions, then the database is regarded as inconsistent, and the negation of the original goal is interpreted as holding. That is, negation as inconsistency is supported. In the database, the goal, which is defined recursively by negations and logical conjunctions, may be a query. That is, the goal takes the form $\neg(a \wedge \neg b)$. a is added to the logic program, whereas b is given to the set of goals. Such dynamical extensions makes the database transformed.

In such a database, even if a double negative of a goal is successful as a query in the database, the goal may not be successful. We see later in the database that the double negative of a goal is successful whenever the goal is successful. We say the double negation law holds if the goal is successful whenever its double negative is successful. In general, the double negation law does not necessarily hold in the database. We consider a class of the databases in which the double negation law holds. It may be effective to apply the double negation law to the database if it holds, when we are concerned with the deductions for a query. This is a motive to consider the law. In addition, we are motivated by the interest in what law in classical logic holds in such a database. Since the database is closely related with negation as inconsistency, the double negation law should be investigated. Gabbay et al. dealt with general logic programs in their presentations of databases, whereas more sophisticated logic programs, like disjunctive logic programs, have not yet been treated. We think the treatment of negation in general logic programs is still significant, because it is fundamental. Therefore we assume that the database involves the general logic program, the model of which we pay attention to.

In model theory from 2 valued logic approach, the stable model [3] is basic, whereas the perfect model [9, 10] is concerned with priorities of predicate symbols included in a given general logic program. Although a stable model is a fixpoint of a nonmonotonic function and there is no constructive way to get it, it is simple and general when we have got it by some estimation. Therefore, we take the stable model to study the database in which the double negation law holds.

We show a negation of goal is the logical consequence of the union of a given general logic program and the set of negated atoms, each of which is not included in a stable model of the general logic program, iff the goal is successful in deductions for the database consisting of the general logic program and the set of atoms, each of which is not included in the stable model. By the above proposition, we present the database which consists of a general logic program and the set of atoms, each of which is not included in a stable model of the general logic program, and show in it the double negation law holds. To make the discussion simpler, we treat with the database in propositional logic. The extension of the database to that in first-order logic will be just mentioned with a brief reason.

2 NAI in Propositional Logic

We have the following terminologies regarding the database Gabbay et al. proposed, and abbreviated by GDB in this paper.

A general logic program is a set of clauses of the form $p \leftarrow l_1 \ldots l_n$, which is said to be a program clause, where p is an atom, and $l_1, \ldots, l_n$ are literals. A literal l is an atom p or the negation of an atom p, that is, $\neg p$, where $\neg$ stands for the classical negation.

An atom is an expression of the form $P(t_1, \ldots, t_n)$ $(n \geq 0)$, where P is a predicate symbol, $t_1, \ldots, t_n$ are terms. The term is recursively defined: (i) A variable is a term. (ii) $f(t_1, \ldots, t_n)$ $(n \geq 0)$ is a term if f is a function symbol, and $t_1, \ldots, t_n$ are terms.

Each program clause $p \leftarrow l_1 \ldots l_n$ is interpreted as a first-order formula

$$\forall x_1 \ldots x_k(\neg l_1 \vee \ldots \vee \neg l_n \vee p),$$

where $x_1, \ldots, x_k$ are variables occurring in the program clause. $l_1 \ldots l_n$ is said its body, and p its head.

In this paper, we restrict our treatments with formulas to propositional. Now we examine the problem of double negation law. To do it, we take the stable model into account, and present a theorem on the double negation law in the database, by applying the stable model theory to the construction of it.

2.1 GDB in Propositional Logic

We define several definitions concerning GDB following [2]. The clausal form mentioned in the following definition will be referred to as just a clause.

Definition 2.1
(1) An atom q is a clause, and a goal as well. A literal l is a clause, and a goal as well.
(2) A conjunction of goals is a goal, and a negation of a goal a goal.
(3) If A is a goal and q an atom, $q \leftarrow A$ is a clause.

Definition 2.2
A database is an expression of the form (P, N), where P is a set of clauses and N a set of goals.

The intuitive meaning of N is as follows: N is regarded as $\bigwedge_{n\in N} n$, and $\bigwedge_{n\in N} \neg n$ as added to P. (P,N) is interpreted as inconsistent if some $n \in N$ succeeds from the database (P,N). The notion $(P,N)(?I)G = 1$ follows [2]. That is, it means the goal G succeeds from a database (P,N) as follows. Below the goal G means a query $?G$ standing for $\neg G$.

Definition 2.3 (a) For G atomic, $(P,N)(?I)G = 1$, iff either

(i) $G \in P$ or

(ii) for some clause $(G \leftarrow A) \in P$, $(P,N)(?I)A = 1$.

The goal G is said to fail finitely, and denoted $(P,N)(?I) = 0$ if $G \notin P$ and for every $(G \leftarrow A) \in P$, $(P,N)(?I) = 0$.

(b) $(P,N)(?I)G_1 \wedge \ldots \wedge G_n = 1$ iff for each i, $(P,N)(?I)$ $G_i = 1$.

The goal $G_1 \wedge \ldots \wedge G_n$ is said to fail finitely if for some i, $(P,N)(?I)G_i$ fails finitely. We denote it by $(P,N)(?I)$ $G_1 \wedge \ldots \wedge G_n = 0$.

(c) $(P,N)(?I)\neg(\bigwedge_i q_i \wedge \bigwedge_j \neg B_j) = 1$ iff, for some $A \in N \cup \bigcup_j\{B_j\}$, $(P \cup \bigcup_i\{q_i\}, N \cup \bigcup_j\{B_j\})(?I)A = 1$.

The original goal is said to fail finitely and denoted $(P,N)(?I)\neg(\bigwedge_i q_i \wedge \bigwedge_j \neg B_j) = 1$ if the present goal fails finitely. We denote it by $(P,N)(?I)$ $\neg(\bigcup_i q_i \wedge \bigwedge_j \neg B_j) = 0$.

The inconsistency or consistency of a database (P,N) is defined as follows.

Definition 2.4

(1) (P,N) is said inconsistent if some goal $n \in N$ succeeds from the database (P,N), that is, $(P,N)(?I)$ $n = 1$ for some $n \in N$.

(2) (P,N) is said consistent if any goal $n \in N$ does not succeed from the database (P,N), that is, $(P,N)(?I)$ $n \neq 1$ for any $n \in N$.

The success of a goal $\neg q$ is defined by the inconsictency: $\neg q$ succeeds from a database (P,N) if the extended database $(P \cup \{q\}, N)$ is inconsistent. Also the success of a goal $\neg\neg G$ is defined by the inconsistency. That is, $\neg\neg G$ succeeds from a database (P,N) if the extended database $(P, N \cup \{G\})$ is inconsistent.

Example 2.5 Let p, q, r be propositions.

(1) We define (P,N) to be as follows, and assume a goal r, where the goal r means $?r$ standing for $\neg r$.

P	N	G
$r \leftarrow p,$	$\neg p \wedge \neg q,$	$?\ r$
$r \leftarrow q.$		

The goal r does not succeed, because the (sub-)goal p nor q does not succeed from (P,N). Note N does not contribute to the success of r.

(2) Next consider the goal $\neg\neg r$.

P	N	G
$r \leftarrow p,$	$\neg p \wedge \neg q,$	$?\neg\neg r$
$r \leftarrow q.$		

The database is transformed according to the definition.

P_1	N_1	G_1
$r \leftarrow p,$	$\neg p \wedge \neg q,$	$?r$ (or $?\neg p \wedge \neg q$)
$r \leftarrow q.$	r	

The candidate for a successful goal is r, or $\neg p \wedge \neg q$. Because a goal in N_1 should be examined. If no goal is successful, the goal $\neg\neg r$ is not successful.

(a) The goal r finitely fails. It is due to the same reason for **(1)**.

(b) The goal $\neg p \wedge \neg q$ is checked. We must examine whether both the goal $\neg p$ and the goal $\neg q$ are successful.

P_2	N_2	G_2
$r \leftarrow p,$	$\neg p \wedge \neg q,$	$?\neg p$ $(?\neg q)$
$r \leftarrow q.$	r	

The database (P_2, N_2) is transformed to:

P_3	N_3	G_3
$r \leftarrow p,$	$\neg p \wedge \neg q,$	$?r$ (or $?\neg p \wedge \neg q$)
$r \leftarrow q.$	r	
p		

P_3'	N_3'	G_3'
$r \leftarrow p,$	$\neg p \wedge \neg q,$	$?r$ (or $?\neg p \wedge \neg q$)
$r \leftarrow q.$	r	
q		

The goal r succeeds from the database (P_3, N_3), which is inconsistent. Similarly (P_3', N_3') is inconsistent. Therefore the goal $\neg\neg r$ succeeds from (P,N). Note that $r \in N_1$, since the goal is $\neg\neg r$. Hence $(P,N)(?I)\neg\neg r = 1$. However, $(P,N)(?I)r \neq 1$. As illustrated here, $(P,N)(?I)$ $\neg\neg r = 1$ does not necessarily imply $(P,N)(?I)r = 1$. We will see that $(P,N)(?I)r = 1$ implies $(P,N)(?I)$ $\neg\neg r = 1$.

2.2 GDB and NAI

The following relationship between NAF and NAI means that NAF may be realized by NAI, that is, NAI is more general than NAF.

Theorem 2.6 [2] *Let P be a general logic program, and G a goal of the form $\bigwedge_i a_i \wedge \bigwedge_j \neg b_j$, where a_i, b_j are atoms. Also let*

$$N = \{m \mid m \text{ is an atom, and } P(?F)m = 0\},$$

where $P(?F)m = x$ means the goal G finitely fails for P if $x = 0$, and G succeeds if $x = 1$, by means of SLD resolution and NAF. (Note the goal G stands for $\leftarrow G$ when applied to SLD resolution.) Then

$$P(?F)G = 1 \Leftrightarrow (P,N)(?I)G = 1.$$

The following lemma is exploited for the proof of a theorem regarding the double negation law.

Lemma 2.7 *Assume databases* (P,N) *and* (P',N'). *Then*

$$\forall P,N,P',N',G:$$
$$[P \subset P' \& N \subset N' \&$$
$$(P,N)(?I)G = 1 \Rightarrow (P',N')(?I)G = 1].$$

Proof We define a predicate Q to be

$$Q(P,N,P',N',G)$$
$$\Leftrightarrow \quad P \subset P' \& N \subset N' \&$$
$$[(P,N)(?I)G = 1 \Rightarrow (P',N')(?I)G = 1].$$

We show by induction on the reverse direction of rewritings of databases and goals.

(1) Assume $(P,N)(?I)G = 1$ for atomic $G \in P$. As long as $P \subset P'$ and $N \subset N'$, $G \in P'$. Hence $(P',N')(?I)G = 1$. That is, if there is no rewriting of the database and the goal (if there is the empty rewriting), $Q(P,N,P',N',G)$.

(2) (a) Assume $(P,N)(?I)\ G = 1$ on condition $G \leftarrow A \in P$ and $(P,N)(?I)A = 1$. On the assumption of

$$Q(P,N,P',N',A),$$

$(P',N')(?I)A = 1$ if $P \subset P'$ and $N \subset N'$. As $G \leftarrow A \in P'$, $(P',N')(?I)G = 1$. Therefore $Q(P,N,P',N',A)$ implies $Q(P,N,P',N',G)$. That is, in this case of rewriting of the goal, the predicate Q is preserved.

(b) Assume $(P,N)(?I) = 1$ on condition that $G = G_1 \wedge \ldots \wedge G_n$ and $(P,N)(?I)G_i = 1$ for each i. Now assume $Q(P,N,P',N',G_i)$ for each i. If $P \subset P'$ and $N \subset N'$, then $(P',N')(?I)\ G_i = 1$ for each i. Hence $(P,N)(?I)G = 1$. That is, $Q(P,N,P',N',G_i)$ for each i implies $Q(P,N,P',N',G)$. Therefore, for the rewriting of the goal in this case, the predicate Q is preserved.

(c) Assume $(P,N)(?I)\ G = 1$ on condition the goal G takes the form $\neg(\bigwedge_i q_i \wedge \bigwedge_j \neg B_j)$ and

$$(P \cup \bigcup_i \{q_i\}, N \cup \bigcup_j \{B_j\})(?I)A = 1$$

for some $A \in N \cup \bigcup_j \{B_j\}$. Now assume $Q(P,N,P',N',A)$. If $P \subset P'$ and $N \subset N'$, then

$$(P' \cup \bigcup_i \{q_i\}, N' \cup \bigcup_j \{B_j\})(?I)A = 1$$

for some $A \in N \cup \bigcup_j \{B_j\} \subset N' \cup \bigcup_j \{B_j\}$. Hence $(P',N')(?I)\ G = 1$. That is, if

$$Q(P \cup \bigcup_i \{q_i\}, N \cup \bigcup_j \{B_j\}, P' \cup \bigcup_i \{q_i\}, N' \cup \bigcup_j \{B_j\}, A),$$

then $Q(P,N,P',N',G)$. Therefore, for the rewriting of the database and the goal, the predicate Q is preserved.

If $(P,N)(?I)G = 1$, then finite number of rewritings of the database and the goal are made. In the reverse direction of rewritings, the predicate Q is preserved. Taking (1) into considerations,

$$\forall P,N,P',N',G:\ Q(P,N,P',N',G).$$

This completes the proof. Q.E.D.

Lemma 2.8 *If* $(P,N)(?I)G = 1$, *then* $(P',N')(?I)\neg\neg G = 1$.

Proof If $(P,N)(?I)G = 1$, then by Lemma 2.7 $(P,N \cup \{G\})(?I)G = 1$. Because $G \in N \cup \{G\}$, $(P,N)(?I)\neg\neg G = 1$. Q.E.D.

3 Double Negation Problem in GDB

3.1 General Logic Program and Stable Model

Concerning GDB, as we have seen in Lemma 2.8, $(P,N)(?I)G = 1$ implies $(P,N)(?I)\neg\neg G = 1$. However, as in Example 2.5, $(P,N)(?I)\neg\neg G = 1$ does not necessarily imply $(P,N)(?I)G = 1$.

In this paper, we propose a class of databases concerned with stable models [3] of general logic programs. Also the relation between the stable model and NAI is now presented.

Definition 3.1 Given a GDB (P,N) and a goal G, we define

$$G \equiv^{(P,N)} \neg\neg G \Leftrightarrow (P,N)(?I)G = (P,N)(?I)\neg\neg G.$$

We are now concerned with the stable model theory briefly.

Rule 3.2 Let P be a general logic program in propositional logic. Also let M be the set of atoms, and P_M stand for the program obtained by removing (1) the program clauses, and (2) the negative literals from P:

(1) the program clause whose body involves $\neg B$ such that $B \in M$.

(2) the negative literal in a program clause which is not removed by the procedure (1).

P_M is a definite clause set (that is, negation-free clause set). If M is equal to the minimal Herbrand model of P_M, then M is said a stable model.

In general, more than one stable model may exist for a given general logic program. If there is a stable model SM_P, then we define the following set by means of SM_P.

Definition 3.3 Let SM_P be a stable model of a general logic program P. We define

$$\Delta_{SM_P} = \{\neg n \mid n \notin SM_P\}, \text{ and } N_{SM_P} = \{n \mid n \notin SM_P\}.$$

$\Gamma \models A$ means Γ is a logical consequence of Γ. Assume

$$P \cup \Delta_{SM_P} \models L \text{ for a literal } L.$$

By Definition 3.3, SM_P is a model of $P \cup \Delta_{SM_P}$. Thus $L \in SM_P$ if L is an atom, and $L \in \Delta_{SM_P}$ if L is a negation of an atom.

In the database involving N_{SM_P}, we show that

$$P \cup \Delta_{SM_P} \models G \Leftrightarrow (P, N_{SM_P})(?I)G = 1.$$

Before proving it, we see an illustration in which the double negation law holds.

Example 3.4 Given a general logic program

$$P' = \{p, p \leftarrow \neg r, r \leftarrow \neg q\},$$

$\{p, r\}$ is a stable model. Let it be denoted by $SM_{P'}$. Then

$$N_{SM_{P'}} = \{q\}.$$

We observe the double negation problem as below.
(1) For the atomic goal p, $(P', N_{SM_{P'}})(?I)p = 1$, because $p \in P'$. Next we assume a goal $\neg\neg p$. The database is referred to by (P, N).

P	N	G
p,	q	$?\neg\neg\, p$
$p \leftarrow \neg\, r$,		
$r \leftarrow \neg\, q$,		

Since the goal is the double negation of p, p may be added to N, and the goal q, or $p \in N$ is to be examined.

P	N	G
p,	q	$?q$ (or $?p$)
$p \leftarrow \neg r$,	p	
$r \leftarrow \neg q$,		

The goal p succeeds, and thus $\neg\neg p$ does from the first database. Thus $(P', N_{SM_{P'}})(?I)\neg\neg p = 1$.
(2) For the goal r, $\neg q$ may be a goal by means of $r \leftarrow \neg q$. Then the goal q may be put to P. Because $q \in N$, $(P', N_{SM_{P'}})(?I)r = 1$. By Lemma 2.8, $(P', N_{SM_{P'}})(?I)\neg\neg r = 1$.
(3) Since the goal q does not succeed for P',

$$(P', N_{SM_{P'}})(?I)q \neq 1.$$

Now we investigate if $(P', N_{SM_{P'}})(?I)\neg\neg q \neq 1$. The goal q may be added to N, since the double negation of q is a goal. Then we can take q as a goal.

P	N	G
p,	q	$?q$
$p \leftarrow \neg\, r$,		
$r \leftarrow \neg\, q$,		

Because the goal q does not succeed,

$$(P', N_{SM_{P'}})(?I)\neg\neg q \neq 1.$$

Finally we see

$$L \equiv^{(P', N')} \neg\neg L$$

if $L = p$, or q, or r.

3.2 GDB by Means of Stable Model

Assuming the goal $\neg G$ takes the form

$$\neg(\bigwedge_i q_i \wedge \bigwedge_j \neg B_j),$$

we show that $G \equiv^{(P, N_{SM_P})} \neg\neg G$, where P is a general logic program, and N_{SM_P} is defined by using a stable model SM_P of P.

We firstly formulate a relation between the stable model and NAI.

Theorem 3.5 *Given a general logic program P, let SM_P stand for a stable model. For a goal G,*

$$P \cup \Delta_{SM_P} \models G \Leftrightarrow (P, N_{SM_P})(?I)G = 1.$$

Proof ($\Rightarrow$) We show by induction on the structure of the goal that if $P \cup \Delta_{SM_P} \models G$, then $(P, N_{SM_P})(?I)G = 1$. Firstly assume $P \cup \Delta_{SM_P} \models G$.
(1) In case G is a literal:
(a) If $G = \neg q$ for an atom q, then

$$P \cup \Delta_{SM_P} \models \neg q \Leftrightarrow q \notin SM_P.$$

Hence $q \in N_{SM_P}$. Because $(P \cup \{q\}, N_{SM_P})(?I)q = 1$, $(P, N_{SM_P})(?I)\neg q = 1$.
(b) If $G = q$ for an atom q, then

$$P \cup \Delta_{SM_P} \models q \Leftrightarrow q \in SM_P \Leftarrow q \in T_{SM_P} \uparrow \omega,$$

where $T_{SM_P} \uparrow \omega$ is defined as follows: For the Herbrand base Her of P_M such that $M = SM_P$, we define T_{SM_P}: $2^{Her} \rightarrow 2^{Her}$ to be

$$T_{SM_P}(I) = \{p \in Her \mid \begin{array}{l}\exists p \leftarrow q_1 \ldots q_n : \\ \{q_1, \ldots, q_n\} \subset I\}.\end{array}$$

Then set $T_{SM_P} \uparrow \omega = \bigcup_{i \in \omega} T_{SM_P} \uparrow i$, where

$$T_{SM_P} \uparrow n = \begin{cases} \emptyset \text{ (the empty set)} & (n = 0), \\ T_{SM_P}(T_{SM_P} \uparrow (n-1)) & (n > 0). \end{cases}$$

We next show by induction that

$$q \in T_{SM_P} \uparrow \omega \Rightarrow (P, N_{SM_P})(?I)q = 1$$

(i) Assume $q \in T_{P_M} \uparrow 1$. Then $q \leftarrow \in P_{SM_P}$, and

$$q \leftarrow \neg p_1 \ldots \neg p_m \in P,$$

where p_j is an atom for each j such that $p_j \notin SM_P$.
(i-a) In case $m = 0$: $q \leftarrow \in P$. Then

$$(P, N_{SM_P})(?I)q = 1.$$

(i-b) In case $m > 0$: As we see in (1)(a),

$$(P, N_{SM_P})(?I)\neg p_j = 1 \ (1 \leq j \leq m).$$

Hence $(P, N_{SM_P})(?I)\neg p_1 \ldots \neg p_m = 1$, where the body of a clause is interpreted as a conjunction of literals (that is, goals). Therefore

$$(P, N_{SM_P})(?I)q = 1.$$

(ii) Assume $q \in T_{SM_P} \uparrow n \ \ (1 \geq 1)$. Then

$$(P, N_{SM_P})(?I)q = 1.$$

Now assume $q \in T_{SM_P} \uparrow (n+1)$ on condition $q \leftarrow p_1 \ldots p_k \in P_{SM_P}$ such that $p_1, \ldots, p_k \in T_{SM_P} \uparrow n$. By the construction of P_{SM_P},

$$q \leftarrow p_1 \ldots p_k \neg r_1 \ldots neg r_m \in P$$

for $r_1, \ldots, r_m \notin SM_P$. By the induction hypothesis for $p_1, \ldots, p_k \in T_{SM_P} \uparrow n$,

$$(P, N)(?I)p_i = 1, 1 \leq i \leq k.$$

As in (1)(a),

$$(P, N)(?I)\neg r_j = 1, 1 \leq j \leq m.$$

Hence $(P, N)(?I)q = 1$. This completes the induction step. That is, if the goal is an atom q, then

$$\begin{array}{ll} & P \cup \Delta_{SM_P} \models q \\ \Leftrightarrow & q \in SM_P \\ \Leftrightarrow & q \in T_{SM_P} \uparrow \omega \\ \Rightarrow & (P, N_{SM_P})(?I)q = 1. \end{array}$$

By (a) and (b), if the goal G is a literal, then

$$P \cup \Delta_{SM_P} \models G \Rightarrow (P, N_{SM_P})(?I)G = 1.$$

(2) Assume $G = G_1 \wedge \ldots \wedge G_l$, and

$$P \cup \Delta_{SM_P} \models G_i \Rightarrow (P, N_{SM_P})(?I)G_i = 1,\ 1 \leq i \leq l.$$

By the induction hypothesis, we have

$$(P, N_{SM_P})(?I)G_i = 1,\ 1 \leq i \leq l.$$

Thus $(P, N_{SM_P})(?I)G = 1$. That is,

$$P \cup \Delta_{SM_P} \models G \Rightarrow (P, N_{SM_P})(?I)G = 1.$$

This completes the induction step.
(3) Assume $G = \neg(\bigwedge_i q_i \wedge \bigwedge_j \neg B_j)$. Then

$$\begin{array}{ll} & P \cup \Delta_{SM_P} \models G \\ \Rightarrow & \text{(a) } \exists q_i : P \cup \Delta_{SM_P} \models \neg q_i, \text{ or} \\ & \text{(b) } \exists B_j : P \cup \Delta_{SM_P} \models B_j. \end{array}$$

(Note $G = \bigvee_i \neg q_i \vee \bigvee_j B_j$.)
In case of (a): As in (1)(a), $q_i \notin SM_P$. Hence

$$\begin{array}{ll} & q_i \in N_{SM_P} \\ \Rightarrow & (P \cup \bigcup_i \{q_i\}, N_{SM_P} \cup \bigcup_j \{B_j\})(?I)q_i = 1 \\ \Rightarrow & (P, N)(?I)G = 1. \end{array}$$

In case of (b): Assume for B_j such that $P \cup \Delta_{SM_P} \models B_j$, $(P, N_{SM_P})(?I)B_j = 1$. Then by Lemma 2.7,

$$(P \cup \bigcup_i \{q_i\}, N_{SM_P} \cup \bigcup_j \{B_j\})(?I)B_j = 1.$$

By Definition 2.3, we have $(P, N_{SM_P})(?I)G = 1$.
By (a) and (b), the induction step is completed.
($\Leftarrow$) We show by induction on the rewritings of the goal that if $(P, N_{SM_P})(?I)G = 1$, then $P \cup \Delta_{SM_P} \models G$. Now assume $(P, N_{SM_P})(?I)G = 1$.
(1) For atomic goal G:
(a) In case $G \in P$: $G \in SM_P$. Hence $P \cup \Delta_{SM_P} \models G$.
(b) In case $G \leftarrow A \in P$ and $(P, N_{SM_P})(?I)A = 1$: By the induction hypothesis, $P \cup \Delta_{SM_P} \models A$. Hence $P \cup \Delta_{SM_P} \models G$. This completes the induction step.
(2) For a goal $G = G_1 \wedge \ldots \wedge G_n$:
From $(P, N_{SM_P})(?I)G = 1$, it follows $(P, N_{SM_P})(?I)G_i = 1$ for each i. By the induction hypothesis, $P \cup \Delta_{SM_P} \models G_i$ for each i. Therefore $P \cup \Delta_{SM_P} \models G$. This completes the induction step.
(3) For a goal $G = \neg(\bigwedge_i q_i \wedge \bigwedge_j \neg B_j)$: If $(P, N_{SM_P})(?I)G = 1$, then

$$\begin{array}{l} \exists A \in N_{SM_P} \cup \bigcup_j \{B_j\} : \\ (P \cup \bigcup_i \{q_i\}, N_{SM_P} \cup \bigcup_j \{B_j\})(?I)A = 1. \end{array}$$

(a) If $\exists q_i : q_i \notin SM_P$, then

$$\begin{array}{ll} & \neg q_i \in \Delta_{SM_P} \\ \Rightarrow & P \cup \Delta_{SM_P} \models \neg q_i \\ \Rightarrow & P \cup \Delta_{SM_P} \models G. \end{array}$$

(b) Assume $\forall q_i : q_i \in SM_P$.
(b-1) Assume $A \in N_{SM_P}$. Then $A \notin SM_P$. It follows $A \notin P$ and

$$\begin{array}{l} \exists A \leftarrow a_1 \ldots a_n \neg b_1 \ldots \neg b_m \in P, \\ \exists b_k\ (1 \leq k \leq m) \in SM_P, \\ \exists B \in N_{SM_P} \cup \bigcup_j \{B_j\} : \\ (P \cup \bigcup_i \{q_i\} \cup \{b_k\}, N_{SM_P} \cup \bigcup_j \{B_j\})(?I)B = 1. \end{array}$$

This case is reduced to another case of (b-2) that $B \in \bigcup_j \{B_j\}$.
(b-2) Assume $A \in \cup_j \{B_j\}$, that is, $A = B_l$ for some l. We have the following lemma.

Lemma 3.6 *Assume* $\forall q_i \in SM_P$, *and for* $B_l \in \cup_j \{B_j\}$

$$(P \cup \bigcup_i \{q_i\}, N_{SM_P} \cup \bigcup_j \{B_j\})(?I)B_l = 1.$$

Then $P \cup \Delta_{SM_P} \models B_l$.

Proof We prove by structural induction on B_l.
(i) (i-a) In case $B_l \in P \cup \bigcup_i \{q_i\}$ for atomic B_l: We have $B_l \in SM_P$. Hence $P \cup \Delta_{SM_P} \models B_l$.
(i-b) In case B_l is atomic and $B_l \leftarrow B \in P$ such that $(P \cup \bigcup_i \{q_i\}, N_{SM_P} \cup \bigcup_j \{B_j\})(?I)B = 1$: Then either
(i-b-α) $B_l \in SM_P$, or
(i-b-β) $B_l \notin SM_P$ and

$$\begin{array}{l} \exists q \in SM_P(\neg q \text{ in } B) : \\ (P \cup \bigcup_i \{q_i\}, N_{SM_P} \cup \bigcup_j \{B_j\})(?I)\neg q = 1. \end{array}$$

In case (i-b-α), $P \cup \Delta_{SM_P} \models B_l$.
The case (i-b-β) is reduced to the following case (ii-b), for the goal to be successful.
(ii) In case $B_l = \neg q$ for an atom q:
(ii-a) If $q \notin SM_P$, then $q \in N_{SM_P}$, and $\neg q \in \Delta_{SM_P}$. Hence $P \cup \Delta_{SM_P} \models \neg q$. That is, $P \cup \Delta_{SM_P} \models B_l$.
(ii-b) Assume $q \in SM_P$. Since

$$(P \cup \bigcup_i \{q_i\}, N_{SM_P} \cup \bigcup_j \{B_j\})(?I)\neg q = 1,$$

$$\begin{array}{l} \exists C \in N_{SM_P} \cup \bigcup_j \{B_j\} : \\ (P \cup \bigcup_i \{q_i\} \cup \{q\}, N_{SM_P} \cup \bigcup_j \{B_j\})(?I)C = 1. \end{array}$$

For the same reason as in (b-1) of the proof of the theorem, the case $C \in N_{SM_P}$ is reduced to the case below. Even if the case $C = B_l$ is repeated, the goal C is not successful. Hence we must consider the case $C \in \cup_j\{B_j\}$, $C = B_j \neq B_l$, as long as we assume the goal C is successful. We can omit the repetition of the same goal, and this case could be reduced to the case (i), (ii-a), (iii) or (iv), where the cases (iii) and (iv) are given below.
(iii) In case $B_l = C_1 \wedge \ldots \wedge C_n$: It follows

$$\forall k\text{: } (P \cup \bigcup_i\{q_i\}, N_{SM_P} \cup \bigcup_j\{B_j\})(?I)C_k = 1.$$

Since B_l is a conjunction of goals,

$$\begin{aligned} & \cup_j\{B_j\} \\ = \ & \{B_1, \ldots, B_l, \ldots, B_m\} \\ = \ & \{B_1, \ldots, B_{l-1}, C_1, \ldots, C_n, B_{l+1}, \ldots, B_m\}, \end{aligned}$$

where C_k is supposed to be in $\bigcup_j\{B_j\}$ for each k. By the induction hypothesis,

$$\forall k : P \cup \Delta_{SM_P} \models C_k.$$

Hence $P \cup \Delta_{SM_P} \models B_l$.
(iv) In case $B_l = \neg(\wedge_{i'} p_{i'} \wedge \bigwedge_{j'} \neg C_{j'})$: It follows

$$\begin{aligned} & \exists C \in N_{SM_P} \cup \bigcup_j\{B_j\} \cup \bigcup_{j'}\{C_{j'}\} : \\ & (P \cup \bigcup_i\{q_i\} \cup \bigcup_{i'}\{p_{i'}\}, \\ & N_{SM_P} \cup \bigcup_j\{B_j\} \cup \bigcup_{j'}\{C_{j'}\})(?I)C = 1. \end{aligned}$$

(iv-a) If $\exists p_{i'} : p_{i'} \notin SM_P$, then

$$\begin{aligned} p_{i'} \in N_{SM_P} \ & \Rightarrow \ P \cup \Delta_{SM_P} \models \neg p_{i'} \\ & \Rightarrow \ P \cup \Delta_{SM_P} \models B_l. \end{aligned}$$

(iv-b) If $\forall p_{i'} : p_{i'} \in SM_P$, then the case $C \in N_{SM_P}$ is reduced to the following, as we see in the case of (ii-b). Hence $C \in \cup_j\{B_j\} \cup \bigcup_{j'}\{C_{j'}\}$.
(iv-b-α) Assume $C = C_{j'}$. Since $C_{j'}$ is the form of a substructure of B_l, by the induction hypothesis, $P \cup \Delta_{SM_P} \models C_{j'}$. Therefore $P \cup \Delta_{SM_P} \models B_l$.
(iv-b-β) Assume $C = B_j$. This case is reduced to the case (i), (ii-a), (iii), (iv-a), or (iv-b-α), for the goal to be successful by finitely applying (i) – (iv). Lemma 3.6 Q.E.D.

By Lemma 2.6, $P \cup \Delta_{SM_P} \models B_l$. Hence $P \cup \Delta_{SM_P} \models G$. This completes the induction step of the case (3)(b). Q.E.D.

4 Theorem of Double Negation Law

In this section, a theorem of double negation law is given. That is, we show the double negation law holds for the database (P, N_{SM_P}) if we take a stable model SM_P for a general logic program P.

Theorem 4.1 *Assume a stable model SM_P for a given general logic program P. Let*

$$N_{SM_P} = \{n \mid n \notin SM_P\}.$$

Then

$$G \equiv^{(P, N_{SM_P})} \neg\neg G.$$

Proof For a goal G, by Theorem 3.5,

$$P \cup \Delta_{SM_P} \models G \Leftrightarrow (P, N_{SM_P})(?I)G = 1,$$

where $\Delta_{SM_P} = \{\neg n \mid n$ is an atom, and $n \notin SM_P\}$. In propositional logic,

$$P \cup \Delta_{SM_P} \models G \Leftrightarrow P \cup \Delta_{SM_P} \models \neg\neg G.$$

Hence $(P, N_{SM_P})(?I)G = (P, N_{SM_P})(?I)\neg\neg G$. That is, $G \equiv^{(P, N_{SM_P})} \neg\neg G$. Q.E.D.

As we see Theorem 2.6, given a general logic program P,

$$N_{NFA_P} = \{m \mid m \text{ is an atom, and } P(?F)m = 0\}$$

is presented so that

$$P(?F)G = 1 \Leftrightarrow (P, N_{NAF_P})(?I)G = 1.$$

In this paper, for a stable model SM_P of P,

$$N_{SM_P} = \{n \mid n \notin SM_P\}$$

is proposed such that

$$P \cup \Delta_{SM_P} \models G \Leftrightarrow (P, N_{SM_P})(?I)G = 1.$$

If $m \in N_{NAF_P}$, then because of $P(?F)m = 0$ $comp(P) \models \neg m$ (see [6], for example). Because the stable model SM_P is a model of $comp(P)$ [8], $SM_P \models \neg m$. Hence $m \notin SM_P$ and $m \in N_{SM_P}$. That is, if $m \in N_{NAF_P}$ then $m \in N_{SM_P}$. Since $N_{NAF_P} \subset N_{SM_P}$ by Theorem 2.6 and Lemma 2.7,

$$\begin{aligned} P(?F)G = 1 \ & \Leftrightarrow \ (P, N_{NAF_P})(?I)G = 1 \\ & \Rightarrow \ (P, N_{SM_P})(?I)G = 1. \end{aligned}$$

However, it does not necessarily follow

$$G \equiv^{(P, N_{SM_P})} \neg\neg G.$$

On the other hand, if we have $P = \{q, p \leftarrow p\}$, then $\{q\}$ is a stable model of P. Let it be abbreviated by SM_P. Then $p \in N_{SM_P}$. But $p \notin N_{NAF_P}$. Because there is no finite failure for $\leftarrow p$.

Therefore the class of databases (P, N_{SM_P}) is different from that of databases (P, N_{NAF_P}) in [2].

5 Concluding Remarks

We have dealt with a problem of the double negation law in the database, GDB. Taking the stable model into account, we have constructed N_{SM_P} as a set of goals for a stable model SM_P of a given general logic program P.

For simplicity of treatment, we are concerned with only propositional logic. For a stable model SM_P, we define

$$\begin{aligned} \Delta_{SM_P} &= \{\neg n \mid n \text{ is an atom, and } n \notin SM_P\}, \\ N_{SM_P} &= \{n \mid \text{ is an atom, and } n \notin SM_P\} \end{aligned}$$

and present the relation:

$$P \cup \Delta_{SM_P} \models G \Leftrightarrow (P, N_{SM_P})(?I)G = 1.$$

By means of this relation, we have proven

$$G \equiv^{(P,N_{SM_P})} \neg\neg G.$$

To lift the primary result to the first-order logic, we can take an existentially quantified goal, where the variables in the goal may be bound to the terms with function symbols occurring in the database. The treatment of existentially quantified goals is exhaustively done by means of that in propositional logic. (We will show it in another paper.)

The significance of the double negation law in GDB comes from the reason $\neg\neg G$ is regarded as the same as G in (P, N_{SM_P}) even if the goal G is hierarchically structured by negations and conjunctions. We may check if the goal G is successful, when $\neg\neg G$ is tested.

References

[1] K.Eshghi and R.A.Kowalski, Abduction Compared with Negation by failure, Proc. 6th Int. Conf. on Logic Programming, pp.234-254 (1989).

[2] D.M.Gabbay and M.J.Sergot, Negation as Inconsistency,I, The Journal of Logic Programming 1, pp.1-35 (1986).

[3] M.Gelfond and V.Lifschitz, The Stable Model Semantics for Logic Programming. Proc. 5th Int. Cont. Symp. on Logic Programming, 1070-1080 (1988).

[4] M.Gelfond et al., On the relation between circumscription and negation as failure, Artificcial Intelligence 38, pp.75-94 (1992).

[5] A.C.Kakas, R.A.Kowalski and F.Toni, Abductive Logic Programming, J. Logic and Computation 2, 6, pp.719-770 (1992).

[6] J.W.Lloyd, Foundations of Logic Programming, 2nd, Extended Edition, Springer - Verlag (1987).

[7] J.Lobo, J.Minker and A.Rajaseker, Foundations of Disjunctive Logic Programming, MIT Press (1992).

[8] W.Marek and V.S.Subrahmanian, The relatinship between stable, supported, default and autoepistemic samantics for general logic programs, Theoretical Computer Science 103, 365-386, (1992).

[9] H.Przymusinska and T.C.Przymusinski, Semantic issues in deductive databases and logic programs, in: R.B.Banerji (ed.), Formal Techniques in Artificial Intelligence: A Sourcebook, North-Holland, 321-367 (1990).

[10] T.C.Przymusinski, On the Declarative and Procedual Semantics of Logic Programs, J. Automated Reasoning 5, pp.167-206 (1985).

[11] R.Reiter, On closed world data bases, in: H.Gallaire and J.Minker (eds.), Logic and Data Bases, Plenum Press, 55-76 (1978).

[12] J.C.Shepherdson, Negation as Failure : A Comparison of Clark's Completed Data Base and Reiter's Closed World Assumption, J. Logic Programming 1, pp.51-79 (1984).

[13] J.C.Shepherdson, Negation as Failure, II, J. Logic Programming 3, pp.185-202 (1985).

[14] J.C.Shepherdson, Negation in Logic Programming, in J.Minker (ed.), Foundations of Deductive Database and Logic Programming, pp.19-88, Morgan Kaufmann Publishers (1987).

[15] J.C.Shepherdson, A Sound and Complete Semantics for a Version of Negation as Failure, Theoret. Comput. Sci. 65, pp.343-371 (1989).