On Positive Occurrences of Negation as Failure*

Katsumi Inoue
Department of Information and Computer Sciences
Toyohashi University of Technology
Tempaku-cho, Toyohashi 441, Japan
inoue@tutics.tut.ac.jp

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

Abstract

Logic programs with positive occurrences of negation as failure have recently been introduced as a subset of the logic of minimal belief and negation as failure (MBNF). A unique feature of such programs, which other traditional logic programs lack, is that the minimality of answer sets does not hold. We reveal in this paper that this property is important for applying logic programming to represent abduction and inclusive disjunctions. With its rich expressiveness, however, the computational complexity of such extended programs is shown to remain in the same complexity class as normal disjunctive programs. Through the elimination of negation as failure from programs, computation of such extended programs is realized using bottom-up model generation techniques. A simple translation of programs into autoepistemic logic is also presented.

1 Introduction

Most of semantics of logic programs proposed so far satisfy the *principle of minimality* in some sense. For example, the least model semantics for definite Horn programs, the minimal model semantics for positive disjunctive programs, the prefect model semantics for stratified programs and the stable model semantics for normal programs satisfy the principle in the sense that every canonical model of a logic program is its minimal model. The answer set semantics for extended logic programs by Gelfond and Lifschitz (1991) also satisfies the principle since no answer set of a program is smaller than its other answer set. Hence, it has been argued that the principle of minimality is one of the most important goals that any "commonsense" semantics should obey (Schlipf, 1992).

The situation is similar in research on nonmonotonic formalisms. Circumscription is directly based on minimal models, and (disjunctive) default logic has the property that an extension of a (disjunctive) default theory is not a subset of other extension. While an exception can be seen in *autoepistemic logic* (Moore, 1985), the definition of *stable expansions* has been modified so that each obtainable expansion "rationally" satisfies the principle of minimality. For example, $\{Bp \supset p\}$ has two stable expansions, one containing p and the other not in their objective parts, but only the latter is the moderately (or strongly) grounded expansion (Konolige, 1988).

On the other hand, recent research on the semantics of logic programming and nonmonotonic reasoning has demonstrated that both fields have influenced each other. The logic of *minimal belief and negation as failure* (MBNF) recently proposed by Lifschitz (1992) is a nonmonotonic modal logic that directly allows the negation-as-failure operator *not* in a theory. MBNF is one of the most expressive logics and can serve as a common framework that unifies several nonmonotonic formalisms. As Lifschitz noted, however, MBNF is purely semantical and too intractable to be used directly for representing knowledge. Then, Lifschitz and Woo (1992) investigate a large subset of propositional MBNF called *PL-theories*—theories with "protected literals". The semantics of PL-theories is similar to the answer set semantics for extended disjunctive programs (Gelfond and Lifschitz, 1991), and can be described in terms of sets of objective literals. Moreover, each PL-theory is shown to be replaced with an equivalent set of disjunctions of protected literals. This "logic programming" fragment of MBNF can be expressed as

*In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, Morgan Kaufmann, pp. 293–304, 1994.

a program consisting of rules of the form:

$$L_1 \mid \ldots \mid L_k \mid not\, L_{k+1} \mid \ldots \mid not\, L_l \leftarrow L_{l+1}, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n$$

where each L_i is a positive or negative literal. Then, the class of logic programs allowing the above form of rules strictly includes the class of extended disjunctive programs. Interestingly, once *not* appears positively as above, the principle of minimality does not hold any more. For example, the program consisting of the rule

$$p \mid not\, p \leftarrow \tag{1}$$

has two answer sets: one containing p, and the other including neither p nor $\neg p$.

Then, two criticisms can be made about the use of negation as failure positively in logic programming or MBNF. The first criticism is argued by the fact that one feels a resistance to the existence of non-minimal answer sets. In other words, from the traditional viewpoint, a non-minimal answer set contains a redundant information and is of no use for representing commonsense knowledge. In fact, Lifschitz and Woo also raise a question about the utility of a disjunction of literals and their negations like rule (1), and discuss (Lifschitz and Woo, 1992, page 608):

> *It remains to be seen whether rules like this may have applications to knowledge representation.*

The second criticism addresses the increase of computational complexity and the difficulty in supplying a procedural semantics in the presence of non-minimal answer sets. Two proof theories for MBNF proposed so far are not sufficient in this respect. Chen (1993) proposes a proof theory for PL-theories, which relies on the proof theory for the logic of only knowing by Levesque, but a procedure would have to deal with modal logic K45. Beringer and Schaub (1993) provide a proof procedure for a subset of MBNF, but this subset neither includes extended disjunctive programs nor allows a positive occurrence of *not*.

In this paper, we discuss the above two issues on non-minimal answer sets of PL-theories. In the first respect, we reveal that the non-minimality of answer sets is an important property for applying logic programming or MBNF to represent abduction and to interpret disjunctions inclusively. In the second regard, we show that the computational complexity of extended programs with positive occurrences of *not* is in the same complexity class as normal disjunctive programs. Furthermore, through the elimination of *not* from programs, computation of such extended programs is shown to be realized using bottom-up model generation theorem proving techniques.

The fact that abduction can be represented by a single logic program is a particularly striking result. Since an abductive program is usually represented by a pair of background knowledge and candidate hypotheses, it is important to know how such meta-level information of hypotheses can be expressed at the object level. Such an expression bridges the gap between abductive and usual (non-abductive) logic programming, and is useful for computational aspect of abduction since we can apply any proof procedure for usual logic programs to programs transformed from abductive frameworks. One of our proposed solutions to Lifschitz and Woo's question about the utility of non-minimal answer sets also appears at this point. Namely, the rule (1), $p \mid not\, p \leftarrow$, can be used to represent the statement that p is a hypothesis.

From the viewpoint of nonmonotonic reasoning, among many nonmonotonic formalisms, Moore's autoepistemic logic can express a stable expansion whose objective part is larger than that of another expansion. We will show that this non-minimal feature of autoepistemic logic is applicable to describe the semantics of logic programming with positive occurrences of *not*. This result is obtained from a simple translation of programs into autoepistemic logic based on results by Lifschitz and Schwarz (1993) and Chen (1993).

The rest of this paper is organized as follows. In Section 2, we give the answer set semantics for programs with positive occurrences of *not*. To show practical applications of non-minimal answer sets, abduction and inclusive disjunctions are characterized by positive occurrences of *not* in Section 3. Section 4 provides complexity results and computation of the answer set semantics, and the connection to autoepistemic logic is shown in Section 5. Some related work is discussed in Section 6, and Section 7 gives a summary.

2 Answer Sets of Programs with Positive Occurrences of *not*

This section overviews the answer set semantics of logic programs with *positive occurrences of negation as failure* (hereafter called *positive not*). Since a rule with variables stands for the set of ground instances in the semantics of logic programming, we can restrict our attention to ground programs.

A *general extended disjunctive program* (GEDP) is a

set of rules of the form

$$L_1 \mid \ldots \mid L_k \mid not\, L_{k+1} \mid \ldots \mid not\, L_l \leftarrow L_{l+1}, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n \quad (2)$$

where L_i's are literals and $n \geq m \geq l \geq k \geq 0$. The disjunction in the left of $\leftarrow$ is called the *head* and the conjunction in the right of $\leftarrow$ is called the *body* of the rule. A GEDP is called an *extended disjunctive program* (EDP) when it contains no positive *not*, i.e., each rule is in the form (2) with $k = l$. An EDP is called (i) an *extended logic program* if for each rule $l \leq 1$; and (ii) a *normal disjunctive program* (NDP) if every L_i is an atom. An NDP is called (i) a *normal logic program* if for each rule $l \leq 1$; and (ii) a *positive disjunctive program* (PDP) if it contains no *not*, i.e., for each rule $m = n$.

The *answer sets* of a GEDP are defined by the following two steps. First, let P be a *not*-free extended disjunctive program (i.e., for each rule $k = l$ and $m = n$), and $S \subseteq Lit$ where Lit is the set of all ground literals in the language. Then, S is an *answer set* of P iff S is a minimal set satisfying the conditions:

1. S *satisfies* each ground rule from P:

   $$L_1 \mid \ldots \mid L_k \leftarrow L_{k+1}, \ldots, L_m\,,$$

   that is, if $\{L_{k+1}, \ldots, L_m\} \subseteq S$, then $L_i \in S$ for some $1 \leq i \leq k$. In particular, for each ground rule $\leftarrow L_1, \ldots, L_m$ from P, $\{L_1, \ldots, L_m\} \not\subseteq S$;

2. If S contains a pair of complementary literals L and $\neg L$, then $S = Lit$.

Secondly, let Π be any general extended disjunctive program, and $S \subseteq Lit$. The *reduct* Π^S *of* Π *by* S is a *not*-free extended disjunctive program obtained as follows: A rule

$$L_1 \mid \ldots \mid L_k \leftarrow L_{l+1}, \ldots, L_m$$

is in Π^S iff there is a ground rule of the form (2) from P such that

$$\{L_{k+1}, \ldots, L_l\} \subseteq S \;\text{ and }\; \{L_{m+1}, \ldots, L_n\} \cap S = \emptyset\,.$$

For programs of the form Π^S, their answer sets have already been defined. Then, S is an *answer set* of Π iff S is an answer set of Π^S. We say an answer set is *consistent* if it is not Lit. An answer set S of a GEDP Π is *minimal* if no other answer set S' of Π satisfies $S' \subset S$; otherwise, it is *non-minimal*.

Note that the above definition of answer sets of a GEDP is given in a way slightly different from that by Lifschitz and Woo (1992) who additionally include in the language two special atoms T and F. When the language does not contain these special atoms, our definition of the reduct is equivalent to that given in (Lifschitz and Woo, 1992, page 606), and thus both definitions of answer sets coincide. Obviously, when a program Π is an extended disjunctive program, the above definition of answer sets reduces to that given by Gelfond and Lifschitz (1991).

An important property of GEDPs, which has been observed by Lifschitz and Woo, is that the minimality of answer sets for EDPs (Lifschitz and Woo, 1992, Theorem 4) no longer holds. For example, the program

$$\{\;\; p \mid not\, p \leftarrow\;,\quad \neg p \mid not\, \neg p \leftarrow\;\;\}$$

has four answer sets: $\emptyset$, $\{p\}$, $\{\neg p\}$ and Lit.

3 Representing Abduction by GEDP

Abduction is a very important form of reasoning not only for various AI problems but also for logic programming. *Abductive logic programming* is an extension of logic programming to perform abductive reasoning (Kakas *et al.*, 1992). In this section, we will show that this extension can be characterized exactly using positive *not* in GEDPs, so that both abductive and non-abductive logic programming have the same expressive power. We will also show that positive *not* is a very useful tool to represent other non-minimal semantics for disjunctive logic programs.

3.1 Belief Sets for Abductive Programs

The semantics of abduction we consider here is based on the *generalized stable model semantics* defined by Kakas and Mancarella (1990) and the *belief set semantics* by Inoue and Sakama (1993), but is extended to handle general extended disjunctive programs.

An *abductive (general extended disjunctive) program* is a pair $\langle P, \Gamma \rangle$, where P is a (general extended disjunctive) program and Γ ($\subseteq Lit$) is a set of ground literals from P called *abducibles*. When P is a normal logic program and Γ is a set of atoms, we will often call an abductive program an *abductive normal logic program*. Let E be a subset of Γ. A set of literals S_E is a *belief set* of $\langle P, \Gamma \rangle$ if it is a consistent answer set of $P \cup E$ and satisfies $E = S_E \cap \Gamma$. A belief set S_E is *minimal* if no belief set $S_{E'}$ satisfies that $E' \subset E$. Note that each belief set reduces to a consistent answer set of P when $\Gamma = \emptyset$. The condition $E = S_E \cap \Gamma$ is necessary since an abducible appearing in the head of a ground rule may become true when other abducibles from E

are true. In this way, each belief set S_E is uniquely associated with its "generating" abducibles E.[1]

Let $\langle P, \Gamma \rangle$ be an abductive program, and O a literal. A set $E \subseteq \Gamma$ is an *explanation of* O (*wrt* $\langle P, \Gamma \rangle$) if there is a belief set S_E which satisfies O. An explanation E of O is *minimal* if no $E' \subset E$ is an explanation of O. Without loss of generality, we can assume that an observation O is a non-abducible ground literal. Further, the problem to find explanations is essentially equivalent to find belief sets, since E is a minimal explanation of O wrt $\langle P, \Gamma \rangle$ iff S_E is a minimal belief set of $\langle P \cup \{ \leftarrow not\, O\}, \Gamma \rangle$ (Inoue and Sakama, 1993).

Example 3.1 Consider an abductive program $\langle P, \Gamma \rangle$ where $\Gamma = \{a, b\}$ and

$$P = \{\ p \leftarrow r,\, b,\, not\, q\,, \quad q \leftarrow a\,, \quad r \leftarrow\ \}\,.$$

Then, $S_{E0} = \{r\}$, $S_{E1} = \{r, p, b\}$, $S_{E2} = \{r, q, a\}$ and $S_{E3} = \{r, q, a, b\}$ are the belief sets of $\langle P, \Gamma \rangle$, in which S_{E0} is the only minimal belief set of $\langle P, \Gamma \rangle$. Suppose that p is an observation. Then, $E1 = S_{E1} \cap \Gamma = \{b\}$ is the (minimal) explanation of p. This observation p can be incorporated in the program as

$$P' = P \cup \{\ \leftarrow not\, p\ \}\,,$$

and the unique belief set of $\langle P', \Gamma \rangle$ is $S_{E1} = \{r, p, b\}$. Note that $E3 = \{a, b\}$ is not an explanation of p because if we would abduce $E3$, q would block to derive p, and $\leftarrow not\, p$ could not be satisfied.

3.2 Characterizing Abductive Programs

The most direct way to embed abducibles into a single program is as follows. Let $\langle P, \Gamma \rangle$ be an abductive program. For each abducible γ in Γ, we supply the rule

$$\gamma \mid not\, \gamma \leftarrow\ . \tag{3}$$

According to the non-minimality of answer sets of GEDPs, this rule has the effect to augment each answer set of P with either γ or nothing. Given an abductive program $\langle P, \Gamma \rangle$, let P_Γ be the GEDP $P \cup abd(\Gamma)$ where $abd(\Gamma)$ is the set of rules (3) obtained from Γ.

Theorem 3.1 *A set S_E is a belief set of $\langle P, \Gamma \rangle$ iff S_E is a consistent answer set of P_Γ.*

[1] Kakas and Mancarella's *generalized stable models* of an abductive normal logic program $\langle P, \Gamma \rangle$ are exactly our belief sets of the program. Note that they require that each abducible must not appear in the heads of rules of P, so that the condition $E = S_E \cap \Gamma$ is always satisfied. They further separate integrity constraints from the background program P, but we include them in P.

Proof: Since $E = S_E \cap \Gamma$, it holds that $abd(\Gamma)^{S_E} = abd(\Gamma)^E = E = E^{S_E}$. Hence,
S_E is a belief set of $\langle P, \Gamma \rangle$
iff S_E is a consistent answer set of $P \cup E$ and $E = S_E \cap \Gamma$
iff S_E is a consistent answer set of $P^{S_E} \cup E^{S_E}$
and $E = S_E \cap \Gamma$
iff S_E is a consistent answer set of $P^{S_E} \cup abd(\Gamma)^{S_E}$
iff S_E is a consistent answer set of $P \cup abd(\Gamma)$. □

Given a GEDP Π and a set Γ of ground literals, we say an answer set S of Π is Γ-*minimal* if no other answer set S' of Π satisfies that $S' \cap \Gamma \subset S \cap \Gamma$.

Corollary 3.2 *S_E is a minimal belief set of $\langle P, \Gamma \rangle$ iff S_E is a consistent Γ-minimal answer set of P_Γ.* □

Example 3.2 The abductive program $\langle P', \Gamma \rangle$ given in Example 3.1 is translated into

$$P'_\Gamma = P' \cup \{\ a \mid not\, a \leftarrow\ ,\quad b \mid not\, b \leftarrow\ \}\,.$$

Then, $\{r, p, b\}$ is the unique (and hence Γ-minimal) answer set of P'_Γ, which is exactly the (minimal) belief set of $\langle P', \Gamma \rangle$. Notice in this example that there is no non-minimal answer set of P'_Γ. In other words, translating abducibles into rules with positive *not* (3) not only enables us to represent non-minimal belief sets of abductive programs, but plays an important role to obtain a (minimal) explanation.

3.3 Assumptions with Preconditions

In the previous subsections, a set Γ of abducibles in an abductive program $\langle P, \Gamma \rangle$ was defined as a set of literals. Often however, we would like to introduce in Γ an *abducible rule* like

$$\gamma \leftarrow L_1,\, \ldots,\, L_m,\, not\, L_{m+1},\, \ldots,\, not\, L_n\,, \tag{4}$$

where γ and L_i's are literals. This abducible rule intuitively means that if the rule is abduced then it is used for inference together with the background rules from P. This kind of extended abductive framework was introduced by Inoue (1991) as a *knowledge system* in which both P and Γ are defined as extended logic programs, and has been shown to be a useful tool for representing commonsense knowledge.

An abducible rule (4) has the effect to introduce the literal γ as an assumption in a particular context in which the body of the rule is true. In this sense, γ in (4) can be considered as an *assumption with preconditions*. On the other hand, each abducible literal γ in an abductive program $\langle P, \Gamma \rangle$ defined in Section 3.1 is viewed as an abducible rule without precondition $\gamma \leftarrow$, and hence can be abduced globally.

An extended abductive framework can be formally defined as a pair $\langle P, \Gamma \rangle$, where P is a GEDP and Γ is now an extended logic program consisting of rules of the form (4). The semantics of this abductive framework is slightly extended from that given in Section 3.1 as follows. Let E be any subset of Γ, and $head(E)$ be the heads of rules in E. A set of literals S_E is a *belief set* of $\langle P, \Gamma \rangle$ if it is a consistent answer set of $P \cup E$ and satisfies that $head(E) = S_E \cap head(\Gamma)$. Clearly, this notion of belief sets reduces to the definition of belief sets in Section 3.1 when Γ is a set of abducible literals without preconditions.

Example 3.3 Suppose that $\langle P, \Gamma \rangle$ is an abductive program where

$$\begin{aligned} P &= \{\ p \leftarrow a\,,\quad \neg p \leftarrow b\,,\quad q \leftarrow c\ \}\,, \\ \Gamma &= \{\ a \leftarrow\,,\quad b \leftarrow\,,\quad c \leftarrow p\ \}\,. \end{aligned}$$

Then, $\langle P, \Gamma \rangle$ has the four belief sets: $\emptyset$, $\{a, p\}$, $\{a, p, c, q\}$, and $\{b, \neg p\}$. Notice that $\{b, \neg p, c, q\}$ is not a belief set since c can be assumed only when p is true.

The embedding of assumptions with preconditions into GEDPs is a straightforward generalization of that of abducibles without preconditions. Each rule (4) in Γ is replaced with the rule

$$\gamma \mid not\, \gamma \leftarrow L_1, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n\,. \quad (5)$$

For example, the abducible rules Γ given in Example 3.3 are translated into

$$a \mid not\, a \leftarrow\,, \quad b \mid not\, b \leftarrow\,, \quad c \mid not\, c \leftarrow p\,.$$

Corollary 3.3 *Let $\langle P, \Gamma \rangle$ be an abductive framework, and $abd(\Gamma)$ the set of rules (5) obtained from the rules (4) in Γ. A set S_E is a belief set of $\langle P, \Gamma \rangle$ iff S_E is a consistent answer set of $P \cup abd(\Gamma)$.* □

3.4 Inclusive Interpretation of Disjunctions

Another important application of positive *not* is to express an alternative semantics for disjunctive logic programs other than Gelfond and Lifschitz's answer set semantics. Here, we show that the *possible model semantics* for normal disjunctive programs by Sakama and Inoue (1993b) can be characterized by the answer set semantics for GEDPs.

The possible model semantics was initially introduced for positive disjunctive programs to enable one to specify both inclusive and exclusive interpretations of disjunctions (Sakama, 1989; Chan, 1993).[2] Recently, Sakama and Inoue (1994) have presented the equivalence between the possible model semantics for NDPs and the generalized stable model semantics for abductive normal logic programs. Utilizing this result and Theorem 3.1, the embedding of the possible model semantics into GEDPs can be obtained. We show below a direct method to do it based on the embedding of abducible rules into GEDPs in Section 3.3.

For an NDP P, let $disj(P)$ be the disjunctive rules of P, i.e., those rules having more than one atoms in their heads. A *split program* of P is a ground normal logic program obtained from P by replacing each ground disjunctive rule from $disj(P)$ of the form

$$A_1 \mid \ldots \mid A_k \leftarrow A_{k+1}, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n \quad (6)$$

$(k > 1)$ with rules

$$A_i \leftarrow A_{k+1}, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n \quad (7)$$

for every $A_i \in S$, where S is some non-empty subset of $\{A_1, \ldots, A_k\}$. Then, a *possible model* of P is an answer set (or *stable model*) of any split program of P (Sakama and Inoue, 1993b). Note that every stable model of P is a possible model of P, but not vice versa. For example, when

$$P = \{\ p \mid q \leftarrow\,,\quad q \leftarrow p,\quad r \leftarrow not\, p\ \}\,,$$

$\{q, r\}$ is both a stable model and a possible model of P, but another possible model $\{p, q\}$ is not a stable model. Clearly, for normal logic programs, possible models coincide with stable models.

To obtain every possible model, let us consider the translation *pm* which maps an NDP to a GEDP. Given an NDP P, $pm(P)$ is obtained by replacing every rule from P of the form (6) with $k + 1$ rules. The first k rules of them are

$$A_i \mid not\, A_i \leftarrow A_{k+1}, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n \quad (8)$$

for $i = 1, \ldots, k$, and the other one is

$$\leftarrow A_{k+1}, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n, \quad not\, A_1, \ldots, not\, A_k\,. \quad (9)$$

Recall that the embedding of abducible rules (4) into GEDPs was based on rules (5). The embedding of possible models is achieved in a similar manner by (8) except that the empty selection from the disjuncts of each disjunction is rejected by (9) in the *pm* translation.

Theorem 3.4 *Let P be an NDP. A set S of atoms is a possible model of P iff S is an answer set of $pm(P)$.*

[2] Possible model semantics is also called *possible world semantics* in (Chan, 1993; Sakama and Inoue, 1993b). While Chan (1993) gives a different definition from that by (Sakama, 1989), these notions are proved to be equivalent.

Proof: Suppose that Γ is the normal logic program obtained from $disj(P)$ by replacing each disjunctive rule (6) with k rules of the form (7) for $i = 1, \ldots, k$, and that IC is the set of rules of the form (9) obtained from the rules of the form (6) in $disj(P)$. Then, a set S of atoms is a possible model of P iff S is a belief set of the abductive program $\langle (P \setminus disj(P)) \cup IC, \Gamma \rangle$ (Sakama and Inoue, 1994). Hence, the theorem follows from Corollary 3.3. □

Example 3.4 (Chan, 1993) Suppose that P consists of rules

$$\begin{aligned} &violent \mid psychopath \leftarrow suspect\,, \\ &dangerous \leftarrow violent,\ psychopath\,, \\ &suspect \leftarrow\,. \end{aligned}$$

The first rule is replaced in $pm(P)$ with three rules

$$\begin{aligned} &violent \mid not\ violent \leftarrow suspect\,, \\ &psychopath \mid not\ psychopath \leftarrow suspect\,, \\ &\leftarrow suspect\,, not\ violent,\ not\ psychopath\,. \end{aligned}$$

Then, $\{suspect, violent\}$, $\{suspect, psychopath\}$ and $\{suspect, violent, psychopath, dangerous\}$ are the three answer sets of $pm(P)$, which coincide with the possible models of P. Note that the first and second possible models of P are also the answer sets of P, while the third possible model is not. If we introduce the closed world assumption

$$\neg A \leftarrow not\ A \quad \text{for any atom } A$$

into P, then the answer set semantics entails $\neg dangerous$,[3] which is too strong. The possible model semantics for P (the answer set semantics for $pm(P)$) in this case does not entail $\neg dangerous$.

4 Complexity and Computation

We now show the computational complexity of GEDPs and an algorithm to compute the answer sets of a finite GEDP. These results indicate that positive *not* can be eliminated from programs so that we can use any proof procedure for computing EDPs.

4.1 Reduction to Extended Disjunctive Programs

We first show a polynomial-time transformation from a GEDP to an extended disjunctive program. Let Π be any GEDP. The extended disjunctive program $edp(\Pi)$ is obtained from Π by replacing each rule with positive *not* in Π of the form:

$$\begin{aligned} &L_1 \mid \ldots \mid L_k \mid not\ L_{k+1} \mid \ldots \mid not\ L_l \\ &\quad \leftarrow L_{l+1}, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n \end{aligned} \tag{10}$$

$(n \geq m \geq l > k \geq 0)$ with rules without positive *not*:

$$\begin{aligned} &\lambda_1 \mid \ldots \mid \lambda_k \mid \lambda_{k+1} \mid \ldots \mid \lambda_l \\ &\quad \leftarrow L_{l+1}, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n\,, \end{aligned} \tag{11}$$

$$L_i \leftarrow \lambda_i \quad \text{for } i = 1, \ldots, k\,, \tag{12}$$

$$\lambda_i \leftarrow L_i, L_{k+1}, \ldots, L_l \quad \text{for } i = 1, \ldots, k\,, \tag{13}$$

$$\leftarrow \lambda_i, not\ L_j \quad \begin{array}{r}\text{for } i = 1, \ldots, k \text{ and} \\ j = k+1, \ldots, l\,,\end{array} \tag{14}$$

$$\leftarrow \lambda_j, L_j \quad \text{for } j = k+1, \ldots, l\,. \tag{15}$$

Here, λ_i is a new atom not appearing elsewhere in Π and is uniquely associated with each disjunct of a ground rule from Π. In the following, we denote by Lit_Π the set of all ground literals in the language of Π. Thus, Lit_Π includes no new atom λ_i.

Theorem 4.1 *Let Π be a GEDP. A set S is an answer set of Π iff a set Σ is an answer set of $edp(\Pi)$ such that $S = \Sigma \cap Lit_\Pi$.*

Proof: Let S be an answer set of Π. First, consider the reduct Π^S. If a rule

$$L_1 \mid \ldots \mid L_k \leftarrow L_{l+1}, \ldots, L_m \tag{16}$$

is in Π^S, then for the corresponding rule (10) in Π, it holds that $\{L_{k+1}, \ldots, L_l\} \subseteq S$ and $\{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$. In this case, the reduct $edp(\Pi)^S$ includes

$$\lambda_1 \mid \ldots \mid \lambda_k \mid \lambda_{k+1} \mid \ldots \mid \lambda_l \leftarrow L_{l+1}, \ldots, L_m, \tag{17}$$

and the rules (12,13,15), but does not contain the rule (14). Since S satisfies each rule in Π^S, for each rule R of the form (16) such that $\{L_{l+1}, \ldots, L_m\} \subseteq S$, there exists $L_i \in S$ for some $1 \leq i \leq k$. Let

$$\Sigma 1 = \bigcup_{R \in \Pi^S} \{\lambda_i \mid L_i \in S,\ \ 1 \leq i \leq k\}.$$

Next, suppose that there is a rule (10) in Π such that $\{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$ but $\exists L_j \notin S$ for some $k+1 \leq j \leq l$. In this case, there is no corresponding rule (16) in Π^S, but the rule (17) is present in $edp(\Pi)^S$. $edp(\Pi)^S$ also contains the rules (12,13,15) and the rules $\leftarrow \lambda_i$ for $i = 1, \ldots, k$ (from the rule (14)). Then, for each such rule R' (17) of $edp(\Pi)^S$, let

$$\Sigma 2 = \bigcup_{R' \in edp(\Pi)^S} \{\lambda_j \mid L_j \notin S,\ \ k+1 \leq j \leq l\}.$$

[3] The answer set semantics for a program P is said to *entail* a literal L if L is included in all answer sets of P.

Now let $\Sigma = S \cup \Sigma 3$, where $\Sigma 3$ is a minimal subset of $\Sigma 1 \cup \Sigma 2$ such that each λ_i or λ_j is chosen in a way that Σ satisfies every rule of the form (17). Obviously, it holds that $S = \Sigma \cap Lit_\Pi$. Because new literals λ_i's never appear within *not*, the program $edp(\Pi)^S$ is exactly the same as $edp(\Pi)^\Sigma$. Then, Σ satisfies all the rules of $edp(\Pi)^\Sigma$, and if $S = Lit_\Pi$ then $Lit_\Pi \subseteq \Sigma$.

To see that Σ is a minimal set satisfying the rules of $edp(\Pi)^\Sigma$, notice that S is a minimal set satisfying the rules of Π^S. From the construction of $\Sigma 3$, it is easy to see that Σ is a minimal set *containing* S and satisfying the rules of $edp(\Pi)^\Sigma$. We thus only need to verify that there is no Σ' such that (i) $\Sigma' \subset \Sigma$, (ii) Σ' satisfies the rules of $edp(\Pi)^\Sigma$, and (iii) $S' \subset S$ for $S' = \Sigma' \cap Lit_\Pi$. Suppose to the contrary that such a Σ' exists. Then, the condition (iii) is satisfied only if there exist rules (12,13) such that $L_i \in S \setminus S'$ and $\lambda_i \in \Sigma \setminus \Sigma'$ for some $1 \leq i \leq k$. For this λ_i, there must be the rule (17) such that $\{L_{l+1}, \ldots, L_m\} \subseteq S'$. By the condition (ii), there is a literal $\lambda_j \in \Sigma'$ for some $k+1 \leq j \leq l$. This λ_j, however, is not included in $\Sigma 2$ by (15), contradicting the condition (i). Therefore, Σ is an answer set of $edp(\Pi)^\Sigma$, and hence an answer set of $edp(\Pi)$.

Conversely, let Σ be an answer set of $edp(\Pi)$, and $S = \Sigma \cap Lit_\Pi$. Since Σ is an answer set of $edp(\Pi)^\Sigma$, for each rule (17) in $edp(\Pi)^\Sigma$, if $\{L_{l+1}, \ldots, L_m\} \subseteq S$, then $\lambda_i \in \Sigma$ for some $1 \leq i \leq l$. There are two cases: (a) If $\lambda_i \in \Sigma$ for some $1 \leq i \leq k$, then $L_i \in S$ by (12) and hence $\{L_{k+1}, \ldots, L_l\} \subseteq S$ by (14). Then, the corresponding rule (16) exists in Π^S and S satisfies it; (b) If $\lambda_i \notin \Sigma$ but $\lambda_j \in \Sigma$ for some $1 \leq i \leq k$ and $k+1 \leq j \leq l$, then $L_j \notin S$ by (15). Then, there is no corresponding rule (16) in Π^S. In either case, S satisfies all rules of Π^S.

Suppose that there is a set S' of literals from Lit_Π such that (i) $S' \subset S$ and (ii) S' satisfies the rules of Π^S. Then, two conditions (i) and (ii) are satisfied only if there is a rule (16) such that $\{L_{l+1}, \ldots, L_m\} \subset S'$ and for some two literals L_{i1} and L_{i2} ($1 \leq i1, i2 \leq k$, $i1 \neq i2$) $L_{i1} \in S'$ but $L_{i2} \in S \setminus S'$. Without loss of generality, we can assume that just one such rule exists in Π^S. Since S and S' contain $L_{k+1}, \ldots, L_l$ in the corresponding rule (10) in Π, $\lambda_{i1}, \lambda_{i2} \in \Sigma$ by (13). Let $\Sigma' = \Sigma \setminus \{L_{i2}, \lambda_{i2}\}$. Then, Σ' satisfies all the rules (17,12,13,15) existing in $edp(\Pi)^\Sigma$. This contradicts the fact that Σ is an answer set of $edp(\Pi)^\Sigma$. Therefore, S is an answer set of Π^S, and hence an answer set of Π. □

We thus see that any GEDP can be reduced to an EDP by eliminating positive *not*. The fact that non-minimal answer sets of GEDPs can be expressed by answer sets of EDPs that must be minimal is a somewhat surprising and unexpected result. The reason why this reduction is possible is that the newly introduced atoms λ_i's have the effect to distinguish each positive *not*, and each answer set of $edp(\Pi)$ becomes minimal by the existence of these new atoms.

Example 4.1 Suppose that a GEDP is given as

$$\Pi = \{\ p \mid not\, q \leftarrow ,\quad q \mid not\, p \leftarrow\ \}.$$

The answer sets of Π are $\{\{p, q\}, \emptyset\}$. Correspondingly,

$$\begin{aligned} edp(\Pi) = \{\ & \lambda_1 \mid \lambda_2 \leftarrow ,\ \ \lambda_3 \mid \lambda_4 \leftarrow , \\ & p \leftarrow \lambda_1,\ \ \lambda_1 \leftarrow p, q,\ \ \leftarrow \lambda_1, not\, q,\ \ \leftarrow \lambda_2, q, \\ & q \leftarrow \lambda_3,\ \ \lambda_3 \leftarrow q, p,\ \ \leftarrow \lambda_3, not\, p,\ \ \leftarrow \lambda_4, p\ \} \end{aligned}$$

has the answer sets $\{\{\lambda_1, \lambda_3, p, q\}, \{\lambda_2, \lambda_4\}\}$.

4.2 Complexity Results

We are now ready to give the complexity results for GEDPs. Since the class of GEDPs includes the class of EDPs and we have shown a polynomial-time transformation from a GEDP to an EDP, the next result follows immediately from the complexity results of EDPs given by Eiter and Gottlob (1993).

Theorem 4.2 *Let Π be a finite propositional GEDP, and L a literal.*
(1) *Deciding the existence of an answer set of Π is Σ_2^P-complete.*
(2) *Deciding whether L is true in some answer set of Π is Σ_2^P-complete.*
(3) *Deciding whether L is true in all answer sets of Π is Π_2^P-complete.* □

Theorem 4.2 demonstrates that allowing positive *not* does not increase the computational complexity of the answer set semantics. Eiter and Gottlob also show that the complexity results for EDPs apply to EDPs without classical negation $\neg$ as well. Therefore, GEDPs are in the same complexity class as normal disjunctive programs. Furthermore, Theorem 4.2 (2) also applies to the minimal model semantics for positive disjunctive programs. This observation leads us to a further translation in Section 4.3.

Ben-Eliyahu and Dechter (1992) have shown the (co-)NP-completeness of a restricted class of EDPs. According to their notations, a *dependency graph* of a ground EDP P is a directed graph in which its nodes are literals in P and there is an edge from L to L' iff there is a rule in P such that L appears in the body and L' appears in the head of the rule. An EDP is

head-cycle free if its dependency graph contains no directed cycle that goes through two different literals in the head of the same disjunctive rule. Then, the three problems in Theorem 4.2 for propositional head-cycle free EDPs are reducible to testing satisfiability or provability of propositional formulas in polynomial-time (Ben-Eliyahu and Dechter, 1992). Here, we show such a reduction of complexity results is also possible for a restricted class of GEDPs, by providing a generalization of their results. Given a ground GEDP Π, its *dependency graph* G_Π is defined in the same way as that of an EDP except that an additional edge from L to L' is added for each $not\, L$ and L' in the head of the same rule. Thus, while each $not\, L$ in bodies is ignored, each $not\, L$ in heads constructs an edge in G_Π. A GEDP Π is *head-cycle free* if G_Π contains no directed cycle that goes through two literals L_{i1}, L_{i2} $(1 \leq i1, i2 \leq k,\ L_{i1} \neq L_{i2})$ in any ground rule (10) from Π. The class of head-cycle free GEDPs obviously includes the class of head-cycle free EDPs and the class of extended logic programs, and includes the class of GEDPs each of whose rule permits in the head at most one L' but any number of $not\, L$'s.

Lemma 4.3 *Let Π be a GEDP. Π is head-cycle free iff $edp(\Pi)$ is head-cycle free.*

Proof: An edge from L_j to L_i for $j = k+1, \ldots, l$ and $i = 1, \ldots, k$ in the same rule (10) is in G_Π iff a path from L_j to L_i through rules (13) and (12) is in $G_{edp(\Pi)}$. Then, each directed path from L to L' in G_Π is contained in $G_{edp(\Pi)}$, and vice versa. Hence, any two literals L_{i1}, L_{i2} $(1 \leq i1, i2 \leq k)$ in the same rule (10) are contained in a cycle in G_Π iff the literals $\lambda_{i1}, \lambda_{i2}$ in the corresponding rule (11) are contained in a cycle in $G_{edp(\Pi)}$. □

The next result follows from Theorem 4.1, Lemma 4.3 and complexity results of head-cycle free EDPs by (Ben-Eliyahu and Dechter, 1992).

Theorem 4.4 *Let Π be a finite propositional head-cycle free GEDP, and L a literal.*
(1) *Deciding the existence of an answer set of Π is NP-complete.*
(2) *Deciding whether L is true in some answer set of Π is NP-complete.*
(3) *Deciding whether L is true in all answer sets of Π is co-NP-complete.* □

Note that the class of head-cycle free GEDPs includes, as a special case, the class of programs $P \cup abd(\Gamma)$ obtained from abductive programs $\langle P, \Gamma \rangle$ where both P and Γ are extended logic programs (see Section 3.3). This fact and results in Section 3 imply that computational problems for abductive normal logic programs (Kakas and Mancarella, 1990), knowledge systems (Inoue, 1991), and the possible model semantics for normal disjunctive programs (Sakama and Inoue, 1993b) have all the same complexity results as in Theorem 4.4. These results are also stated in (Sakama and Inoue, 1994) based on translations of such programs into normal logic programs.

4.3 Computing Answer Sets of GEDP

To compute the answer set semantics for any GEDP Π, we can apply any proof procedure for extended disjunctive programs to the EDP $edp(\Pi)$ obtained in Section 4.1. To this end, a bottom-up proof procedure for EDPs has been proposed by Inoue *et al.* (1992) to compute answer sets of EDPs using model generation techniques. Here, we present an essence of the method of (Inoue *et al.*, 1992). First, each EDP P is converted into its *positive form* P^+, which is obtained from P by replacing each negative literal $\neg L$ with a new atom $\mathsf{-}L$. Note that P^+ is a normal disjunctive program. We also denote the positive form of a set S of literals as S^+. Next, P^+ is translated into the set $fo(P^+)$ of first-order formulas by completely eliminating *not* as follows. For each rule in P^+ of the form:

$$L_1 \mid \ldots \mid L_k \leftarrow L_{k+1}, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n \tag{18}$$

where L_i's are atoms, $fo(P^+)$ contains the formula:

$$L_{k+1} \wedge \ldots \wedge L_m \supset H_1 \vee \ldots \vee H_k \vee \mathsf{K}L_{m+1} \vee \ldots \vee \mathsf{K}L_n\,, \tag{19}$$

where

$$H_i \equiv L_i \wedge \mathsf{-K}L_{m+1} \wedge \ldots \wedge \mathsf{-K}L_n \quad (i = 1, \ldots, k),$$

and $fo(P^+)$ contains the formulas:

$$\neg(L \wedge \mathsf{-K}L) \quad \text{for each } L \in {Lit_P}^+, \tag{20}$$
$$\neg(L \wedge \mathsf{-}L) \quad \text{for each pair } L, \mathsf{-}L \in {Lit_P}^+. \tag{21}$$

Here, $\mathsf{K}L$ is a new atom which denotes L *should be true*, and $\mathsf{-K}L$ is the positive form of $\neg\mathsf{K}L$. Now, let I be an Herbrand interpretation of $fo(P^+)$, i.e., a set of ground atoms in the language of $fo(P^+)$. We say that I *satisfies the stability condition* if it holds that

$$\mathsf{K}L \in I \ \text{ implies } \ L \in I \ \text{ for every atom } L \in {Lit_P}^+.$$

Lemma 4.5 *(Inoue* et al.*, 1992) Let P be an EDP, and $S \subseteq Lit_P$. S is a consistent answer set of P iff M is a minimal Herbrand model of $fo(P^+)$ such that $S^+ = M \cap {Lit_P}^+$ and M satisfies the stability condition.* □

The next theorem, which follows from Theorem 4.1 and Lemma 4.5, completely characterizes the consistent answer sets of a GEDP in terms of the above first-order translation.[4]

Theorem 4.6 *Let* Π *be any GEDP, and* $S \subseteq Lit_\Pi$. *S is a consistent answer set of* Π *iff M is a minimal Herbrand model of* $fo(edp(\Pi)^+)$ *such that* $S^+ = M \cap Lit_\Pi{}^+$ *and M satisfies the stability condition.* □

It is well known that for positive disjunctive programs minimal models coincide with answer sets. Then, as the formula (19) can be identified with the rule

$$H_1 \mid \ldots \mid H_k \mid \mathsf{K}L_{m+1} \mid \ldots \mid \mathsf{K}L_n \leftarrow L_{k+1}, \ldots, L_m\,,$$

the set $fo(P^+)$ can be viewed as a PDP. We thus now have a polynomial-time transformation from GEDPs to PDPs. Hence, to obtain answer sets of GEDPs, any procedure to compute minimal models of PDPs can be applied as well. There are many techniques for this computation such as (Bell *et al.*, 1992; Inoue *et al.*, 1992; Fernandez and Minker, 1992). In particular, our transformation is suitable for applying a bottom-up model generation procedure to compute answer sets of function-free, range-restricted GEDPs. Since we have characterized abductive programs as GEDPs in Section 3, abduction can also be computed by model generation procedures. Inoue *et al.* (1993) have developed such a parallel abductive procedure, and Inoue and Sakama (1993) have given a fixpoint semantics that accounts for the correctness of such bottom-up procedures using a similar transformation.

Example 4.2 The abductive program $\langle\, P', \Gamma \,\rangle$ given in Examples 3.1 and 3.2 is now transformed into $fo(P'_\Gamma)$ that consists of the propositional formulas

$$\begin{gathered} r \wedge b \supset (p \wedge \mathsf{-K}q) \vee \mathsf{K}q, \quad a \supset q, \quad r, \quad \mathsf{K}p, \\ \lambda_1 \vee \lambda_2, \quad \lambda_1 \equiv a, \quad \lambda_1 \supset \mathsf{K}a, \quad \neg(\lambda_2 \wedge a), \\ \lambda_3 \vee \lambda_4, \quad \lambda_3 \equiv b, \quad \lambda_3 \supset \mathsf{K}b, \quad \neg(\lambda_4 \wedge b), \end{gathered}$$

and the schema (20).[5] Then, there are five minimal models of $fo(P'_\Gamma)$:

$$\begin{aligned} M_1 &= \{r, \mathsf{K}p, \lambda_1, a, \mathsf{K}a, q, \lambda_3, b, \mathsf{K}b, \mathsf{K}q\}, \\ M_2 &= \{r, \mathsf{K}p, \lambda_1, a, \mathsf{K}a, q, \lambda_4\}, \\ M_3 &= \{r, \mathsf{K}p, \lambda_2, \lambda_3, b, \mathsf{K}b, p, \mathsf{-K}q\}, \\ M_4 &= \{r, \mathsf{K}p, \lambda_2, \lambda_3, b, \mathsf{K}b, \mathsf{K}q\}, \\ M_5 &= \{r, \mathsf{K}p, \lambda_2, \lambda_4\}. \end{aligned}$$

Among these, only M_3 satisfies the stability condition, and corresponds to the belief set $\{r, p, b\}$ of $\langle\, P', \Gamma \,\rangle$.

[4] Although Theorem 4.6 does not cover the contradictory answer set of Π, the methods used in (Inoue *et al.*, 1992) can be applied to identify the answer set Lit_Π.

[5] When an EDP P is an NDP, $P^+ = P$ holds and $fo(P)$ need not include the schema (21).

5 Relation to Autoepistemic Logic

Recall that the class of GEDPs is the "logic programming" fragment of propositional MBNF (Lifschitz, 1992). The embedding of the rule (2)

$$\begin{aligned} L_1 \mid \ldots \mid L_k \mid \mathit{not}\, L_{k+1} \mid \ldots \mid \mathit{not}\, L_l \\ \leftarrow L_{l+1}, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n \end{aligned}$$

into MBNF is given by Lifschitz and Woo (1992) as the formula

$$\begin{aligned} \mathrm{B}L_{l+1} \wedge \ldots \wedge \mathrm{B}L_m \wedge \mathit{not}\, L_{m+1} \wedge \ldots \wedge \mathit{not}\, L_n \\ \supset \mathrm{B}L_1 \vee \ldots \vee \mathrm{B}L_k \vee \mathit{not}\, L_{k+1} \vee \ldots \vee \mathit{not}\, L_l\,. \end{aligned}$$

Besides MBNF, there are many nonmonotonic formalisms to which EDPs can be embedded. Gelfond *et al.* (1991) use their disjunctive default logic, and Sakama and Inoue (1993a) show transformations into Reiter's default logic, Moore's autoepistemic logic and McCarthy's circumscription. Since we have presented the reduction of GEDPs to EDPs, these previous results can be directly applied to embed GEDPs into such nonmonotonic formalisms, and they are all well defined. Although these results are all correct, one often wants to see a stronger result such that the logical closure of an answer set is exactly the same as an *extension* of a nonmonotonic formalism and that the set of literals true in an extension is exactly an answer set. In such an extension, the introduction of new literals like λ_i's should be avoided. Then, those formalisms that obey the principle of minimality such as (disjunctive) default logic and circumscription are rejected for this purpose. With this regard, the remaining candidate is autoepistemic logic. Lifschitz and Schwarz (1993) and Chen (1993) have independently provided the correct embedding of EDPs into autoepistemic logic. Moreover, both results are proved in a way applicable to a more general class of programs including consistent PL-theories of (Lifschitz and Woo, 1992). Here, we can take advantage of their proofs.[6]

Given a GEDP Π, its *autoepistemic translation* $ae(\Pi)$ is defined as follows: Each rule of the form (2) in Π is transformed into the following formula in $ae(\Pi)$:

$$\begin{aligned} (BL_{l+1} \wedge L_{l+1}) \wedge \ldots \wedge (BL_m \wedge L_m) \\ \wedge \neg BL_{m+1} \wedge \ldots \wedge \neg BL_n \\ \supset \quad (BL_1 \wedge L_1) \vee \ldots \vee (BL_k \wedge L_k) \\ \vee \neg BL_{k+1} \vee \ldots \vee \neg BL_l\,. \end{aligned} \tag{22}$$

[6] Marek and Truszczyński (1993) also show a different translation of EDPs into *reflexive autoepistemic logic* (Schwarz, 1991). Lifschitz and Schwarz (1993) further prove that reflexive autoepistemic logic can be used for the embedding of consistent PL-theories.

Recall that given a set A of formulas (called a *premise set*) in autoepistemic logic, T is a *stable expansion* of A iff

$$T = cons(A \cup \{B\varphi \mid \varphi \in T\} \cup \{\neg B\varphi \mid \varphi \notin T\}),$$

where $cons(X)$ denotes the set of propositional consequences of X. It is also well known that for each set F of objective formulas, there is a unique stable set $E(F)$ containing F such that the objective formulas in $E(F)$ are exactly the same as those in $cons(F)$. If a premise set A contains only objective formulas, then $E(A)$ is a unique stable expansion of A. Then, the next result follows from (Lifschitz and Schwarz, 1993, Main Theorem).

Theorem 5.1 *Let Π be a consistent GEDP, and S a set of literals. S is an answer set of Π iff $E(S)$ is a stable expansion of $ae(\Pi)$.* □

The autoepistemic translation $ae(\Pi)$ can be simplified for some class of GEDPs. When Π is a GEDP that consists of rules of the form

$$\begin{aligned} &A_1 \mid not\, A_2 \mid \ldots \mid not\, A_l \\ &\quad \leftarrow A_{l+1}, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n\,, \end{aligned} \tag{23}$$

where $0 \le l \le m \le n$ (A_1 may be empty) and A_i's are atoms, each $(BA_i \wedge A_i)$ for $i = 1$ and $i = l+1, \ldots, m$ in $ae(\Pi)$ can be replaced simply with A_i as

$$\begin{aligned} &A_{l+1} \wedge \ldots \wedge A_m \wedge \neg BA_{m+1} \wedge \ldots \wedge \neg BA_n \\ &\quad \supset A_1 \vee \neg BA_2 \vee \ldots \vee \neg BA_l\,. \end{aligned} \tag{24}$$

Note that this class of GEDPs is a subset of the class of head-cycle free GEDPs, and includes the class of normal logic programs and programs P_Γ that are translated from abductive normal logic programs $\langle P, \Gamma \rangle$. Let us denote as $ae_n(\Pi)$ the set of autoepistemic formulas obtained from a GEDP Π by replacing each rule of the form (23) with (24). An essential difference between $ae(\Pi)$ and $ae_n(\Pi)$ for a set Π of rules of the form (23) is that, while ae_n may map two different programs with the same answer sets into two autoepistemic theories with different stable expansions, the stable expansions of $ae(\Pi)$ are uniquely determined by the answer sets of Π (Lifschitz and Schwarz, 1993). Nevertheless, we have the following one-to-one correspondence between the answer sets of Π and the stable expansions of $ae_n(\Pi)$.

Corollary 5.2 *Let Π be a consistent GEDP such that Π is a set of rules of the form (23), and S a set of atoms. S is an answer set of Π iff S is the set of objective atoms true in a stable expansion of $ae_n(\Pi)$.*

Proof: Suppose that S is an answer set of Π. By Theorem 5.1, there is a stable expansion E of $ae(\Pi)$ such that $S = E \cap At$ where At is the set of atoms occurring in Π and that

$$E = cons(S \cup \{B\varphi \mid \varphi \in E\} \cup \{\neg B\varphi \mid \varphi \notin E\}).$$

The set $\{B\varphi \mid \varphi \in E\}$ includes BA for each $A \in S$. In the presence of these subjective atoms, all the objective atoms S in E also follows from some stable expansion E' of $ae_n(\Pi)$, and vice versa. Hence, $S = E' \cap At$. The converse direction can also be shown in the same manner. □

The above corollary can also be applied to the embedding of the possible model semantics for a normal disjunctive program P since each rule in the translated GEDP $pm(P)$ is in the form (23).

Theorem 5.3 *Let P be a consistent NDP that consists of rules of the form*

$$A_1 \mid \ldots \mid A_k \leftarrow A_{k+1}, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n.$$

A set S of atoms is a possible model of P iff S is the set of objective atoms true in a stable expansion of the set of formulas obtained by translating each above rule in P into the formula

$$\begin{aligned} &A_{k+1} \wedge \ldots \wedge A_m \wedge \neg BA_{m+1} \wedge \ldots \wedge \neg BA_n \\ &\quad \supset (A_1 \vee \neg BA_1) \wedge \ldots \wedge (A_k \vee \neg BA_k) \\ &\qquad \wedge (BA_1 \vee \ldots \vee BA_k)\,. \end{aligned}$$

Proof: The translated formula is equivalent to the conjunction of the ae_n translation of rules (8) and (9) in $pm(P)$. Then, the result follows from Theorem 3.4 and Corollary 5.2. □

Now, let us look again at the embedding of abduction into GEDPs given in Theorem 3.1. The rule (3)

$$\gamma \mid not\, \gamma \leftarrow$$

is translated into

$$(B\gamma \wedge \gamma) \vee \neg B\gamma$$

by the autoepistemic translation, which is then equivalent to

$$B\gamma \supset \gamma\,. \tag{25}$$

The set consisting of the formula (25) produces two stable expansions, one containing γ and $B\gamma$, the other containing $\neg B\gamma$ but neither γ nor $\neg\gamma$. Historically, the first expansion has been regarded as anomalous since the belief of γ is based solely on the assumption that γ is believed with no other support (Konolige, 1988).

However, this situation is most welcome for abduction. The fact that the formula (25) is the archetype to generate hypotheses strongly justifies the correctness of our use of positive *not* in the corresponding rule (3).

Finally, from the relationship between PL-theories and autoepistemic logic, we obtain the next result.

Theorem 5.4 *Let T be a set of propositional combinations of* protected literals*, i.e., formulas of the form* BL *or not L where L is a literal. Deciding whether T has an MBNF-model is Σ_2^P-complete.*

Proof: Σ_2^P-hardness follows from Theorem 4.2 (1) and the embedding of GEDPs into MBNF. Membership in Σ_2^P is shown by the complexity result for autoepistemic logic (Gottlob, 1992) and a polynomial-time translation of an MBNF formula of the above form into autoepistemic logic (Lifschitz and Schwarz, 1993), which replaces each protected literal BL with $L \wedge BL$ and each protected literal *not* L with $\neg BL$. □

6 Discussion

1. Brewka and Konolige (1993) give another semantics for GEDPs which is different from the answer set semantics in this paper. They allow positive *not* in a program but still obey the principle of minimality. Consequently, their semantics can never represent non-minimal canonical models and its relationship to autoepistemic logic must be different from ours. In this respect, they suggest the use of *moderately grounded expansions* (Konolige, 1988) for the embedding. However, moderately grounded expansions are of no use to characterize the *minimal* answer sets of GEDPs. Instead, *parsimonious stable expansions* (Eiter and Gottlob, 1992) appropriately characterize the minimal answer sets.[7] For example, the GEDP

$$\Pi = \{\ p \mid not\, p \leftarrow ,\quad q \leftarrow p,\quad \leftarrow not\, q\ \}.$$

has the unique (and hence minimal) answer set $\{p, q\}$, but its autoepistemic translation

$$ae_n(\Pi) = \{\ Bp \supset p,\quad p \supset q,\quad Bq\ \}.$$

has no moderately grounded expansion. In fact, $E(\{p, q\})$ is not a minimal stable set that includes $ae_n(\Pi)$ since $E(\{q\})$ is a stable set containing $ae_n(\Pi)$ and its objective part is smaller than that of $E(\{p, q\})$. On the other hand, $E(\{p, q\})$ is the unique parsimonious stable expansion. In general, the next result follows from Theorem 5.1 and the definition of parsimonious stable expansions.

Corollary 6.1 *Let Π be a consistent GEDP, and S a set of literals. S is a minimal answer set of Π iff $E(S)$ is a parsimonious stable expansion of $ae(\Pi)$.* □

Recall that our answer set semantics for GEDPs is characterized by stable expansions of the translated autoepistemic theories. From the complexity viewpoint, Eiter and Gottlob have shown that deciding whether an objective formula belongs to some parsimoniously grounded expansion of an autoepistemic theory is Σ_3^P-complete in general (Eiter and Gottlob, 1992), while the same problem for some stable expansion is Σ_2^P-complete (Gottlob, 1992). From their results, it is conjectured that computing with a minimal answer set of a GEDP is strictly harder than computing with its any answer set unless the polynomial hierarchy collapses.

2. An interesting property of the rule (3) $\gamma \mid not\, \gamma \leftarrow$ is that it is *valid* in the sense that every answer set satisfies it, that is, γ is either contained or not contained in it. In autoepistemic logic, the corresponding formula (25) is always contained in any stable expansion. However, the modal axiom schema of the same form

$$T: \quad B\varphi \supset \varphi$$

cannot be put into the premise set without changing its stable expansions (Moore, 1985). Similarly, adding the rule $L \mid not\, L \leftarrow$ to a program allows the literal L to be sanctioned that otherwise would not be, but this may cause literals that are entailed by the program to decrease since the number of answer sets increases. For example, q is entailed by

$$\{\ q \leftarrow not\, p\ \}$$

but once $p \mid not\, p \leftarrow$ is adopted q is no longer entailed. This property is effectively used in Example 3.4 for cautious closed world reasoning. Sometimes such an addition of valid rules may make an incoherent program to get an answer set. For example,

$$\{\ q \leftarrow not\, p,\quad \neg q \leftarrow\ \}$$

has no answer set, but with the rule $p \mid not\, p \leftarrow$ it obtains the answer set $\{\neg q, p\}$. The schema T in autoepistemic logic and the rule $L \mid not\, L \leftarrow$ in GEDPs can thus be applied to various domains other than abduction such as contradiction resolution, meta-programming and reflection (Konolige, 1992).

[7] A stable expansion of a premise set A is *moderately grounded* if its objective part is not smaller than the objective part of any other stable set that includes A. A stable expansion of A is *parsimonious* if its objective part is not smaller than the objective part of any other stable expansion of A. Note that each moderately grounded expansion is parsimonious but the converse does not necessarily holds.

3. Gelfond gives another cautious semantics for the closed world assumption in order to treat Example 3.4 properly by introducing the concept of *strong introspection* (Gelfond, 1991). However, unlike Theorem 3.4 for our possible model semantics, this concept cannot be embedded into MBNF (Lifschitz, 1992).

4. Positive *not* can be used to represent conditional rules. For example,

$$p \mid not\, q \leftarrow$$

can be viewed as a conditional formula which states that p is true if q is true. In this sense, the rule is similar to

$$p \leftarrow q\,.$$

In fact, there is a case that the former rule can be replaced with the latter rule by shifting positive *not* into the body. However, once a "deadlock" loop is constructed with these conditional formulas as Example 4.1, we will have two alternative answer sets, one including every element of the loop and the other including nothing in the loop. We expect that this kind of conditional rules may have interesting applications. On the other hand, a rule having no literals but positive *not* in its head can be used to represent *integrity constraints*. In this case,

$$not\, p \leftarrow q$$

has exactly the same effect as the rule

$$\leftarrow p,\, q\,.$$

It may also be interesting to investigate for what class of programs such shifting of positive *not* is possible.

5. Concerning generalizations of the results in this paper, there are a couple of interesting topics. First, the first-order abductive framework $\langle T, \Gamma \rangle$, where T and Γ are sets of first-order formulas, can be translated into the MBNF bimodal formula

$$\bigwedge_{F \in T} \mathrm{B}F \wedge \bigwedge_{G \in \Gamma} (\mathrm{B}G \vee not\, G)\,,$$

or into the set of formulas in autoepistemic logic

$$T \cup \{\, BG \supset G \mid G \in \Gamma \,\}\,.$$

Secondly, Eiter *et al.* recently proposed a new nonmonotonic formalism called *curbing* (Eiter *et al.*, 1993) which interprets disjunctions inclusively. Since their *good models* are not necessarily minimal models, it is interesting to see whether MBNF can express curbing or not. In the context of logic programming, it turns out that there is a close relationship between good models and possible models. Thirdly, the computational complexity of propositional MBNF and the possibility of embedding MBNF into autoepistemic logic in general are both left open.

7 Conclusion

This paper has provided a number of new results in the class of general extended disjunctive programs, i.e., disjunctive programs which permit negation as failure and classical negation both positively and negatively. In particular, we have shown in this paper

- an embedding of abductive programs into GEDPs,
- a translation the possible model semantics for NDPs into the answer set semantics for GEDPs,
- a complexity characterization of GEDPs based on a translation of GEDPs into EDPs,
- a computational method for GEDPs based on a translation of GEDPs into PDPs, and
- a relationship between GEDPs and autoepistemic logic.

With these results, we can conclude that the concept of positive occurrences of negation as failure is a useful tool for representing knowledge in various domains in which the principle of minimality is too strong.

The class of GEDPs is a natural extension of previously proposed logic programs, and is an "ultimate" extension with negation as failure. A number of new nonmonotonic logics have recently been proposed in order to cover the growing expressiveness of logic programming. These logics include disjunctive default logic, reflexive autoepistemic logic, and MBNF. On the other hand, existing nonmonotonic formalisms have been tested in various ways whether they are better suited than others for applications to the semantics of logic programming. Circumscription, autoepistemic logic, default logic, and their variants have competed with each other for victory, and winners frequently changed in this decade. We suggest in this paper that, among all these formalisms, autoepistemic logic is one of the best because of its "non-minimal" nature. That is, abduction and inclusive disjunctions in knowledge representation are naturally expressed by this unique feature of autoepistemic logic. Introspective natures involved by autoepistemic logic enable us to believe a certain proposition either from the lack of belief in another proposition or from no additional precondition. These properties can completely describe the meanings of negative and positive occurrences of negation as failure in logic programming.

Acknowledgements

We thank Thomas Eiter and George Gottlob for their comments on complexity results in this paper.

References

C. Bell, A. Nerode, R.T. Ng and V.S. Subrahmanian (1992). Implementing deductive databases by linear programming. In: *Proc. 11th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems*, San Diego, CA, 283–292.

R. Ben-Eliyahu and R. Dechter (1992). Propositional semantics for disjunctive logic programs. In: *Proc. Joint Int. Conf. Symp. Logic Programming*, Washington, D.C., 813–827, MIT Press.

A. Beringer and T. Schaub (1993). Minimal belief and negation as failure: a feasible approach. In: *Proc. AAAI-93*, Washington, D.C., 400–405.

G. Brewka and K. Konolige (1993). An abductive framework for general logic programs and other nonmonotonic systems. In: *Proc. IJCAI-93*, Chambery, 9–15.

E.P.F. Chan (1993). A possible world semantics for disjunctive databases. *IEEE Trans. Data and Knowledge Engineering*, 5(2):282–292.

J. Chen (1993). Minimal knowledge + negation as failure = only knowing (sometimes). In: *Proc. 2nd Int. Workshop Logic Programming and Non-monotonic Reasoning*, Lisbon, 132–150, MIT Press.

T. Eiter and G. Gottlob (1992). Reasoning with parsimonious and moderately grounded expansions. *Fundamenta Informaticae*, 17(1,2):31–53.

T. Eiter and G. Gottlob (1993). Complexity results for disjunctive logic programming and application to nonmonotonic logics, In: *Proc. 1993 Int. Logic Programming Symp.*, Vancouver, 266–278, MIT Press.

T. Eiter, G. Gottlob and Y. Gurevich (1993). Curb your theory!—a circumscriptive approach for inclusive interpretation of disjunctive information. In: *Proc. IJCAI-93*, Chambery, 634–639.

J.A. Fernandez and J. Minker (1992). Disjunctive deductive databases. In: *Proc. Int. Conf. Logic Programming and Automated Reasoning*, Lecture Notes in Artificial Intelligence, 624, 332–356, Springer-Verlag.

M. Gelfond (1991). Strong introspection. In: *Proc. AAAI-91*, Anaheim, CA, 386–391.

M. Gelfond and V. Lifschitz (1991). Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385.

M. Gelfond, V. Lifschitz, H. Przymusinska and M. Truszczynski (1991). Disjunctive defaults. In: *Proc. KR '91*, Cambridge, MA, 230–237, Morgan Kaufmann.

G. Gottlob (1992). Complexity results for nonmonotonic logics. *J. Logic and Computation*, 2(3):397–425.

K. Inoue (1991). Extended logic programs with default assumptions. In: *Proc. 8th Int. Conf. Logic Programming*, Paris, 490–504. An extended version: Hypothetical reasoning in logic programs. *J. Logic Programming*, 18, to appear, 1994.

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

K. Inoue, Y. Ohta, R. Hasegawa and M. Nakashima (1993). Bottom-up abduction by model generation. In: *Proc. IJCAI-93*, Chambery, 102–108.

K. Inoue and C. Sakama (1993). Transforming abductive logic programs to disjunctive programs. In: *Proc. 10th Int. Conf. Logic Programming*, Budapest, 335–353, MIT Press.

A.C. Kakas and P. Mancarella (1990). Generalized stable models: a semantics for abduction. In: *Proc. ECAI-90*, Stockholm, 385–391.

A.C. Kakas, R.A. Kowalski and F. Toni (1992). Abductive logic programming. *J. Logic and Computation*, 2(6):719–770.

K. Konolige (1988). On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35:343–382.

K. Konolige (1992). An autoepistemic analysis of metalevel reasoning in logic programming. In: *Proc. 3rd Int. Workshop Meta-Programming in Logic*, Lecture Notes in Computer Science, 649, 26–48, Springer-Verlag.

V. Lifschitz (1992). Minimal belief and negation as failure. An earlier version: Nonmonotonic databases and epistemic queries. In: *Proc. IJCAI-91*, Sydney, 381–386, 1991.

V. Lifschitz and G. Schwarz (1993). Extended logic programs as autoepistemic theories. In: *Proc. 2nd Int. Workshop Logic Programming and Non-monotonic Reasoning*, Lisbon, 101–114, MIT Press.

V. Lifschitz and T.Y.C. Woo (1992). Answer sets in general nonmonotonic reasoning (preliminary report). In: *Proc. KR '92*, Boston, MA, 603–614, Morgan Kaufmann.

W. Marek and M. Truszczynski (1993). Reflexive autoepistemic logic and logic programming. In: *Proc.*

2nd Int. Workshop Logic Programming and Non-monotonic Reasoning, Lisbon, 115–131, MIT Press.

R.C. Moore (1985). Semantical considerations on non-monotonic logic. *Artificial Intelligence*, 25:75–94.

C. Sakama (1989). Possible model semantics for disjunctive databases. In: *Proc. 1st Int. Conf. Deductive and Object-Oriented Databases*, Kyoto, 337–351.

C. Sakama and K. Inoue (1993a). Relating disjunctive logic programs to default theories. In: *Proc. 2nd Int. Workshop Logic Programming and Non-monotonic Reasoning*, Lisbon, 266–282, MIT Press.

C. Sakama and K. Inoue (1993b). Negation in disjunctive logic programs. In: *Proc. 10th Int. Conf. Logic Programming*, Budapest, 703–719, MIT Press.

C. Sakama and K. Inoue (1994). On the equivalence between disjunctive and abductive logic programs. In: *Proc. 11th Int. Conf. Logic Programming*, Santa Margherita Ligure, to appear, MIT Press.

J.S. Schlipf (1992). Formalizing a logic for logic programming. *Ann. Mathematics and Artificial Intelligence*, 5:279–302.

G.F. Schwarz (1991). Autoepistemic logic of knowledge. In: *Proc. 1st Int. Workshop Logic Programming and Non-monotonic Reasoning*, Washington, D.C., 260–274, MIT Press.