TR-605

Computing Abduction by Using the TMS

by
K. Satoh & N. Iwayama

November, 1990

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Computing Abduction by Using the TMS

Ken Satoh, Noboru Iwayama
Institute for New Generation Computer Technology
4-28 Mita 1-Chome Minato-ku, Tokyo 108 Japan
email: ksatoh@icot.or.jp

November 2, 1990

Abstract

We present a method to compute abduction in [Eshghi89, Kakas90a, Kakas90b]. We translate an abductive framework into a general logic program with integrity constraint and show the correspondence between generalized stable models defined in [Kakas90a, Kakas90b] and stable models for the translation of abductive framework. Then, we show a procedure to compute stable models for a general logic program with integrity constraint. This procedure is based on a procedure of calculating grounded extension of Doyle's TMS [Doyle79, Satoh90] and can be regarded as an extension of a well-founded procedure to calculate stable models for a general logic program without integrity constraint[Fages90, Sacca90].

1 Introduction

In this paper, we present a method of calculating abduction in [Eshghi89, Kakas90a, Kakas90b]. Recent researches have revealed that abduction plays important role in artificial intelligence, as stated in [Eshghi89]. Various researchers have studied abduction in terms of logic programming framework [Eshghi89, Kakas90a, Kakas90b] and shown relationships with non-monotonic reasoning framework such as negation as failure, truth maintenance system and autoepistemic logic.

[Kakas90a] shows the abduction procedure to calculate hypotheses to explain a given observation. This procedure extends the top-down procedure in [Eshghi89] in order to manipulate arbitrary hypothesis. However, this procedure inherits the problem of the previous procedure of [Eshghi89] that correctness does not hold in general for logic programs with recursion as shown in [Eshghi89, p. 251].

In this paper, we give an abduction procedure which is correct in any general logic program. To do that, we first give a translation of abductive framework to a general logic prorgram with integrity constraint and show that a stable model [Gelfond88] for the logic program coincides with some general stable models defined in [Kakas90b]. We also show that explanation defined in abductive framework coincides with the stable models for the translated logic program plus a special integrity constraint.

Then, we provide a bottom-up procedure which calculates a stable models for a general logic program with integrity constraint. This procedure is based on a procedure calculating grounded extension of TMS [Doyle79, Satoh90] and can be regarded as an extension of well-founded bottom-up procedure for calculating stable model for logic programs *without* integrity constraint [Fages90, Sacca90].

One might think that it is sufficient to calculate stable models for a general logic programs by the method of [Fages90, Sacca90] and then select stable models which satisfy integrity constraint. Theoretically speaking, this naive process does not cause any problem. Practically speaking, however, we can save search space by excluding unsatisfiable stable models by checking integrity constraint during computing stable models. Our procedure not only checks integrity constraint dynamically but also uses integrity constraint actively to derive some facts.

2 Translating Abductive Framework to Logic Program with Integrity Constraint

We follow the definition in [Kakas90a, Kakas90b] but restrict ourselves to considering propositional case. If we consider predicate case, we change it into ground logic programs by instantiating every variable to an element of Herbrand universe of considered logic program to obtain propositional

program.

Firstly, we define general logic program and integrity constraint.

Definition 1 *Let A be a proposition symbol, and $L_1, ..., L_m (m \geq 0)$ be propositional literals. A* general logic program *consists of (possibly countably infinite) rules of the form:*

$$A \leftarrow L_1, L_2, ..., L_m.$$

We call A the *head* of the rule and $L_1, ..., L_m$ the *body* of the rule. Let R be a rule. We denote the set of positive literals in the body of R as $pos(R)$ and the set of atoms which are obtained by removing negation symbol from negated atoms in the body of R as $neg(R)$.

Definition 2 *Let $L_1, ..., L_m (m \geq 0)$ be propositional literals. An* integrity constraint *consists of (possibly countably infinite) expressions of the form:*

$$\leftarrow L_1, L_2, ..., L_m.$$

Let C be an integrity constraint. We denote a set of positive literals in C as $pos(C)$ and a set of atoms which are obtained by removing negation symbol from negated atoms in C as $neg(C)$.

We also define stable models for a general logic program with integrity constraint as follows.

Definition 3 *A general logic program with integrity constraint be a pair (T, I) where T consists of rules and I consists of integrity constraints. A* stable model *for a general logic program with integrity constraint is a set of propositions M.*

1. *M is equal to the minimal model of T^M where T^M is obtained by the following operation from T. We say that M is the stable model of T.*
   (a) *Deleting every rule R from T that some $N \in neg(R)$ is in M*
   (b) *Deleting every negated atom in the remained rules.*
2. *For every $C \in I$, there is some $P \in pos(C)$ which is not in M or some $N \in neg(C)$ which is in M. We say that M* satisfies *or* does not violate *C and write as $M \models C$.*

This definition gives a stable model of T which satisfies any integrity constraints in C.

We follow the definition of abductive framework in [Kakas90a, Kakas90b].

Definition 4 *An* abductive framework *is a triple* $\langle T, A, I \rangle$ *where*

1. *A is a (possibly countably infinite) set of propositional symbols called* abducible propositions.
2. *T is a general logic program where no rule's head is equal to any element of A.*
3. *I is a set of integrity constraints.*

To be precise, we only consider a special form of integrity constraints whereas Kakas and Mancarella allow any closed formulas as integrity constraints. However, those constraints can be translated into our form of integrity constraints.

We follow the definition of generalized stable models and explanation with respect to $\langle T, A, I \rangle$ in [Kakas90a, Kakas90b].

Definition 5 *Let $\langle T, A, I \rangle$ be an abductive framework and Δ be a subset of A. A* generalized stable model *$M(\Delta)$ of $\langle T, A, I \rangle$ is a stable model of $\langle T \cup \Delta, I \rangle$ where $T \cup \Delta$ denotes $T \cup \{P \leftarrow \mid P \in \Delta\}$.*

Definition 6 *Let $\langle T, A, I \rangle$ be an abductive framework, and Q be an atom called* observation, *and Δ be a subset of A. Q has an abductive* explanation *with set of hypothesis Δ if and only if there exists a generalized stable model $M(\Delta)$ such that $Q \in M(\Delta)$.*

At first sight, an abducible has somewhat different character, but it turns out that there is a correspondence between an abductive framework and a logic program with integrity constraint as follows. We translate an abductive framework as follows.

For each abducible P in A, we introduce a new propositional symbol $\tilde{P}$ which is not used in $\langle T, A, I \rangle$. We denote a set of such new propositional symbols as A'. Then, we add the following rules in T for each abducible P in A and obtain a logic program with integrity constraint.

$$P \leftarrow \neg \tilde{P}$$

and

$$\tilde{P} \leftarrow \neg P.$$

We denote a set of all of the above rules for every abducible as $\Gamma(A)$. The first rule expresses that if $\tilde{P}$ is not believed, then P is believed and the second rule expresses that if P is not believed, then $\tilde{P}$ is believed. $\tilde{P}$, therefore, means that P is not believed. The first rule is used to assume P and the second rule is used not to assume P and is especially used to avoid contradiction if P is forced not to be believed by the program.

Then we can have the following correspondence between abductive framework and its translation.

Theorem 1 *Let be $\langle T, A, I\rangle$ be an abductive framework and $\langle T \cup \Gamma(A), I\rangle$ be its translation and Δ be a subset of A. $M(\Delta)$ is a generalized stable model of $\langle T, A, I\rangle$ if and only if there exists a stable model M' for*

$$\langle T \cup \Gamma(A), I\rangle$$

such that

$$M' = M(\Delta) \cup (A' - \Delta')$$

where Δ' is a set of corresponding newly introduced symbols in A' for all symbols in Δ.

Proof: See Appendix.

Example 1 *Generalized stable models [Kakas90b, p. 387]*

Consider the following logic programs T:

$$p \leftarrow b \tag{1}$$
$$q \leftarrow a \tag{2}$$

with abducibles $A = \{a, b\}$,

and the following integrity constraints I:

$$\leftarrow q \wedge b \tag{3}$$
$$\leftarrow \neg q \wedge \neg b \tag{4}$$

From the above logic programs, we can get $M_1 = \{b, p\}$ and $M_2 = \{a, q\}$ as generalized stable models.

Translation from this abductive framework is as follows. We will add the following rules, $\Gamma(A)$ to logic programs.

$$a \leftarrow \neg\tilde{a} \quad (5)$$
$$\tilde{a} \leftarrow \neg a \quad (6)$$
$$b \leftarrow \neg\tilde{b} \quad (7)$$
$$\tilde{b} \leftarrow \neg b \quad (8)$$

Then, consider the following two sets of propositions $M_1' = \{a, q, \tilde{b}\}$ and $M_2' = \{b, p, \tilde{a}\}$ for (1)~(8). We show that these sets are actually stable models.

Since the above sets satisfies integrity constraint (3) and (4), it suffices to show that those sets are stable models of $T \cup \Gamma(A)$. We denote $T \cup \Gamma(A)$ as T'.
$T'^{M_1'}$ becomes the following:

$$p \leftarrow b \quad (1)$$
$$q \leftarrow a \quad (2)$$
$$a \leftarrow \quad (5)'$$
$$\tilde{b} \leftarrow \quad (8)'$$

whose minimal model is equal to M_1'.
$T'^{M_2'}$ becomes the following:

$$p \leftarrow b \quad (1)$$
$$q \leftarrow a \quad (2)$$
$$\tilde{a} \leftarrow \quad (6)'$$
$$b \leftarrow \quad (7)'$$

whose minimal model is equal to M_2'.
Therefore, M_1' and M_2' are stable models of T'. Moreover,

$$M_1 \cup (A' \quad \Delta') = \{a, q, \tilde{b}\} = M_1'$$

and

$$M_2 \cup (A' - \Delta') = \{b, p, \tilde{a}\} = M_2'.$$

We have another correspondence with respect to explanation.

Theorem 2 *Let $\langle T, A, I\rangle$ be an abductive framework and $\langle T \cup \Gamma(A), I\rangle$ be its translation and Q be an observation and I' be $I \cup \{\leftarrow \neg Q\}$. Q has an*

explanation with a set of hypothesis Δ *if and only if there is a stable model* M' *for*

$$(T \cup \Gamma(A), I')$$

such that

$$\Delta = M' \cap A.$$

Proof: See Appendix.

Example 2 *Explanation*

Consider the abductive framework in Example 1. Suppose an observation q is given. Only explanation for this observation is $\{a\}$. We put the following integrity constraint into I to obtain I'

$$\leftarrow \neg q$$

and consider $\langle T', I' \rangle$. Then, among models of $\langle T', I \rangle$, only $M_1' = \{a, q, \tilde{b}\}$ is a stable model which satisfies $\leftarrow \neg q$ and $M_1' \cap A = \{a\}$ which is equal to explanation.

3 Computing Stable Models for Logic Program with Integrity Constraint

In this section, we give a bottom-up procedure to compute stable models for logic program with integrity constraint. To combine the previous translation and the following procedure, we can calculate abduction.

From the definition of stable models, one might think that it is sufficient to use the procedure of [Sacca90, Fages90] and remove every stable model which does not satisfy some integrity constraint in order to obtain all stable models. However, we can save search space if we can check integrity constraint during the process of constructing stable models. The following procedure performs not only such dynamic checking of integrity constraint but active use of integrity constraint to derive some facts.

Let $\langle T, I \rangle$ be a general logic program with integrity constraint.
A Procedure to Compute a Stable Model
$i := 0$,
$M_0, \widetilde{M}_0 := propagate(\emptyset, \emptyset)$.
If $M_0 \cap \widetilde{M}_0 \neq \emptyset$ then **fail**.

Step 1:

Select a rule $R = A \leftarrow L_1, L_2, ..., L_m$ in T satisfying the following conditions and go to **Step 2.**

1. $A \notin M_i$,
2. For every $P \in pos(R), P \in M_i$,
3. For every $N \in neg(R), N \notin M_i$.

If such rule is not found and there is an integrity constraint C in I s.t.
$M_i \not\models C$
then **fail**
else **return** M_i.

Step 2:

$i := i + 1$,
$M_i, \widetilde{M}_i := propagate(M_{i-1} \cup \{A\}, \widetilde{M}_{i-1} \cup neg(R))$
If $M_i \cap \widetilde{M}_i \neq \emptyset$ then **fail** else go to **Step 1**.

propagate$(M, \widetilde{M})$
begin
$k := 0$, $M_i^0 := M$, $\widetilde{M}_i^0 := \widetilde{M}$.
do
$k := k + 1$, $M_i^k := M_i^{k-1}$, $\widetilde{M}_i^k := \widetilde{M}_i^{k-1}$.
For every rule $R = A \leftarrow L_1, L_2, ..., L_m$ in T

1. If $A \notin M_i^{k-1}$ and for every $P \in pos(R), P \in M_i^{k-1}$ and for every $N \in neg(R), N \in \widetilde{M}_i^{k-1}$, then add A to M_i^k.
2. If $A \in \widetilde{M}_i^{k-1}$ and there exists $P \in pos(R)$ s.t. for every $P' \in pos(R)$ except P, $P' \in M_i^{k-1}$, for every $N \in neg(R), N \in \widetilde{M}_i^{k-1}$, then add P to $\widetilde{M}_i^k$.
3. If $A \in \widetilde{M}_i^{k-1}$ and for every $P \in pos(R), P \in M_i^{k-1}$ and for every $N \in neg(R), N \in \widetilde{M}_i^{k-1}$, then **fail.**

For every integrity constraint $C = \leftarrow L_1, L_2, ..., L_m$ in I,

4. If there exists $P \in pos(C)$ s.t. for every $P' \in pos(C)$ except P, $P' \in M_i^{k-1}$, and for every $N \in neg(C), N \in \widetilde{M}_i^{k-1}$, then add P to $\widetilde{M}_i^k$.

5. If for every $P \in pos(C), P \in M_i^{k-1}$ and for every $N \in neg(C), N \in \widetilde{M}_i^{k-1}$, then **fail**.

until $M_i^k = M_i^{k-1}$ **and** $\widetilde{M}_i^k = \widetilde{M}_i^{k-1}$.
return $M_i^k, \widetilde{M}_i^k$
end

In the procedure, **select** in Step 1 expresses nondeterminism and **fail** expresses going back to the recent choice point. M_i expresses a set of propositions which is decided to be in the belief set after selecting i rules and $\widetilde{M}_i$ expresses a set of propositions which is decided to be out of the belief set. And if there is a conflict between M_i and $\widetilde{M}_i$ then M_i is not possible candidate for a stable model. These set, M_i and $\widetilde{M}_i$ are equivalent to M_i and $\widetilde{M}_i$ in the procedure of [Sacca90, p.215] except that in our procedure, we check integrity constraint dynamically (the conflict checking in Step 2 and cases of 3 and 5 in the subprocedure of *propagate*) and $\widetilde{M}_i$ might increase by cases of 2 and 4 in the subprocedure of *propagate*.

We compare our procedure with the procedure of [Sacca90] with integrity constraint check afterwards. The following simple example expresses the difference.

Example 3 *Difference of Two Procedures*

Consider the following program:

$$p \leftarrow \neg q \quad (1)$$
$$q \leftarrow \neg p \quad (2)$$

and the integrity constraint:

$$\leftarrow p \quad (3)$$

The procedure of [Sacca90] produces stable models $\{p\}$ and $\{q\}$ for a logic program of (1) and (2) and then we select $\{q\}$ by integrity constraint (3). So, this execution has nondeterminism.

On the other hand, the execution of our procedure is as follows.

0. $M_0 = \{q\}, \widetilde{M}_0 = \{p\}$,
because from (3), p must be in $\widetilde{M}_0$ by case 4 in *propagate*,
and from (2), q must be in M_0 by case 1 in *propagate*.

1. Since there is no selected rule, M_0 is returned.

Therefore, in this example, we can calculate a stable model deterministically in our procedure. Note that in this execution, integrity constraint is used not only to derive that p is out of belief but also to derive a fact q. So, this example shows active usage of integrity constraint in our procedure.

In more complex example, we can propagate the information that a proposition is out of belief to other rules in a top-down manner thanks to case 2 in *propagate*.

Example 4 *Propagation of Information of Disbelieved Proposition*

Consider the following program:

$$p \leftarrow q \quad (1)$$
$$r \leftarrow \neg q \quad (2)$$
$$q \leftarrow \neg r \quad (3)$$

and the integrity constraint:

$$\leftarrow p \quad (4)$$

The execution of our procedure is as follows.

0. $M_0 = \{r\}, \widetilde{M}_0 = \{p, q\}$,
because from (4), p must be in $\widetilde{M}_0$ by case 4 in *propagate*,
and from (1), q must be in $\widetilde{M}_0$ by case 2 in *propagate*,
and from (2), r must be in M_0 by case 1 in *propagate*.

1. Since there is no selected rule, M_0 is returned.

We can show that the above procedure returns every stable model by appropriate selection of rules.

Theorem 3 *Let $\langle T, I\rangle$ be a logic program with integrity constraint.*

1. *If the procedure outputs M, then M is a stable models for $\langle T, I\rangle$.*

2. *If the procedure terminates without any outputs, then there is no stable models for $\langle T, I\rangle$.*

3. *If T and I are finite, then the procedure outputs all stable models by an exhaustive search.*

Proof: See Appendix.

Therefore, we can calculate abduction by combining the translation from abductive framework into logic program with integrity constraint and the above procedure to compute stable models for the logic program with integrity constraint.

Example 5 *Combination of Translation and Bottom-up Procedure*

Consider Example 1. We calculate the stable model for (1)$\sim$(8) in Example 1. We show a process of execution. We show how M_i and $\widetilde{M_i}$ are constructed for all combinations for selection of the rule.

Selection 1.

0. $M_0 = \emptyset, \widetilde{M_0} = \emptyset$.
1. Select rule (5). Then, $M_1 = \{a, q, \tilde{b}\}, \widetilde{M_1} = \{\tilde{a}, b\}$.
2. Since there is no selected rule, M_1 is returned.

Selection 2.

0. $M_0 = \emptyset, \widetilde{M_0} = \emptyset$.
1. Select rule (6). Then, $M_1 = \{\tilde{a}\}, \widetilde{M_1} = \{a\}$.
2. Select rule (7). Then, $M_2 = \{\tilde{a}, b, p\}, \widetilde{M_2} = \{a, \tilde{b}, q\}$.
3. Since there is no selected rule, M_2 is returned.

Selection 3.

0. $M_0 = \emptyset, \widetilde{M_0} = \emptyset$.
1. Select rule (6). Then, $M_1 = \{\tilde{a}\}, \widetilde{M_1} = \{a\}$.
2. Select rule (8). Then, $M_2 = \{\tilde{a}, \tilde{b}\}, \widetilde{M_2} = \{a, b\}$.

3. Although there is no selected rule, M_2 does not satisfy integrity constraint (4). So, this process fails.

Selection 4.

0. $M_0 = \emptyset, \widetilde{M}_0 = \emptyset$.
1. Select rule (7). Then, $M_1 = \{b, p, \tilde{a}\}, \widetilde{M}_1 = \{\tilde{b}, q, a\}$.
2. Since there is no selected rule, M_1 is returned.

Selection 5.

0. $M_0 = \emptyset, \widetilde{M}_0 = \emptyset$.
1. Select rule (8). Then, $M_1 = \{\tilde{b}\}, \widetilde{M}_1 = \{b\}$.
2. Select rule (5). Then, $M_2 = \{\tilde{b}, a, q\}, \widetilde{M}_2 = \{b, \tilde{a}\}$.
3. Since there is no selected rule, M_2 is returned.

Selection 6.

0. $M_0 = \emptyset, \widetilde{M}_0 = \emptyset$.
1. Select rule (8). Then, $M_1 = \{\tilde{b}\}, \widetilde{M}_1 = \{b\}$.
2. Select rule (6). Then, $M_2 = \{\tilde{b}, \tilde{a}\}, \widetilde{M}_2 = \{b, a\}$.
3. Although there is no selected rule, M_2 does not satisfy integrity constraint (4). So, this process fails.

So by exhaustive search, we find all stable models for (1)~(8), that is, $\{a, q, \tilde{b}\}$ and $\{b, p, \tilde{a}\}$.

4 Related Work

4.1 Eshghi's Topdown Procedure

In [Eshghi89], Eshghi and Kowalski give a restricted top-down procedure for abduction to handle negation by failure. Kakas and Mancarella extend this procedure so that arbitrary abducible can be used [Kakas90a]. However, these procedure is not correct for the following program [Eshghi89, p. 251].

$r \leftarrow \neg r$
$r \leftarrow q$
$p \leftarrow \neg q$
$q \leftarrow \neg p$

This program is translated into the following abductive framework of [Kakas90a].

$r \leftarrow r^*$
$r \leftarrow q$
$p \leftarrow q^*$
$q \leftarrow p^*$

with abducible $\{p^*, q^*, r^*\}$ and integrity constraint:

$\leftarrow p, p^*$
$p \vee p^*$
$\leftarrow q, q^*$
$q \vee q^*$
$\leftarrow r, r^*$
$r \vee r^*$

If an observation p is given then their procedure return the explanation $\{q^*\}$. However, this explanation is not correct because there is only one generalized stable model $M(\Delta)$ whose Δ is $\{p^*\}$ which does not include $\{q^*\}$ and this means that there is no explanation for the observation p.

On the other hand, thanks to our bottom-up nature of our procedure, we can not produce any explanation for the observation p by exhaustive search and this is a correct answer. More generally, our procedure is guaranteed to be correct in producing explanation by Theorem 2 and Theorem 3.

4.2 Truth Maintenance System

There are a lot of researches on semantics of Doyle's TMS [Elkan90, Fujiwara89, Junker90, Pimentel89, Reinfrank89]. However, none of works except [Fujiwara89] considers integrity constraint (nogoods in TMS terminology) explicitly in the definition of TMS. We have given a procedure which computes a grounded extension of TMS including nogoods [Satoh90]. From the relationship between TMS and logic programming which Elkan gives in [Elkan90], stable model for logic program without integrity constraint is equivalent to the grounded extension of TMS. In this paper, by generalizing this relationship, we translate our procedure of TMS into logic program with integrity constraint.

In this connection, an algorithm in [Junker90] is also related, but it does not consider nogoods explicitly.

Eshghi [Eshghi90] gives an algorithm using ATMS and a filtering mechanism to compute labels of propositions in stable models of propositional logic programs with negation as failure. However, like [Junker90], he only considers logic programs without integrity constraint.

Giordano and Martelli [Giordano90] gives translation of TMS program with nogoods to another TMS without nogoods to produce every grounded extension including extension obtained by dependency-directed backtracking(DDB). Although it is important in its own right to give a semantics for DDB of Doyle's TMS, it seems to be unsuitable for checking update in integrity constraint. Even if update is violated by the current integrity constraint, we might get other consistent states by performing DDB. Consider the following program

$$p \leftarrow q, \neg r$$

with integrity constraint

$$\leftarrow p.$$

If we add q to the program, we will get inconsistency and therefore q should be prohibited. However, Giordano and Martelli translate this program into the following program

$$p \leftarrow q, \neg r$$
$$r \leftarrow p^*, q$$
$$q^* \leftarrow p^*, \neg r$$

p^*

If we add q to this program, we will no longer get inconsistency and therefore q can be added. This change corresponds with performing DDB.

5 Conclusion

In this paper, we give a method of calculating abduction by translating abductive framework into logic program with integrity constraint and computing a stable model for the program.

Since our bottom-up procedure calculates all states of propositions, some of them is not relevant to explanation. In this case, we should pursue some top-down expectation to control bottom-up construction. This should be done as a future research.

Acknowledgments

We thank Katsumi Inoue from ICOT and Vladimir Lifschitz from Stanford University and The University of Texas at Austin for helpful discussion.

References

[Doyle79] Doyle, J., A Truth Maintenance System, *Artificial Intelligence*, **12**, pp. 231 - 272 (1979).

[Elkan90] Elkan, C., A Rational Reconstruction of Nonmonotonic Truth Maintenance Systems, *Artificial Intelligence*, **43**, pp. 219 - 234 (1990).

[Eshghi89] Eshghi, K., Kowalski, R. A., Abduction Compared with Negation by Failure, *Proc. of ICLP'89*, pp. 234 - 254 (1989).

[Eshghi90] Eshghi, K., Computing Stable Models by Using the ATMS *Proc. of AAAI'90*, pp. 272 - 277 (1990).

[Fages90] Fages, F., A New Fixpoint Semantics for General Logic Programs Compared with the Well-Founded and the Stable Model Semantics, *Proc. of ICLP'90*, pp. 442 – 458 (1990).

[Fujiwara89] Fujiwara, Y., Honiden, S., Relating the TMS to Autoepistemic Logic, *Proc. IJCAI'89*, pp. 1199 – 1205 (1989).

[Gelfond88] Gelfond, M., Lifschitz, V., The Stable Model Semantics for Logic Programming, *Proc. of LP'88*, pp. 1070 – 1080 (1988).

[Giordano90] L, Giordano, Martelli, A., Generalized Stable Models, Truth Maintenance and Conflict Resolution *Proc. of ICLP'90*, pp. 427 – 441 (1990).

[Junker90] Junker, U., Konolige, K., Computing the Extensions of Autoepistemic and Default Logics with a Truth Maintenance System, *Proc. of AAAI'90*, pp. 278 – 223 (1990).

[Kakas90a] Kakas, A. C., Mancella, P., On the relation between Truth Maintenance and Abduction, *Proc. of 3rd Nonmonotonic Reasoning Workshop*, pp. 158 – 176 (1990).

[Kakas90b] Kakas, A. C., Mancella, P., Generalized Stable Models: A Semantics for Abduction, *Proc. of ECAI'90*, pp. 385 – 391 (1990).

[Pimentel89] Pimentel, S. G., Cuadrado, J. L., A Truth Maintenance System based on Stable Models, *Proc. of NACLP'89*, pp. 274 – 290 (1990).

[Reinfrank89] Reinfrank, M., Dressler, O., Brewka, G., On the Relation between Truth Maintenance and Autoepistemic Logic, *Proc. IJCAI'89*, pp. 1206 – 1212 (1989).

[Sacca90] Sacca, D., Zaniolo, C., Stable Models and Non-Determinism in Logic Programs with Negation, *Proc. of PODS'90*, pp. 205 – 217 (1990).

[Satoh90] Satoh, K., Iwayama, N., Sugino, E., Konolige, K., An Implementation of TMS in Concurrent Logic Programming Language: Preliminary Report, *ICOT-TR-568*, ICOT (1990).

Appendix

Proof of Theorem 1:
We first prove the following Lemma.

Lemma 1 *Let* $M' = M(\Delta) \cup (A' - \Delta')$ *and* $T' = T \cup \Gamma(A)$. *Then,*

$$min(T'^{M'}) = min((T \cup \Delta)^{M(\Delta)}) \cup (A' - \Delta')$$

where $min(\Upsilon)$ *means a minimal model of a negated-atom-free logic program* Υ.

Proof:

$min(T'^{M'})$
$= min((T \cup \Gamma(A))^{M'})$
$= min(T^{M'} \cup \Gamma(A)^{M'})$
$= min(T^{M(\Delta) \cup (A' - \Delta')} \cup \Gamma(A)^{(M(\Delta) - \Delta) \cup \Delta \cup (A' - \Delta')})$

Since T does not contain any symbols in $A' - \Delta'$,
$T^{M(\Delta) \cup (A' - \Delta')} = T^{M(\Delta)}$.

And since $\Gamma(A)$ does not contain any symbols in $M(\Delta) - \Delta$,
$\Gamma(A)^{(M(\Delta) - \Delta) \cup \Delta \cup (A' - \Delta')}) = \Gamma(A)^{\Delta \cup (A' - \Delta')})$.

For every rule in $\Gamma(A)$, if $P \in \Delta$ then

$$\{P \leftarrow \neg \tilde{P}, \tilde{P} \leftarrow \neg P\}^{\Delta \cup (A' - \Delta')} = \{P\}$$

and if $P \notin \Delta$, that is, $\tilde{P} \in (A' - \Delta')$ then

$$\{P \leftarrow \neg \tilde{P}, \tilde{P} \leftarrow \neg P\}^{\Delta \cup (A' - \Delta')} = \{\tilde{P}\}$$

Therefore,

$$\Gamma(A)^{\Delta \cup (A' - \Delta')} = \Delta \cup (A' - \Delta').$$

Thus,

$min(T^{M(\Delta) \cup (A' - \Delta')} \cup \Gamma(A)^{(M(\Delta) - \Delta) \cup \Delta \cup (A' - \Delta')})$
$= min(T^{M(\Delta)} \cup \Delta \cup (A' - \Delta'))$
$= min((T \cup \Delta)^{M(\Delta)} \cup (A' - \Delta'))$ since $T^{M(\Delta)} \cup \Delta = (T \cup \Delta)^{M(\Delta)}$.
$= min((T \cup \Delta)^{M(\Delta)}) \cup (A' - \Delta')$ since no connection in $T \cup \Delta$ and A'. □

Now we prove Theorem 1.
(1) Assume $M(\Delta) = min((T \cup \Delta)^{M(\Delta)})$ and $M(\Delta)$ satisfies all of integrity constraints in I.

$min(T^{\prime M'})$
$= min((T \cup \Delta)^{M(\Delta)}) \cup (A' - \Delta')$ by Lemma 1,
$= M(\Delta) \cup (A' - \Delta')$ by the assumption,
$= M'$

This means that M' is a stable model of T'. Since $M(\Delta) \subseteq M'$, M' also satisfies all of integrity constraints in I.
(2) Assume $M' = min(T^{\prime M'})$ and M' satisfies all of integrity constraints in I.

$min(T^{\prime M'})$
$= min((T \cup \Delta)^{M(\Delta)}) \cup (A' - \Delta')$ by Lemma 1,
$= M(\Delta) \cup (A' - \Delta')$ by the assumption.

Since $min((T \cup \Delta)^{M(\Delta)})$ and $(A' - \Delta')$ are exclusive and $M(\Delta)$ and $(A' - \Delta')$ are exclusive,

$min((T \cup \Delta)^{M(\Delta)}) = M(\Delta)$.

This means that $M(\Delta)$ is a stable model of $T \cup \Delta$. Since every integrity constraint in I uses propositions only in T and Δ which receive same interpretation in M' and $M(\Delta)$, $M(\Delta)$ also satisfies all of integrity constraints in I. □

Proof of Theorem 2:
Suppose $M(\Delta)$ is a generalized stable model for $\langle T, A, I \rangle$ and $Q \in M(\Delta)$. This means $M(\Delta) \models (\leftarrow \neg Q)$. Therefore, $M(\Delta)$ is also a generalized stable model for $\langle T, A, I' \rangle$. From Theorem 1, there exists a stable model M' for $\langle T \cup \Gamma(A), I' \rangle$ such that $M' = M(\Delta) \cup (A' - \Delta')$. Thus, $M' \cap A = \Delta$.
Suppose M' is a stable model for $\langle T \cup \Gamma(A), I' \rangle$. From Theorem 1, there exists a generalized stable model $M(\Delta)$ for $\langle T, A, I' \rangle$ such that $M' = M(\Delta) \cup (A' - \Delta')$, that is, $M' \cap A = \Delta$. □

Proof of Theorem 3(Sketch):
Consider the following simple procedure to compute a stable model.

Let $\langle T, I \rangle$ be a general logic program with integrity constraint.
A Simple Procedure to Compute a Stable Model

$i := 0$

Step 1:

Select a rule $R_i = A_i \leftarrow L_1, L_2, ..., L_m$ in T satisfying the following conditions and go to **Step 2**.

1. $A_i \notin M_i$,
2. For every $P \in pos(R_i), P \in M_i$,
3. For every $N \in neg(R_i), N \notin M_i$.

If such rule is not found and there is an integrity constraint C in I s.t.
$M_i \not\models C$
then **fail**
else **return** M_i.

Step 2:

$M_{i+1} = M_i \cup \{A_i\}$
If there exists $R_k (0 \leq k \leq i)$ such that
for some $N \in neg(R_k), N \in M_{i+1}$
then **fail**
else i:=i+1, go to **Step 1**.

We denote our procedure in Section 3 as $proc(O)$ and denote the above simple procedure as $proc(S)$. We can show the following.

Lemma 2

1. *If there is a selection of rules such that $proc(O)$ outputs M, then there is a selection of rules such that $proc(S)$ also outputs M.*
2. *If $proc(O)$ terminates without any outputs, then $proc(S)$ also terminates without any outputs.*
3. *If T and I are finite and there is a selection of rules such that $proc(S)$ outputs M, then there is a selection of rules such that $proc(O)$ also outputs M.*

Then, we show the following.

Lemma 3

1. *If $proc(S)$ outputs M, then M is a stable models for $\langle T, I \rangle$.*

2. *If $proc(S)$ terminates without any outputs, then there is no stable models for $\langle T, I\rangle$.*

3. *If T and I are finite, then proc(S) outputs all stable models by an exhaustive search.*

To show Lemma 3, we need the following definition of *grounded model*.

Definition 7 *Let be $\langle T, I\rangle$ a logic program T with integrity constraint I. A set of propositions M is a* grounded model *for $\langle T, I\rangle$ if the following are satisfied.*

1. *M is a model of T.*

2. *M satisfies every $C \in I$.*

3. *M can be written as a sequence of propositions $\langle P_1, P_2, ...\rangle$ such that each P_j has at least one rule R_j such that $pos(R) \subseteq \{P_1, ..., P_{j-1}\}$ where $P_1, ..., P_{j-1}$ are the element of the sequence and for every $N \in neg(R)$, $N \notin M$. We say a sequence of such rules for every propositions in M, $\langle R_1, R_2, ...\rangle$, is a* a sequence of supporting rules *for M.*

And we need the following lemma.

Lemma 4 *Let be $\langle T, I\rangle$ a logic program T with integrity constraint I. A set of propositions M is a* grounded model *for $\langle T, I\rangle$ if and only if M is a stable model for $\langle T, I\rangle$.*

Proof: From [Elkan90, Theorem 3.8], M is a *grounded model* for $\langle T, \emptyset\rangle$ if and only if M is a stable model for $\langle T, \emptyset\rangle$. (Note that his result is applicable even if T is countably infinite.) If M is a grounded model for $\langle T, I\rangle$, then M satisfies every $C \in I$, so M is a stable model for $\langle T, I\rangle$. The converse is similarly true. □

Proof of Lemma 3:

1. If $proc(S)$ outputs M with a finite sequence of selected rules $R_0, ..., R_n$, then this sequence actually gives a sequence of supporting rules. And it is clear that M satisfies every $C \in I$. Therefore, M is a grounded model and, so, a stable model for $\langle T, I\rangle$ by Lemma 4.

2. Suppose $proc(S)$ terminates without any output and there is a stable model M. If M is infinite there is a infinite sequence of supporting rules $R_0, ...$ and $proc(S)$ does not terminate by applying this sequence as a selection of rules, so, M must be finite. Then, there is a finite sequence of supporting rules $R_0, ..., R_n$. We can select rules along this sequence in $proc(S)$ and the above procedure outputs M. Contradiction.

3. Let $\langle T, I \rangle$ be a finite logic program T with integrity constraint I. Suppose M is a stable model for $\langle T, I \rangle$. Then, there is a finite sequence of supporting rules $R_0, ..., R_n$. We can select rules along this sequence in $proc(S)$ and the above procedure outputs M.

□

Theorem 3 is proved by Lemma 2 and Lemma 3. □