Translating Multi-Agent Autoepistemic Logic into Logic Program

TOYAMA, Katsuhiko\textsuperscript{a,b,1}, KOJIMA, Takahiro\textsuperscript{a},
and INAGAKI, Yasuyoshi\textsuperscript{a,2}

\textsuperscript{a} Graduate School of Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, JAPAN
\textsuperscript{b} Center for Integrated Acoustic Information Research, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, JAPAN

Abstract

In order to develop a proof procedure of multi-agent autoepistemic Logic (MAEL),
a natural framework to formalize belief and reasoning including inheritance, persist-
tence, and causality, we introduce a method that translates a MAEL theory into a
logic program with integrity constraints. It is proved that there exists one-to-one
correspondence between extensions of a MAEL theory and stable models of a logic
program translated from it. Our approach has the following advantages: (1) We
can obtain all extensions of a MAEL theory if we compute all stable models of the
translated logic program. (2) We can fully use efficient techniques or systems for
computing stable models of a logic program. We also investigate the properties of
reasoning in MAEL through this translation. The fact that the extension computing
problem can be reduced to the stable model computing problem implies that there
are close relationships between MAEL and other formalizations of nonmonotonic
reasoning.

1 Introduction

It is well known that there are difficult problems such as multi-extension
problem\cite{3,17} and temporal projection problem\cite{5} which state that undesir-
able reasoning results could arise simultaneously when we formalize belief by
using nonmonotonic logics.

As a method to handle such problems in the framework of formal logic,
multi-agent autoepistemic logic (MAEL)\cite{19} has been proposed. MAEL is

\textsuperscript{1} Email: toyama@nuie.nagoya-u.ac.jp
\textsuperscript{2} Email: inagaki@nuie.nagoya-u.ac.jp

©2002 Published by Elsevier Science B. V. Open access under CC BY-NC-ND license.
Toyma, Kojima and Inagaki formalized by extending autoepistemic logic (AEL)[13] to the one with multimodality and multi-theory, where belief acquisition relations between agents and their priorities can be specified by formulas. Thus, MAEL is a so flexible framework that we can naturally represent belief and reasoning including inheritance, persistence, and causality[19].

Since it is required to realize a belief processing system based on MAEL, an efficient theorem proof procedure for it (computation method of extensions of a theory) should be developed. The procedures proposed in [6,7] were based on the tableau method combined with resolution principle. However, it only determines whether a given formula may be contained in the reasoning results derived from the initial belief set and it is not easy to obtain whole results of the reasoning (what and how many result sets are derived from the initial belief set). Therefore, it is a fundamental issue to develop a new proof procedure for MAEL in which all reasoning results can be obtained, and it is also important to clarify the properties of reasoning in MAEL.

In order to model the reasoning systems based on incomplete belief or common sense knowledge, the following formalizations of nonmonotonic reasoning have been proposed so far: default logic, NML, AEL, and circumscription, logic programming, truth maintenance system (TMS), and abduction, etc. Although these formalizations have been proposed from different backgrounds and motivations, it was shown in various researches that they are closely related to each other[2,4,8,9,11,12,14]. In particular, stable model semantics[4] which was proposed as declarative semantics of logic programs is constituted by applying the concept of stability of a theory in AEL. Gelfond[4] has shown that a logic program with stable model semantics can be regarded as a theory of AEL with restricted syntax. This means that a logic program can be interpreted in AEL through the translation to a theory of AEL.

In this paper, we show that the inverse translation to Gelfond’s result is also possible, i.e., the translation of a theory of AEL into a logic program with stable model semantics, and also that the translation can be extended to the multi-agent system, i.e., MAEL.

In the light of the goal to realize a reasoning system based on MAEL, this translation is important at the following two points:

• This translation can be used as a new theorem proof procedure for MAEL. A concrete method of translating a theory of MAEL into a logic program is given in this paper, where one-to-one correspondence between the reasoning results before and after the translation is guaranteed. Thus, if a theory of MAEL is once translated into a logic program and stable models of it are computed, we can obtain extensions of the theory. This suggests that it comes to be able to carry out direct use of the research results and the existing systems about the computation of stable models in the theorem proof of MAEL. Moreover, we can obtain all extensions of the theory if all the stable models of the translated logic program are computed.
• This translation shows clearly that two different formalizations of nonmonotonic reasoning, MAEL and logic programming, are closely related. The result of this paper shows that MAEL can be characterized in terms of logic programming which is a simpler formalization of nonmonotonic reasoning if the translation rule is described appropriately. The attempt to understand MAEL from a simpler formalization is useful when we investigate the properties of the reasoning formalized in MAEL. Also, this suggests the possibility of logic programming as a fundamental language which describes nonmonotonic reasoning systems for multi-agents.

In order to enable this translation, the following two problems are to be solved:

• How to translate a theory of MAEL which was extended to describe a multi-agent system into a single agent’s theory, and how to obtain expressions in a logic program.

• How to describe each agent’s reasoning capability (classically logical reasoning capability and autoepistemic reasoning capability) which are formalized in MAEL in terms of rules of a logic program.

In this paper, we show that these problems are solved and that the translation from a theory of MAEL into a logic program is possible.

The remaining of the paper is organized as follows: Section 2 surveys definitions and characterizations about MAEL. Also definitions about a logic program including integrity constrains and stable model semantics are described. In Section 3, a fixed point called an extension base is defined, which characterizes an extension of a MAEL theory. The extension base will be a finite fixed point set if a theory is a finite set. Section 4 shows how to translate a MAEL theory into a logic program including integrity constraints. The reasoning results before and after the translation (extensions and stable models) have one-to-one correspondence. In Section 5, we have discussions on the properties of the reasoning in MAEL and relations to the previous studies. Section 6 shows conclusions of this paper.

2 Preliminaries

2.1 Multi-Agent Autoepistemic Logic

MAEL[19] is a nonmonotonic logic for multi-agent systems and is formalized by extending AEL[13] to the one with multi-modality and multi-theory. It characterizes belief states which an agent can obtain by propositional reasoning and autoepistemic reasoning in multi-agent systems. A set of formulas $\mathcal{L}_{mael}$ of MAEL is constructed from the set of propositional formulas $\mathcal{L}_0$ and modal operators $L_i(i = 1, \ldots, n)$ in a usual way. A formula $L_ip$ denotes a meta belief that agent $i$ has a belief $p$ (or, agent $i$ believes $p$). A theory is a $n$-tuple $T = (T_1, \ldots, T_n)$ of formula sets, where $T_i$ denotes a set of beliefs of
An agent formalized in MAEL is assumed to be ideal on the capability of logical reasoning and autoepistemic reasoning in the following way:

- **Logical reasoning capability**: each agent is logically omniscient and his beliefs are closed under propositionally logical consequence.

- **Autoepistemic reasoning capability**: Belief acquisition from agents is formalized in terms of autoepistemic reasoning in multi-agent systems, i.e., each agent can obtain a meta belief $L_i p$ if agent $i$ has a belief $p$, and each agent can obtain a meta belief $\neg L_i p$ if agent $i$ does not have a belief $p$.

If initial belief states of each agent are given, MAEL characterizes final belief states (an *extension* of a theory) which are obtained by the propositionally logical reasoning and the autoepistemic reasoning.

**Definition 2.1** An ordinal atomic formula is a propositional variable in $L_0$. A modal atomic formula is a formula of the form $L_i p$ where $p$ is a formula. An atomic formula is either an ordinal atomic formula or a modal one. Atomic formulas and their negations are called *literals*.

**Definition 2.2** A propositional interpretation $I$ is a usual assignment of truth values to formulas. If $I$ assigns true to a formula $p$, we denote $I |\!|= p$. A propositional interpretation which assigns true to each element of a set of formulas $P_i$ is a propositional model of $P_i$.

Since a modal atomic formula $L_i p$ is treated as like a new propositional variable in a propositional interpretation, its truth value does not depend on the value of $p$. Thus, we define an interpretation which reflects intended meaning of $L_i p$.

**Definition 2.3** An autoepistemic interpretation with respect to a theory $T = (T_1, \ldots, T_n)$ is a propositional interpretation $I$ which satisfies

$$I |\!|= L_i p \text{ iff } p \in T_i \quad (i = 1, \ldots, n)$$

for any formula $p$. An autoepistemic interpretation with respect to $T$ which assigns true to each element of a set of formulas $P_i$ is an autoepistemic model of $P_i$ with respect to $T$.

Next, we define a logical consequence relation $|=T$ in terms of the autoepistemic model.

**Definition 2.4** Let $P_i$ be a set of formulas and $T = (T_1, \ldots, T_n)$ be a theory. Then, for a formula $p$, $P_i |=_T p$ if $I |\!|= p$ for any autoepistemic model $I$ of $P_i$ with respect to $T$.

We define an extension of a theory which is considered to be a reasoning result (a set of theorem) in MAEL.
Definition 2.5 Let $P = (P_1, \ldots, P_n)$ be a theory. A theory $T = (T_1, \ldots, T_n)$ which satisfies

$$T_i = \{q \in L_{mael} \mid P_i \models_T q\} \quad (i = 1, \ldots, n)$$

is an extension of $P$.

From this definition, 0 or more extensions may exist for a theory in general.

Definition 2.6 A theory $T = (T_1, \ldots, T_n)$ is stable if it satisfies the following conditions (1)–(3) for any $i$ ($i = 1, \ldots, n$):

1. $T_i$ is closed under propositional consequence.
2. If $p \in T_i$, then $L_i p \in T_j$ for any $j$ ($j = 1, \ldots, n$).
3. If $p \notin T_i$, then $\lnot L_i p \in T_j$ for any $j$ ($j = 1, \ldots, n$).

It is known that an extension of a theory is stable[19].

2.2 Knowledge Representation based on MAEL

As an application of MAEL, a method to represent inheritance of knowledge is proposed in [19] where a class of knowledge in the taxonomic hierarchy is regarded as an agent. When beliefs of agent $i$ are inherited from beliefs of agent $j$, a formula of the form

$$L_j p \land \lnot L_i \lnot p \supset p$$

is given to a set of initial beliefs $P_i$ of agent $i$, where $p$ is a belief to be inherited. We can read this formula as follows: if agent $i$ obtain both a meta belief that agent $j$ has a belief $p$ and a meta belief that agent $i$ does not have any belief that is inconsistent to $p$, then agent $i$ can obtain a belief $p$.

Example 2.7 Consider the following beliefs (“Nixon Diamond”) which arise multiple inheritance of attributes: “A quaker is a pacifist.”, “A republican is not a pacifist.”, “Nixon is a quaker.”, “Nixon is a republican.” These beliefs can be represented by MAEL. Let agent 1, 2, 3 be attribute holders of a quaker, a republican, and Nixon, respectively. Then, as initial belief states of the agents, we describe a theory $P = (P_1, P_2, P_3)$ as follows:

$$P_1 = \{\text{Pacifist}\}$$

$$P_2 = \{\neg\text{Pacifist}\}$$

$$P_3 = \{L_1 p \land \lnot L_3 \lnot p \supset p, \ L_2 p \land \lnot L_3 \lnot p \supset p\}$$

where $p$ is an arbitrary formula. Formulas in $P_3$ denote that Nixon inherits attributes of both quakers and republicans.

The theory $P$ has two extensions, $T = (T_1, T_2, T_3)$ and $T' = (T'_1, T'_2, T'_3)$, where Pacifist $\in T_3$, while $\neg\text{Pacifist} \in T'_3$. That is, we can obtain two reasoning results “Nixon is a pacifist” and “Nixon is not a pacifist” which contradict each other. Thus, by using MAEL, we can represent beliefs so that two or more
reasoning results may be obtained without making a theory contradictory when competition about reasoning between agents arises.

2.3 Normal Forms of MAEL

We show that any formula of MAEL can be translated into clausal form and normal form.

Definition 2.8 A formula of the following form is in clausal form.

\begin{align*}
L_1a_{1,1} & \land \ldots \land L_1a_{1,l(1)} \land \neg L_1b_{1,1} & \land \ldots & \land \neg L_1b_{1,m(1)} \land \ldots \land \\
L_na_{n,1} & \land \ldots \land L_na_{1,l(n)} \land \neg L_nb_{n,1} & \land \ldots & \land \neg L_nb_{n,m(n)} \supset c,
\end{align*}

where \( a_{i,j}, b_{i,j}, c \in L_0 \), and each element of the conjunction in the antecedent may not exist. A theory which is a \( n \)-tuple of a set of clausal form formulas is also called to be in clausal form.

The next theorem[16] gives an algorithm which translates any formula into a clausal form, that is, by applying (1)–(7) to each formula in the left to right direction repeatedly, we can obtain its clausal form.

Theorem 2.9 For an interpretation \( I \) with respect to any stable theory \( T = (T_1, \ldots, T_n) \), the following equivalence (1)–(7) hold for any formulas \( p, q \):

\begin{align*}
(1) \quad I \models L_i(p \land q) & \iff I \models L_i p \land L_i q, \\
(2) \quad I \models L_i L_i p & \iff I \models L_i p, \\
(3) \quad I \models L_j L_i p & \iff I \models L_i p \lor L_j \bot \quad (i \neq j), \\
(4) \quad I \models L_j \neg L_i p & \iff I \models \neg L_i p \lor L_j \bot, \\
(5) \quad I \models L_i (L_i p \lor q) & \iff I \models L_i p \lor L_i q, \\
(6) \quad I \models L_j (L_i p \lor q) & \iff I \models L_i p \lor L_j q \lor L_j \bot \quad (i \neq j), \\
(7) \quad I \models L_j (\neg L_i p \lor q) & \iff I \models \neg L_i p \lor L_j q \lor L_j \bot,
\end{align*}

where \( \bot \) is a special variable which denotes inconsistency.

Note that (4) and (7) also hold even if \( i = j \).

This translation is illustrated in Fig. 1 where \( P, Q, R \) are propositional variables. A theory whose elements are obtained by this translation is in clausal form. From the fact that extensions of a theory are stable and the above theorem that shows formulas are equivalent through the translation, Ogawa[16] has shown that we can translate any theory into clausal form without changing its extensions.

Definition 2.10 A formula of the following form is in normal form:

\begin{align*}
L_1a_1 & \land \neg L_1b_{1,1} & \land \ldots & \land \neg L_1b_{1,m(1)} \land \ldots \land \\
L_na_n & \land \neg L_nb_{n,1} & \land \ldots & \land \neg L_nb_{n,m(n)} \supset c,
\end{align*}
where \( a_i, b_{i,j}, c \in L_0 \), and each element of the conjunction in the antecedent may not exist. A theory which is a \( n \)-tuple of a set of normal form formulas is also called to be in normal form.

By Theorem 2.9 (1) again, we can replace the subformula \( L_i a_{i,1} \land \ldots \land L_i a_{i,l(i)} \) in clausal form with \( L_i(a_{i,1} \land \ldots \land a_{i,l(i)}) \), so that it is obvious that there exists an equivalent normal form for any clausal form formula. Thus, we can immediately obtain the following theorem, which is a natural extension of the result in [11] to the case of multi-agents.

**Theorem 2.11** Any theory can be translated into a normal form theory without changing its extensions.

Note that any nest of modal operators does not occur in normal form formula. Thus, we can remove them without changing its meaning.

In the following, we assume that a theory is in normal form.

### 2.4 Stable Model Semantics of Logic Program

In this section, we define a stable model of a logic program with integrity constrains as the same way as [18]. We assume that all the variables occurred in program rules are grounded.

**Definition 2.12** Let \( A_i(i = 0, \ldots, n) \) be an atom. A *logic program* is a set of

\[
L_1((L_2 P \land Q) \lor \neg L_1 P) \lor R \\
\downarrow \\
L_1((L_2 P \lor \neg L_1 P) \land (Q \lor \neg L_1 P)) \lor R \\
\downarrow \\
(L_1(L_2 P \lor \neg L_1 P) \land L_1(\neg L_1 P \lor Q)) \lor R \\
\downarrow \\
((L_2 P \lor L_1 \neg L_1 P \lor L_1 \bot) \land (\neg L_1 P \lor L_1 Q \lor L_1 \bot)) \lor R \\
\downarrow \\
((L_2 P \lor \neg L_1 P \lor L_1 \bot \lor L_1 \bot) \land (\neg L_1 P \lor L_1 Q \lor L_1 \bot)) \lor R \\
\downarrow \\
(L_2 P \lor \neg L_1 P \lor L_1 \bot \lor R) \land (\neg L_1 P \lor L_1 Q \lor L_1 \bot \lor R) \\
\downarrow \\
(L_1 P \lor \neg L_1 \bot \lor \neg L_2 P \lor R) \land (L_1 P \lor \neg L_1 Q \lor L_1 \bot \lor R)
\]
program rules of the form
\[ A_0 \leftarrow A_1, \ldots, A_m, \text{not}A_{m+1}, \ldots, \text{not}A_n. \]
where \(0 \leq m \leq n\). \(A_0\) is a head of the rule and \(A_1, \ldots, A_m, \text{not}A_{m+1}, \ldots, \text{not}A_n\) are bodies of it.

**Definition 2.13** An integrity constraint is a program rule where the head is a special atom \(false\) which denotes incoherence.

**Definition 2.14** Let \(P\) be a logic program with integrity constraints. A stable model of \(P\) is a set \(S\) of atoms which satisfies the following conditions (1) and (2):

(1) \(S\) coincides with a minimal model of reduct\((P, S)\) which is a definite logic program obtained from \(P\) by the following procedures (a) and (b).
   (a) Remove all rules such that \(\{A_{m+1}, \ldots, A_n\} \cap S \neq \emptyset\) from \(P\).
   (b) Remove all negative literals \(\text{not}A_{m+1}, \ldots, \text{not}A_n\) from the bodies of the remaining rules.

(2) \(false \notin S\).  

3 Extension Bases

In this section, we define a fixed point which characterizes extensions of a theory, where the formulas to which the modal operators in the theory refer play important roles.

**Definition 3.1** Let \(P = (P_1, \ldots, P_n)\) be a normal form theory. A theory \(L_P = (L_{P_1}, \ldots, L_{P_n})\) which is defined as follows is a test domain of \(P\):

\[ L_{P_i} = \{q \mid L_i q \text{ occurs in } \bigcup_{j=1}^{n} P_j\} \quad (i = 1, \ldots, n). \]

Since formulas in \(P\) are assumed to be in normal forms, \(L_{P_i} \subseteq L_0\).

From this definition, if a theory \(P\) is consisted of finite sets of formulas, \(L_P\) is also finite.

Next, for a normal form theory, we define a fixed point which is a subset of the test domain of it.

**Definition 3.2** Let \(P = (P_1, \ldots, P_n)\) be a normal form theory and \(L_P = (L_{P_1}, \ldots, L_{P_n})\) be a test domain of \(P\). Then a theory \(B = (B_1, \ldots, B_n)\) such that

\[ B_i = \{q \in L_{P_i} \mid P_i \models_B q\} \quad (i = 1, \ldots, n) \]

is an extension base of \(P\).

\[1\] The condition (2) can be deleted if we add “odd loop”

\[ false' \leftarrow \text{not} false', false. \]

to the logic program, where \(false'\) is an atom which never occurs in other rules.
Theorem 3.3 Let \( P = (P_1, \ldots, P_n) \) be a normal form theory and \( \mathcal{L}_P = (\mathcal{L}_{P_1}, \ldots, \mathcal{L}_{P_n}) \) be a test domain of \( P \). Then there exists one-to-one correspondence between extensions and extension bases of \( P \). An extension \( T = (T_1, \ldots, T_n) \) and an extension base \( B = (B_1, \ldots, B_n) \) which correspond to each other satisfy \( T_i \cap \mathcal{L}_{P_i} = B_i \) \((i = 1, \ldots, n)\).

This theorem shows the fact that extensions of a theory can be characterized in terms of the condition that whether the formulas to which the modal operators in the theory refer belong to the extension of it or not.

Example 3.4 Consider Example 2.7 again. To be a theory \( P \) a triple of finite sets of formulas, we restrict the attributes to be inherited as follows:

\[
P_1 = \{\text{Pacifist}\}, \quad P_2 = \{\neg\text{Pacifist}\}, \quad P_3 = \{L_1\text{Pacifist} \land \neg L_3\neg\text{Pacifist} \supset \text{Pacifist}, \quad L_2\neg\text{Pacifist} \land \neg L_3\neg\neg\text{Pacifist} \supset \neg\text{Pacifist}\}.
\]

As well as Example 2.7, \( P = (P_1, P_2, P_3) \) has two extensions. The test domain of \( P \) is

\[
\mathcal{L}_P = (\mathcal{L}_{P_1}, \mathcal{L}_{P_2}, \mathcal{L}_{P_3}) = (\{\text{Pacifist}\}, \{\neg\text{Pacifist}\}, \{\neg\text{Pacifist}, \neg\neg\text{Pacifist}\}).
\]

In the case that \( B = (B_1, B_2, B_3) \)

\[
= (\{\text{Pacifist}\}, \{\neg\text{Pacifist}\}, \{\neg\neg\text{Pacifist}\}),
\]

\( B \) is an extension base of \( P \) since

\[
\{q \in \mathcal{L}_{P_1} \mid P_1 \models B q \} = \{\text{Pacifist}\} = B_1,
\]

\[
\{q \in \mathcal{L}_{P_2} \mid P_2 \models B q \} = \{\neg\text{Pacifist}\} = B_2,
\]

\[
\{q \in \mathcal{L}_{P_3} \mid P_3 \models B q \} = \{\neg\neg\text{Pacifist}\} = B_3.
\]

As the same way,

\[
B' = (B'_1, B'_2, B'_3)
\]

\[
= (\{\text{Pacifist}\}, \{\neg\text{Pacifist}\}, \{\neg\text{Pacifist}\})
\]

is also an extension base of \( P \).

4 Translation from MAEL into Logic Program

In this section, we show a method to translate a normal form theory into a logic program. We can compute extension bases from this translated logic program.
**Definition 4.1** For a normal form theory $P = (P_1, \ldots, P_n)$, $tr(P)$ is a logic program which consists of program rules generated by the following translation rules (1)–(3):

1. For any $i$ ($i = 1, \ldots, n$), if
   
   $$L_1a_1 \land \neg L_1b_{1,1} \land \ldots \land \neg L_1b_{1,m(1)} \land \ldots \land L_na_n \land \neg L_nb_{n,1} \land \ldots \land \neg L_nb_{n,m(n)} \supset c \in P_i,$$
   
   then
   
   $$fact(i,c) \leftarrow bel(1,a_1), not\_bel(1,b_{1,1}), \ldots, not\_bel(1,b_{1,m(1)}), \ldots, bel(n,a_n), not\_bel(n,b_{n,1}), \ldots, not\_bel(n,b_{n,m(n)}).$$

2. For any $i$ ($i = 1, \ldots, n$), if $p \in L_i$ and $Q = \{q_1, \ldots, q_k\}$ is a minimal subset of $C_{P_i}$ such that $(P_i \cap L_0) \cup Q \models p$,
   
   $$fact(i,p) \leftarrow fact(i,q_1), \ldots, fact(i,q_k). \in tr(P),$$
   
   where
   
   $$C_{P_i} = \{c \mid L_1a_1 \land \ldots \land \neg L_nb_{n,m(n)} \supset c \in (P_i - L_0)\}.$$

3. For any $i$ ($i = 1, \ldots, n$), if $p \in L_{P_i}$, then
   
   $$bel(i,p) \leftarrow not\_not\_bel(i,p).$$
   
   $$not\_bel(i,p) \leftarrow not\_bel(i,p).$$
   
   $$false \leftarrow bel(i,p), not\_fact(i,p).$$
   
   $$false \leftarrow not\_not\_bel(i,p), fact(i,p).$$

$$\in tr(P).$$

We explain the meaning of each type of the program rules.

(1) is a program rule which was translated from a belief or an inference rule which agent $i$ believes. At this time, new predicates $bel$, $not\_bel$, and $fact$ are introduced to a logic program, and literals of the form $L_ip$ and $\neg L_ip$ are replaced with $bel(i,p)$ and $not\_bel(i,p)$, respectively. An ordinal atomic formula $p$, which does not include modal operators, is replaced with an atom $fact(i,p)$ if $p$ occurs in $P_i$.

(2) is a program rule for giving the capability of logical implication to a logic program since agents are assumed to be logically omniscient. Programs rules in a logic program can be used only in the right to left direction, so that it is needed to describe the logical consequence relations as rules clearly. However, since it is difficult to describe all the relations, we describe only the parts required for the computation of extension bases. When it is verified whether the condition in Definition 3.2 is satisfied, the information about whether formulas in a test domain $L_P$ is derived from each agent’s initial
belief is needed. Thus, for any agent $i$ ($i = 1, \ldots, n$), only the rules about the propositional derivability of formulas in $L_{P_i}$ from initial beliefs $P_i \cap L_0$ and beliefs $C_{P_i}$ which may be newly obtained should be described.

(3) is a program rule to for giving the capability of autoepistemic reasoning to a logic program. First two rules of (3) describe the properties of predicates $\text{bel}$ and $\text{not\_bel}$, which divide stable models into the one where $\text{bel}(i, p)$ holds and the others where $\text{not\_bel}(i, p)$ holds. This corresponds to the fact that one of the literals of $L_i p$ and $\neg L_i p$ holds for any agent $i$ in MAEL. Since these rules are generated for any formula in a test domain $L_P$, we can take into consideration all the belief states about whether each agent has a formula in $L_P$ as his belief. This exhaustiveness guarantees that we can obtain all the extension bases from stable models (completeness).

The last two rules of (3) are integrity constraints which describe that $\text{fact}(i, p)$ have to hold in the models where $\text{bel}(i, p)$ holds and that $\text{fact}(i, p)$ does not have to hold in the models where $\text{not\_bel}(i, p)$ holds. Intended meanings of those constraints are the following ones about agents’ beliefs: agent $i$ can derive a belief $p$ from his initial beliefs by some reasoning ($P_i \models_B p$) if he has a belief $p$ ($p \in B_i$), and agent $i$ cannot derive a belief $p$ from his initial beliefs by some reasoning ($P_i \not\models_B p$) if has a belief $p$ ($p \notin B_i$). They become the constraints that the definition of an extension base requires if they are restricted to the formulas in a test domain $L_P$. Thus, the existence of these rules guarantees that a theory obtained from a stable model is surely an extension base (soundness).

**Example 4.2** Consider a translation of a theory $P = (P_1, P_2, P_3)$ in Example 3.4 into a logic program. A test domain of $P$ is

$$L_P = (\{\text{Pacifist}\}, \{\neg\text{Pacifist}\}, \{\neg\text{Pacifist}, \neg\neg\text{Pacifist}\})$$

and we obtain

$$C_P = (C_{P_1}, C_{P_2}, C_{P_3})$$

$$= (\{\}, \{\}, \{\text{Pacifist}, \neg\text{Pacifist}\}) .$$

The logic program $tr(P)$ translated from $P$ is shown in Fig. 2. The rule

$$\text{fact}(3, \neg\neg\text{Pacifist}) \leftarrow \text{fact}(3, \text{Pacifist}).$$

in (2) describes that agent 3 can also obtain a belief $\neg\neg\text{Pacifist}$ by propositionally logical reasoning when he obtain a belief $\text{Pacifist}$ by some reasoning.

Now we give a theorem which shows that reasoning results before and after the translation correspond to each other.

**Definition 4.3** For a set of formulas $P_i$ and a theory $T = (T_1, \ldots, T_n)$, we
define an operator $\text{reduct}_M$ as follows:

$$\text{reduct}_M(P_i, T) = \{ c \mid L_1 a_1 \land \neg L_1 b_{1,1} \land \ldots \land \neg L_1 b_{1,m(1)} \land \ldots \land L_n a_n \land \neg L_n b_{n,1} \land \ldots \land \neg L_n b_{n,m(n)} \supset c \in P_i, a_j \in T_j, b_{j,1}, \ldots, b_{j,m(j)} \notin T_j (j = 1, \ldots, n) \}.$$ 

**Lemma 4.4** Let $P = (P_1, \ldots, P_n)$ be a theory, and $S$ be a stable model of $\text{tr}(P)$. Then there uniquely exists an extension base $B = (B_1, \ldots, B_n)$ of $P$.
such that

\[
S = \bigcup_{i=1}^{n} \{ \text{bel}(i, p) \mid p \in B_i \}
\]

\[
\bigcup_{i=1}^{n} \{ \text{not}_\text{bel}(i, p) \mid p \in (\mathcal{L}_{P_i} - B_i) \}
\]

\[
\bigcup_{i=1}^{n} \{ \text{fact}(i, p) \mid p \in B_i \cup \text{reduct}_M(P_i, B) \}
\].

**Lemma 4.5** Let \( P = (P_1, \ldots, P_n) \) be a theory and \( B = (B_1, \ldots, B_n) \) be an extension base of \( P \). Then

\[
S = \bigcup_{i=1}^{n} \{ \text{bel}(i, p) \mid p \in B_i \}
\]

\[
\bigcup_{i=1}^{n} \{ \text{not}_\text{bel}(i, p) \mid p \in (\mathcal{L}_{P_i} - B_i) \}
\]

\[
\bigcup_{i=1}^{n} \{ \text{fact}(i, p) \mid p \in B_i \cup \text{reduct}_M(P_i, B) \}
\]

is a stable model of \( \text{tr}(P) \), i.e., \( S = T_{\text{reduct}(\text{tr}(P), S)} \uparrow \omega \).

From these lemmas, we can prove the next result.

**Theorem 4.6** Let \( P \) be a normal form theory and \( \text{tr}(P) \) be a translation from \( P \). Then there exists one-to-one correspondence between extension bases of \( P \) and stable models of \( \text{tr}(P) \). An extension base \( B = (B_1, \ldots, B_n) \) and a stable model \( S \) which correspond to each other satisfy \( B_i = \{ p \mid \text{bel}(i, p) \in S \} \) (\( i = 1, \ldots, n \)).

This theorem guarantees that there exists a stable model for any extension base. Thus, if all the stable models of the logic program translated from a theory of MAEL are computed, we can obtain all the extension bases of the theory. Moreover, we can obtain a corresponding extension base from the stable model by taking out atoms of the form \( \text{bel}(i, p) \) and taking out arguments from them.

**Example 4.7** There exist two stable models \( S \) and \( S' \) for \( \text{tr}(P) \) in Example 4.2:

\[
S = \{ \text{bel}(1, \text{Pacifist}), \text{bel}(2, \neg\text{Pacifist}), \text{bel}(3, \neg\neg\text{Pacifist}),
\]

\[
\text{not}_\text{bel}(3, \neg\text{Pacifist}),
\]

\[
\text{fact}(1, \text{Pacifist}), \text{fact}(2, \neg\text{Pacifist}), \text{fact}(3, \text{Pacifist}),
\]

\[
\text{fact}(3, \neg\neg\text{Pacifist}) \}
\]
Fig. 3. Correspondences between MAEL and Logic Program

\[ S' = \{ \text{bel}(1, \text{Pacifist}), \text{bel}(2, \neg \text{Pacifist}), \text{bel}(3, \neg \text{Pacifist}), \text{not} \text{bel}(3, \neg \neg \text{Pacifist}), \text{fact}(1, \text{Pacifist}), \text{fact}(2, \neg \text{Pacifist}), \text{fact}(3, \neg \text{Pacifist}) \} \] .

An extension base \( B \) which corresponds to \( S \) is

\[ B = (\{\text{Pacifist}\}, \{\neg \text{Pacifist}\}, \{\neg \neg \text{Pacifist}\}) . \]

On the other hand, an extension base \( B' \) which corresponds to \( S' \) is

\[ B' = (\{\text{Pacifist}\}, \{\neg \text{Pacifist}\}, \{\neg \neg \text{Pacifist}\}) . \]

This result coincides with Example 3.4. Since stable models of \( \text{tr}(P) \) do not exist except \( S \) and \( S' \), extension bases of \( P \) do not exist except \( B \) and \( B' \) by Theorem 4.6. Moreover, by Theorem 3.3, there does not exist extensions of \( P \) except \( T \) and \( T' \) which correspond to \( B \) and \( B' \), respectively.

All correspondences shown in this paper are illustrated in Fig. 3.

5 Discussions

In this section, we discuss the properties of reasoning in MAEL and the relations to previous studies through the translation proposed in this paper.

5.1 Relations to Junker’s Result

Junker[10] has proposed a method that reduces the computation of extensions of default logic and AEL to the one of TMS. Since it is known that semantics of TMS coincides with stable model semantics of a logic program[2], Junker’s method can be considered to be a translation method from default logic and AEL to logic programming. Taking into accounts of the characteristics of
Toyma, Kojima and Inagaki

TMS, i.e., finiteness and the lack of the logical reasoning capability, Junker showed that extensions of a theory of default logic and AEL are characterized in terms of the finite extension bases and proposed a description of justifications which can compute the extension bases. In fact, extension bases in MAEL and program rules (1) and (2) in Definition 4.1 are defined by naturally extending Junker’s result to multi-agent systems.

Junker’s TMS for computing extensions of AEL, however, does not correspond to stable model semantics of logic program since he has changed semantics of TMS so that circulation of justifications may be allowed. On the other hand, without changing both semantics of MAEL and stable model semantics of logic programs, we have associated these two systems directly in this paper. The reason that this became possible is that weakly-groundedness[11] of semantics of MAEL is described in terms of program rules (see Section 5.4).

5.2 Characterization of Extensions by Finite Theory

According to Definition 2.5, any extension of a theory of MAEL becomes the one which contains infinite number of formulas. However, if a theory is a finite set of formulas, an extension of the theory can be characterized in terms of its finite subset. Characterization of extensions in terms of extension bases defined in Definition 3.2 is a method to do this.

Although the translation method in this paper does not require the finiteness of a theory of MAEL, the translated logic program will contain infinite number of program rules if the theory is infinite. On the other hand, the computing systems for stable models[1,15] require the finiteness of a logic program. Thus, when we consider the computation of stable models of the translated logic program by using existing systems, it is desirable to generate a finite logic program for a theory whose extensions can be characterized finitely. The translation method in this paper shows that the translated logic program become finite if a theory of MAEL is finite.

5.3 Relations to Abduction

Satoh[18] has proposed a method which reduces a computation of explanations in the abductive framework to a computation of stable models. He called an atom which can be used as a hypothesis abducible, and the following rule pair is given for any abducible \( p \):

\[
p \leftarrow \neg \tilde{p}.
\]
\[
\tilde{p} \leftarrow \neg p.
\]

This rule pair divides stable models into the ones where \( p \) holds and the others where \( \tilde{p} \) holds. In the formers, \( p \) holds as a hypothesis without any other conditions.
The first two rules generated by the translation rule (3) in Definition 4.1 are in the same forms as this rule pair. In fact, the intension of giving the rules is to make a model which admits \( \text{bel}(i, p) \), that is, \( \text{bel}(i, p) \) is regarded as an abducible.

The translation method in this paper can be considered to be based on the viewpoints of abduction since the reasoning formalized in MAEL is as follows:

(i) Either a meta belief \( L_i p \) or \( \neg L_i p \) is assumed to be held.

(ii) It is verified whether the assumption in (i) satisfies the stability of a theory.

The translation rule (3) describes this view in terms of program rules, where (i) is described by using the representation method of abducibles which Satoh has proposed, and (ii) is described by using integrity constraints.

5.4 Semantics with Weakly-Groundedness

As mentioned in Section 5.3, the method to regard not only \( \neg L_i p \) but \( L_i p \) as assumptions is needed to describe weakly-groundedness\[11\] of semantics of MAEL. Roughly speaking, weakly-groundedness means that circulated justification of beliefs is allowed. For example, MAEL gives two extensions \( T = (\{\neg L_1 p, \ldots\}) \) and \( T' = (\{p, L_1 p, \ldots\}) \) for a theory \( P = (\{L_1 p \supset p\}) \). Thus, under the semantics which has weakly-groundedness, if agent \( i \) can derive \( p \) by assuming \( L_i p \), then it is valid that he has a belief \( p \). To take into accounts of this semantics, models of a logic program where the reasoning begins after once \( \text{bel}(i, p) \) is accepted were required.

6 Conclusions

In this paper, in order to develop a proof procedure of MAEL, we showed that the extension computing problem of MAEL can be reduced to the stable model computing problem of a logic program. To do this, we proposed a translation method from normal form formulas of MAEL into program rules with integrity constraints, and proved theorems which guarantee that reasoning results do not change through this translation. If this translation is available efficiently, then it comes to be able to carry out direct use of the research results about stable model computation of a logic program in the process of extension computation of a theory of MAEL.

Furthermore, to clarify the properties of reasoning in MAEL, we characterized semantics of MAEL through the translation. In particular, comparing with \[10\], we formalized the logic program that can describe the weakly-groundedness of semantics of MAEL.

As having been discussed on AEL\[11\], however, weakly-groundedness may become a factor which brings undesirable reasoning results. Since other semantics which do not have weakly-groundedness are proposed\[11,14\], applying them to MAEL is a candidate to solve the problem.
References


