ICOT Technical Report: TR-0874

TR-0874

Bottom-Up Modal Theorem Provers based on Modal Clause Transformation

by
J. Akahani (NTT), K. Inoue (Toyohashi Univ.)
& R. Hasegawa

April, 1994

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5

Institute for New Generation Computer Technology

Bottom-Up Modal Theorem Provers based on Modal Clause Transformation

Jun-ichi Akahani
NTT Communication Science Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan
+81 7749-5-1828
akahani@cslab.kecl.ntt.jp

Katsumi Inoue
Department of Information and Computer Sciences
Toyohashi University of Technology
Tempaku-cho, Toyohashi 441, Japan
+81 532-47-0111 (ext. 525)
inoue@tutics.tut.ac.jp

Ryuzo Hasegawa
ICOT
Mita Kokusai Bldg., 21F
1-4-28 Mita, Minato-ku, Tokyo 108, Japan
+81 3-3456-4365
hasegawa@icot.or.jp

Abstract

This paper presents a technique to implement efficient modal theorem provers that transforms modal formulae into input clauses of the model generation theorem prover MGTP. The technique, called the *modal clause transformation method*, is based on partial evaluation of the rewriting rules for the modal tableau method. We first propose a transformation method in which close condition testing is replaced with pattern matching in the antecedents of transformed clauses. This method avoids the generation of redundant branches that can easily be checked to be closed. Then, the method is extended by making use of goal information and by simulating top-down reasoning to reduce the search space. The cost of modal clause transformation is shown to be linear with respect to the length of the input modal formula. Finally, the results of experiments illustrate the usefulness of the method.

1 Introduction

Modal logics have been gaining popularity in various domains of Computer Science. For example, in the logics of programs [Kro87], in the analyses of distributed systems [HM90], and in the logics of knowledge, belief and intention [Kon86, Sho93]. For such applications, modal logics require fast and efficient theorem provers. Many modal theorem provers have been proposed, but only a few quantitative evaluation results have been reported. These results [Cat91, KH91] are restricted to simple theorems only. Moreover, in practical applications, we have to deal with a large set of axioms in the domain which may contain many axioms irrelevant to proofs for the given theorems. In this paper, we present an efficient modal theorem prover capable of handling such practical problems.

Current proof methods for modal logics can be classified into two categories: the *direct* approach and the *translation* approach. In the direct approach, existing proof methods for classical logic are extended for use in modal logics. Typical examples are modal tableau methods [Fit88] and modal resolution methods [AM86, Far86]. In the translation approach, on the other hand, modal formulae are translated into classical formulae [Ohl88]. The translation approach has been proven to be useful because it can be applied to various modal systems. Many proof methods have recently been proposed for this approach [Non93, CD93], which focus only on implementing *accessibility relations* to be applied to various modal systems.

The translation approach has another merit in that it can employ many control strategies developed for theorem proving in classical logic. Unfortunately, the previous proposed methods do not address the issue of controlling inference to reduce the search space.

In this paper, to take advantage of the above merit, we propose a version of the translation method, called the *modal clause transformation method*. This method translates propositional modal formulae into input clauses of the model generation theorem prover MGTP [FH91]. The MGTP is a parallel theorem prover that generates models of input clauses in a bottom-up manner. Our method is based on work by Koshimura and Hasegawa [KH91] who proposed a modal propositional tableau prover on MGTP using *meta-programming* techniques. Their prover implements the rewriting rules for the tableau method as schemata encoded as MGTP input clauses, and simulates the modal tableau method on MGTP. Thus, their system can be classified as the direct approach. We realize the modal clause transformation method on MGTP by applying a *partial evaluation* technique to the meta-programming method.

In general, modal tableau methods have a problem that they may create too many branches. To suppress the generation of redundant branches that can easily be checked to be closed, we translate modal formulae so that close condition testing is replaced with pattern matching in the antecedents

of translated clauses, instead of generating such branches. We call this translation method the *basic modal clause transformation method.*

Furthermore, to avoid the generation of branches irrelevant to the proof, we adapt the Non-Horn Magic Set (NHM) [HOI93] method that transforms input clauses so as to simulate top-down reasoning. We analyze input modal formulae to incorporate control information specific to a given input modal formula into its translated formula. The integrated translation method is called the *NHM modal clause transformation method.*

In both transformation methods, every transformed clause is *range-restricted* [MB88], that is, every variable in the clause appears in the antecedent. Since matching is sufficient instead of full unification for range-restricted clauses, efficient theorem proving is possible for transformed clauses.

In this paper, since we are interested in the transformation process of modal formulae, we will focus on the modal system K in which the accessibility relation has no special property.

This paper is organized as follows: Section 2 provides an overview of the modal tableau prover on MGTP. Section 3 presents the basic modal transformation method. We incorporate the NHM approach and describe the NHM modal transformation method in Section 4. In Section 5, we present both theoretical and experimental evaluation results. We discuss the extensions for other modal systems and future work in Section 6.

2 Modal Tableaux on MGTP

This section gives an overview of modal logics, MGTP, and the modal propositional tableau prover on MGTP.

2.1 Propositional Modal Logic

The syntax and semantics of propositional modal logics are outlined below. *Formulae* are constructed from a non-empty set Φ of propositional symbols, logical connectives $\neg$ and $\vee$, and the modal operator $\Box$. We add the connectives $\wedge$ and $\supset$, and the modal operator $\Diamond$, their meanings being as usual.

We use a standard possible world semantics [Che80]. A Kripke model M is a triple $< W, R, V >$, where W is a non-empty set of worlds, R is a binary accessibility relation on W, and $V : \Phi \rightarrow 2^W$ is a function that assigns, to each propositional symbol $p \in \Phi$, a subset $V(p)$ of W.

We write $M, w \models \varphi$ for "a formula φ is true at a world w in a model M" defined as follows:

1. $M, w \models p$ (for $p \in \Phi$) iff $w \in V(p)$.
2. $M, w \models \neg\varphi$ iff $M, w \not\models \varphi$.

3. $M, w \models \varphi \vee \psi$ iff $M, w \models \varphi$ or $M, w \models \psi$.

4. $M, w \models \Box\varphi$ iff $M, w' \models \varphi$ for all w' such that wRw'.

We say that φ is *valid* if $M, w \models \varphi$ for every $w \in W$ and every model M. φ is said to be *satisfiable* if $M, w \models \varphi$ for some $w \in W$ and some M; φ is said to be *unsatisfiable* otherwise. It is easy to check that φ is valid iff $\neg\varphi$ is unsatisfiable.

Various modal systems can be defined according to the properties associated with accessibility relations. This paper, however, focuses on the most fundamental modal system, known as K, in which the accessibility relation has no special property.

2.2 MGTP

The MGTP [FH91] is a parallel theorem prover that generates models of input clauses in a bottom-up manner. Each clause is expressed in the form:

$$A_1, \cdots, A_n \rightarrow C_{1,1}, \cdots, C_{1,l_1} | \cdots | C_{m,1}, \cdots, C_{m,l_m},$$

where $n, m \geq 0, l_j \geq 1 (j = 1, \cdots, m)$, and $A_i (i = 1, \cdots, n)$ and $C_{j,k} (j = 1, \cdots, m; k = 1, \cdots, l_j)$ are atoms. All variables are assumed to be universally quantified at the front of the clause. The clause shown above represents the formula $A_1 \wedge \cdots \wedge A_n \supset (C_{1,1} \wedge \cdots \wedge C_{1,l_1}) \vee \cdots \vee (C_{m,1} \wedge \cdots \wedge C_{m,l_m})$ in the standard notation of first-order logic. The left-hand side of $\rightarrow$ is called the *antecedent* of the clause, while the right-hand side is called the *consequent*.

When $n = 0$, we write $true \rightarrow C_{1,1}, \cdots, C_{1,l_1} \mid \cdots \mid C_{m,1}, \cdots, C_{m,l_m}$, and the clause is called a *positive clause*. Similarly, if $m = 0$, we write $A_1, \cdots, A_n \rightarrow false$, and the clause is called a *negative clause*. Otherwise, the clause is called a *mixed clause*.

The MGTP applies the following operations to a set of atoms called a *model candidate*. An empty set is given as an initial model candidate.

- Model candidate extension: If there is a non-negative clause

  $$A_1, \cdots, A_n \rightarrow C_{1,1}, \cdots, C_{1,l_1} | \cdots | C_{m,1}, \cdots, C_{m,l_m}$$

  and a substitution σ, such that all atoms $A_1\sigma, \cdots, A_n\sigma$ are satisfied by a model candidate M and that for all $j = 1, \cdots, m$, some $C_{j,k}\sigma (1 \leq k \leq l_j)$ is not satisfied by M, then expand M in m ways by adding a set $\{C_{j,1}\sigma, \cdots, C_{j,l_j}\sigma\}$ to M for every $j = 1, \cdots, m$.

- Model candidate rejection: If there is a negative clause

  $$A_1, \cdots, A_n \rightarrow false$$

  and a substitution σ, such that all atoms $A_1\sigma, \cdots, A_n\sigma$ are satisfied by a model candidate M, then discard M.

If no operation can be applied to any model candidate, then MGTP returns the model candidate sets as *models* of the input clauses. If every model candidate is rejected, then the set of input clauses is *unsatisfiable*.

The process of obtaining a substitution σ is called "*conjunctive matching* of the antecedent against elements in M". The conjunctive matching process does not require full unification if every clause is *range-restricted* [MB88], that is, if every variable in the clause appears in the antecedent. In this case, since every model candidate constructed by MGTP contains only ground atoms, it is sufficient to consider matching instead of full unification.

2.3 Modal Propositional Tableaux on MGTP

The modal tableau method [Fit88] is a refutation proof method which rewrites *signed formulae*, that is, formulae prefixed with either t or f, intuitively denoting whether the following formula is true or false. A signed formula $t(\varphi, w)$ indicates that a formula φ is true in world w, and $f(\varphi, w)$ indicates that φ is false in w.

Koshimura and Hasegawa [KH91] described a modal propositional tableau prover on MGTP using meta-programming techniques. The prover implements the rewriting rules of the tableau method as schemata encoded as MGTP input clauses, and simulates the modal tableau method on MGTP. In this paper, we refer to this method as the *meta-programming* method.

Figure 1 shows MGTP input clauses that represent tableau rules. Symbols starting with a capital letter denote variables in MGTP. Predicate $path(W, V)$ denotes that world V is accessible from world W, and $\{new\_world(V)\}$ is a call for the built-in predicate $new\_world(V)$, which creates a new symbol corresponding to world V. Note that all of the clauses are range-restricted.

It is possible to extend the prover for many modal systems other than K by adding clauses that represent the corresponding properties of accessibility relations. In fact, a prover for PTL [Kro87] was implemented with the meta-programming method [KH91]. It is also possible to extend the prover for multi-modal systems by providing an extra argument indicating a modal operator for the *path* predicate.

Given modal formula φ, the prover generates a set $\Sigma(\varphi)$ of MGTP input clauses that consists of clause

$$true \rightarrow f(\varphi, w_0),$$

and the clauses shown in Figure 1. As the method is a direct implementation of the modal tableau method, we have:

Lemma 1 *A formula φ is valid iff MGTP returns the result* unsatisfiable *for the set $\Sigma(\varphi)$ of input clauses.*

- α rules.
  $t(P \wedge Q, W) \rightarrow t(P, W), t(Q, W).$
  $f(P \vee Q, W) \rightarrow f(P, W), f(Q, W).$
  $f(P \supset Q, W) \rightarrow t(P, W), f(Q, W).$
  $t(\neg P, W) \rightarrow f(P, W).$
  $f(\neg P, W) \rightarrow t(P, W).$

- β rules.
  $t(P \vee Q, W) \rightarrow t(P, W) \mid t(Q, W).$
  $f(P \wedge Q, W) \rightarrow f(P, W) \mid f(Q, W).$
  $t(P \supset Q, W) \rightarrow f(P, W) \mid t(Q, W).$

- close condition.
  $f(P, W), t(P, W) \rightarrow false.$

- π rule.
  $f(\Box P, W) \rightarrow \{new\_world(V)\}, path(W, V), f(P, V).$

- ν rule.
  $t(\Box P, W), path(W, V) \rightarrow t(P, V).$

Figure 1: MGTP clauses used to implement the Modal Tableau Method

3 Modal Clause Transformation Method

This section presents a basic algorithm for the modal clause transformation method. Since the method described in the previous section simulates tableau method on MGTP, the prover tends to create too many branches. We therefore apply a partial evaluation technique to the meta-programming method, thereby suppressing the generation of branches which can easily be checked to be closed.

3.1 Modal Clause

In the following, we will represent modal formulae as *modal clauses*. Formally, the syntax of modal clauses is defined as follows.

1. If $p \in \Phi$, p is a *modal atom*, called a *propositional atom*.

2. If $\varphi_1, \cdots, \varphi_n, \psi_1, \cdots, \psi_m$ are modal atoms, $\neg\varphi_1 \vee \cdots \vee \neg\varphi_n \vee \psi_1 \vee \cdots \vee \psi_m$ is a modal clause, where $n, m \geq 0$.

3. If φ is a modal clause, $\Box\varphi$ is a modal atom.

Rule 2 implies that modal atoms are modal clauses. We sometimes write the formula shown in Rule 2 as $\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m$. The left-hand

side $\varphi_1 \wedge \cdots \wedge \varphi_n$ of $\supset$ is called the antecedent, and the right-hand side $\psi_1 \vee \cdots \vee \psi_m$ is called the consequent. For example, let $p, q, r \in \Phi$, then $\Box(\Box p \supset q)$ is a modal atom, and $\Box p \wedge \Box(\Box p \supset q) \supset \Box q \vee \Box\Box r$ is a modal clause.

The definition of modal clauses is similar to that of Enjalbert and Farinas del Cerro [EF89]. They do not allow negations in the clause except in the front of propositional atoms but allow $\Diamond$ followed by a conjunction of clauses, while we do not allow $\Diamond$ but allow negations in front of the modal operators. For example, the above modal clause can be written in the form $\Diamond\neg p \vee \Diamond(\Box p \wedge \neg q) \vee \Box q \vee \Box\Box r$.

We represent the conjunction of modal clauses as a set of modal clauses. Any modal formula can be represented as a set of modal clauses. This can be proved by induction on the structure of modal formulae.

3.2 Basic Transformation

The basic idea of the modal clause transformation method is to apply *partial evaluation* to the meta-programming method. Specifically, given a set of modal clauses, the translator first tries to apply the α rules or β rules to the modal clause set. The result consists of *signed modal atoms*, that is, MGTP atoms in the form of either $t(\varphi, w)$ or $f(\varphi, w)$ where φ is a modal atom. If there is a signed modal atom to which the π rule (or the ν rule) can be applied, an MGTP clause that is a specialization of the π rule (or the ν rule) is generated.

For example, given a modal clause $\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m$, the meta-programming method would generate an MGTP clause of the form

$$true \rightarrow f(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m, w_0).$$

The α rules can be applied to the above clause; the result of this application would be:

$$true \rightarrow t(\varphi_1, w_0), \cdots, t(\varphi_n, w_0), f(\psi_1, w_0), \cdots, f(\psi_m, w_0),$$

where each φ_i and ψ_j are modal atoms, i.e. either propositional atoms or modal formulae of the form $\Box\phi$. In the latter case, the π rule or ν rule can be applied to the modal atom.

Now, imagine there is a signed modal atom $f(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m, X)$. Then, the π rule can be applied, and the result would be

$$f(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W) \\ \quad \rightarrow \{new\_world(V)\}, path(W, V), f(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m, V).$$

The α rules are applicable to the consequent. We therefore have

$$f(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W) \\ \quad \rightarrow \{new\_world(V)\}, path(W, V), \\ \quad t(\varphi_1, V), \cdots, t(\varphi_n, V), f(\psi_1, V), \cdots, f(\psi_m, V).$$

1. For each modal atom φ_i in the antecedent, generate the following clause.

   $true \rightarrow t(\varphi_i, w_0)$.

   For each modal atom ψ_i in the consequent, generate the following clause.

   $true \rightarrow f(\psi_j, w_0)$.

2. If the consequent of a generated clause contains atoms of the form $f(\Box\varphi, X)$ or $t(\Box\varphi, X)$, repeat the following.

   (a) If the atom is of the form $f(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), X)$, generate the following clause (*specialization of the π rule*).

   $$f(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W) \rightarrow \{new\_world(V)\}, path(W, V), t(\varphi_1, V), \cdots, t(\varphi_n, V), f(\psi_1, V), \cdots, f(\psi_m, V).$$

   (b) If the atom is of the form $t(\Box(\Box\varphi_1 \wedge \cdots \wedge \Box\varphi_n \wedge p_1 \wedge \cdots \wedge p_k \supset \psi_1 \vee \cdots \vee \psi_m), X)$, where each p_i is a propositional atom, then generate the following clause (*specialization of the ν rule*).

   $$t(\Box(\Box\varphi_1 \wedge \cdots \wedge \Box\varphi_n \wedge p_1 \wedge \cdots \wedge p_k \supset \psi_1 \vee \cdots \vee \psi_m), W), path(W, V), t(p_1, V), \cdots, t(p_k, V) \rightarrow f(\Box\varphi_1, V) \mid \cdots \mid f(\Box\varphi_n, V) \mid t(\psi_1, V) \mid \cdots \mid t(\psi_m, V).$$

3. Generate the following clause that denotes the close condition.

   $t(P, W), f(P, W) \rightarrow false$.

Figure 2: Algorithm for the Basic Modal Clause Transformation Method

1. For each modal atom in the antecedent and the consequent, generate the following clauses.
   $true \rightarrow t(\Box(p \wedge q \wedge \Box\neg r \supset s \vee r), w_0)$.
   $true \rightarrow t(\Box p, w_0)$.
   $true \rightarrow t(\Box\Box(q \supset r), w_0)$.
   $true \rightarrow t(\Box\neg s, w_0)$.
   $true \rightarrow f(\Box(\Box q \wedge q \supset r), w_0)$.

2-a. For atoms of the form $f(\Box\varphi, X)$ in the consequent of a generated clause, generate the following clause.
   $f(\Box(\Box q \wedge q \supset r), W)$
   $\rightarrow \{new\_world(V)\}, path(W, V), t(\Box q, V), t(q, V), f(r, V)$.

2-b. For atoms of the form $t(\Box\varphi, X)$ in the consequent of a generated clause, generate the following clauses.
   $t(\Box(p \wedge q \wedge \Box\neg r \supset s \vee r), W), path(W, V), t(p, V), t(q, V)$
   $\rightarrow f(\Box\neg r, V) \mid t(s, V) \mid t(r, V)$.
   $t(\Box p, W), path(W, V) \rightarrow t(p, V)$.
   $t(\Box\Box(q \supset r), W), path(W, V) \rightarrow t(\Box(q \supset r), V)$.
   $t(\Box\neg s, W), path(W, V), t(s, V) \rightarrow false$.

2'. Repeat Step 2.
   $f(\Box\neg r), W) \rightarrow \{new\_world(V)\}, path(W, V), f(r, V)$.
   $t(\Box q, W), path(W, V) \rightarrow t(q, V)$.
   $t(\Box(q \supset r), W), path(W, V), t(q, V) \rightarrow t(r, V)$.

3. Generate the clause denoting the close condition.
   $t(P, W), f(P, W) \rightarrow false$.

Figure 3: Example of Basic Modal Clause Transformation

It is similar when a signed modal atom $t(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m, X)$ appears. Applying the ν rule to the modal atom and then applying the β rules to the consequent of the resulting clause, gives

$$t(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W), path(W, V) \\ \quad \rightarrow f(\varphi_1, V) \mid \cdots \mid f(\varphi_n, V) \mid t(\psi_1, V) \mid \cdots \mid t(\psi_m, V).$$

This MGTP clause generates too many branches. For example, a model candidate containing $t(\Box(p \wedge q \wedge \Box r \supset s), w_1)$ and $path(w_1, w_2)$ is expanded in four ways corresponding to $f(p, w_2)$, $f(q, w_2)$, $f(\Box r, w_2)$, and $t(s, w_2)$, even if the model candidate contains $t(p, w_2)$ or $t(q, w_2)$. If we check the satisfiability of $t(p, w_2)$ and $t(q, w_2)$ before model extension, we can avoid generating branches corresponding to $f(p, w_2)$ and $f(q, w_2)$. We therefore transform the clause to

$$t(\Box(p \wedge q \wedge \Box r \supset s), W), path(W, V), t(p, V), t(q, V) \rightarrow f(\Box r, V) \mid t(s, V).$$

Note that we cannot move $f(\Box r, V)$ to the antecedent because the atom may invoke the π rule. It can be proven that this transformation preserves the satisfiability.

Now, the specialization of the ν rule is defined as follows. If $t(\Box(\Box\varphi_1 \wedge \cdots \wedge \Box\varphi_n \wedge p_1 \wedge \cdots \wedge p_k \supset \psi_1 \vee \cdots \vee \psi_m), X)$ appears, we generate the following clause.

$$t(\Box(\Box\varphi_1 \wedge \cdots \wedge \Box\varphi_n \wedge p_1 \wedge \cdots \wedge p_k \supset \psi_1 \vee \cdots \vee \psi_m), W), \\ \quad path(W, V), t(p_1, V), \cdots, t(p_k, V) \\ \quad \rightarrow f(\Box\varphi_1, V) \mid \cdots \mid f(\Box\varphi_n, V) \mid t(\psi_1, V) \mid \cdots \mid t(\psi_m, V).$$

The algorithm for the basic modal transformation method is shown in Figure 2. For the sake of simplicity, we assume that a modal clause $\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m$ to be proven is given in Figure 2. Figure 3 shows the transformed clauses for modal clause $\Box(p \wedge q \wedge \Box\neg r \supset s \vee r) \wedge \Box p \wedge \Box\Box(q \supset r) \wedge \Box\neg s \supset \Box(\Box q \wedge q \supset r)$.

If a set of modal clauses $\{\mathcal{C}_1, \cdots, \mathcal{C}_n\}$ is given, we first generate the following clause

$$true \rightarrow c_1 \mid \cdots \mid c_n,$$

where each c_i corresponds to modal clause $\mathcal{C}_i$. Then, for each modal clause $\mathcal{C}_i$, we apply the transformation obtained by replacing $true$ with c_i in Step 1 of Figure 2. These clauses are actually obtained by applying the β rules to the MGTP clause $true \rightarrow f(\mathcal{C}_1 \wedge \cdots \wedge \mathcal{C}_n, w_0)$.

The next theorem assures the correctness of the basic modal transformation method.

Theorem 1 *The basic modal transformation method is a sound and complete proof method for modal system K.*

4 NHM Modal Clause Transformation Method

In practical applications, formulae usually contain many subformulae irrelevant to the proof. For example, consider formula $\Box(p \supset a \vee b \vee c) \wedge \Box(p \supset q) \wedge \Box p \supset \Box q$. The first element in the antecedent contributes nothing to the proof, but irrelevant model candidates $\{\Box a\}$, $\{\Box b\}$, and $\{\Box c\}$ might be generated from a model candidate containing $\Box p$. Thus, the control of proof processes is required to prove theorems efficiently, especially in bottom-up proof methods such as the tableau method and MGTP.

To overcome the drawback of bottom-up provers, the notion of the Non-Horn Magic Set (NHM) [HOI93] was proposed. The NHM method transforms input clauses so as to simulate top-down reasoning. Specifically, clause

$$A_1, \cdots, A_n \rightarrow C_1 \mid \cdots \mid C_m$$

is transformed into clauses:

$$goal(C_1), \cdots goal(C_m) \rightarrow goal(A_1), \cdots, goal(A_n),$$

$$goal(C_1), \cdots goal(C_m), A_1, \cdots, A_n \rightarrow C_1 \mid \cdots \mid C_m,$$

where the first clause is not generated if $n = 0$. The atom $goal(A)$ denotes that atom A should be proven. Thus, the first clause intuitively means that if the consequent of the original clause has to be proven then the antecedent must be proven first. The intuitive meaning of the second clause is that the original clause shall be invoked only when its consequent has to be proven.

We can regard label f as *goal* to incorporate the NHM method into the basic modal transformation method. However, replacing f with *goal* would cause the close condition to be

$$t(P, W), goal(P, W) \rightarrow false$$

which contradicts with the second clause generated by the NHM method. Hence, we have to eliminate the close condition

To eliminate the close condition, we have to reexamine the specialization of the π rule. Recall that the π rule has two roles. One is the propagation of goal information: if $\Box\varphi$ has to be proven in world W then we must create a world V accessible from W and then prove φ in V. The other is the propagation of negative information: if $\neg\Box\varphi$ holds in W then $\neg\varphi$ holds in each V accessible from W. Regarding label f as *goal* for the specialization of the π rule in Figure 2, we have the following clause.

$$\begin{aligned} &goal(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W) \\ &\quad \rightarrow \{new\_world(V)\}, path(W, V), t(\varphi_1, V), \cdots, t(\varphi_n, V), \\ &\quad goal(\psi_1, V), \cdots, goal(\psi_m, V). \end{aligned}$$

1. For each modal atom φ_i in the antecedent, generate the following clause.

   $goal(\varphi_i, w_0) \rightarrow t(\varphi_i, w_0)$.

   For each modal atom ψ_i in the consequent, generate the following clauses.

   $true \rightarrow goal(\psi_j, w_0)$.
   $t(\psi_j, w_0) \rightarrow false$.

2. If the consequent of a generated clause contains atoms of the form $goal(\Box\varphi, X)$ or $t(\Box\varphi, X)$, then repeat the following.

   (a) If the atom is of the form $goal(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), X)$, generate the following $m+1$ clauses, where $j = 1, \cdots, m$.

   $goal(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W)$
   $\rightarrow \{new\_world(V)\}, path(W, V), t(\varphi_1, V), \cdots, t(\varphi_n, V),$
   $goal(\psi_1, V), \cdots, goal(\psi_m, V)$.
   $t(\psi_j, V), path(W, V) \rightarrow t(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W)$.

   (b) If the atom is of the form $t(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), X)$,

   i. If $m = 0$, that is, the atom is $t(\Box(\neg\varphi_1 \vee \cdots \vee \neg\varphi_n), X)$, then generate the following 3 clauses.

   $path(W, V) \rightarrow goal(\Box(\neg\varphi_1 \vee \cdots \vee \neg\varphi_n), W)$.
   $path(W, V), t(\Box(\neg\varphi_1 \vee \cdots \vee \neg\varphi_n), W)$
   $\rightarrow goal(\varphi_1, V), \cdots, goal(\varphi_n, V)$.
   $path(W, V), t(\Box(\neg\varphi_1 \vee \cdots \vee \neg\varphi_n), W), t(\varphi_1, V), \cdots, t(\varphi_n, V)$
   $\rightarrow false$.

   ii. If $m > 0$, generate the following 2 clauses.

   $path(W, V), goal(\psi_1, V), \cdots, goal(\psi_m, V)$
   $\rightarrow goal(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W),$
   $goal(\varphi_1, V), \cdots, goal(\varphi_n, V)$.
   $path(W, V), goal(\psi_1, V), \cdots, goal(\psi_m, V),$
   $t(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W),$
   $t(\varphi_1, V), \cdots, t(\varphi_n, V), \rightarrow t(\psi_1, V) \mid \cdots \mid t(\psi_m, V)$.

Figure 4: Algorithm for the NHM Modal Clause Transformation Method

1. For each modal atom in the antecedent and the consequent, generate the following clauses.
   $goal(\Box(p \supset a \vee b \vee c), w_0) \rightarrow t(\Box(p \supset a \vee b \vee c), w_0).$
   $goal(\Box p, w_0) \rightarrow t(\Box p, w_0).$
   $goal(\Box(p \wedge q \supset r \vee s), w_0) \rightarrow t(\Box(p \wedge q \supset r \vee s), w_0).$
   $true \rightarrow goal(\Box(q \supset r \vee s), w_0).$
   $t(\Box(q \supset r \vee s), w_0) \rightarrow false.$

2-a. For atoms of the form $goal(\Box\varphi, X)$ in the consequent of a generated clause, generate the following clauses.
   $goal(\Box(q \supset r \vee s), W)$
   $\quad \rightarrow \{new\_world(V)\}, path(W, V), t(q, V), goal(r, V), goal(s, V).$
   $t(r, V), path(W, V) \rightarrow t(\Box(q \supset r \vee s), W).$
   $t(s, V), path(W, V) \rightarrow t(\Box(q \supset r \vee s), W).$

2-b. For atoms of the form $t(\Box\varphi, X)$ in the consequent of a generated clause, generate the following clauses.
   $path(W, V), goal(a, V), goal(b, V), goal(c, V)$
   $\quad \rightarrow goal(\Box(p \supset a \vee b \vee c), W), goal(p, V).$
   $path(W, V), goal(a, V), goal(b, V), goal(c, V),$
   $\quad t(\Box(p \supset a \vee b \vee c), W), t(p, V) \rightarrow t(a, V) \mid t(b, V) \mid t(c, V).$
   $path(W, V), goal(p, V) \rightarrow goal(\Box p, W).$
   $path(W, V), goal(p, V), t(\Box p, W) \rightarrow t(p, V).$
   $path(W, V), goal(r, V), goal(s, V)$
   $\quad \rightarrow goal(\Box(p \wedge q \supset r \vee s), W), goal(p, V), goal(q, V).$
   $path(W, V), goal(r, V), goal(s, V),$
   $\quad t(\Box(p \wedge q \supset r \vee s), W), t(p, V), t(q, V) \rightarrow t(r, V) \mid t(s, V).$

Figure 5: Example of NHM Modal Clause Transformation

As the clause only propagates goal information, we have to encode the propagation of negative information.

Since the propagation of negative information can be represented as the formula

$$f(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W) \wedge path(W, V)$$

$$\supset t(\varphi_1, V) \wedge \cdots \wedge t(\varphi_n, V) \wedge f(\psi_1, V) \wedge \cdots \wedge f(\psi_m, V),$$

we have

$$f(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W) \wedge path(W, V) \supset f(\psi_j, V)$$

for each $j = 1, \cdots, m$. We transform the formula to a logically equivalent clause of the form

$$t(\psi_j, V), path(W, V) \rightarrow t(\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m), W).$$

Now that negative information has been eliminated from the translated clauses. Thus, the close condition becomes unnecessary.

The specialization of the π rule has the following meaning: " If $\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m)$ has to be proven in world W, then we must create a world V accessible from W, set each φ_i true in V, and prove each ψ_j in V. If any ψ_j is proven in V, $\Box(\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m)$ is true in world W."

Figure 4 shows the algorithm for the NHM modal transformation method. We assume that modal clause $\varphi_1 \wedge \cdots \wedge \varphi_n \supset \psi_1 \vee \cdots \vee \psi_m$ is given, as in Figure 2. The clauses in Step 1 of Figure 4 can be obtained by applying the NHM method described above to $true \rightarrow t(\varphi_i, w_0)$ and $t(\psi, w_0) \rightarrow false$, respectively. As mentioned before, Step 2(a) is a specialization of the π rule. Step 2(b)ii is the result of direct application of the NHM method to Step 2(b) of Figure 2. Note that any special treatment for the antecedent is now unnecessary. Step 2(b)i is a special case of Step 2(b); since the first clause of Step 2(b)ii generates irrelevant goal information if $m = 0$, the clause is divided into two clauses.

Figure 5 shows the result of NHM modal clause transformation for modal clause $\Box(p \supset a \vee b \vee c) \wedge \Box p \wedge \Box(p \wedge q \supset r \vee s) \supset \Box(q \supset r \vee s)$. For the clause set, MGTP first generates a model candidate containing $goal(\Box(q \supset r \vee s), w_0)$, then creates a new world, say w_1, and expands the model candidate by adding $goal(r, w_1)$, $goal(s, w_1)$. These goals generate a goal $goal(\Box(p \wedge q \supset r \vee s), w_0)$ in w_0. Note that $t(\Box(p \wedge q \supset r \vee s), w_0)$ is generated only by the demands in the world w_1. Atom $t(\Box(p \supset a \vee b \vee c), w_0)$ is never generated because it is not in demand. Hence, we can avoid generating branches irrelevant to the proof.

5 Results of Experiments

In this section, we describe both the theoretical and experimental analyses for the computational cost of the modal clause transformation.

The cost of the modal clause transformation can be divided into the cost of analyzing input modal clauses and the cost of generating MGTP clauses. The former is linear to the length of the input modal clauses since the analysis is performed by scanning the input modal clauses once only. It is easy to see from the transformation algorithm that the latter is linear to the number of modal operators in the input clauses. Thus, the cost of the modal clause transformation is linear to the length of the input modal clauses.

We tested the prover on several modal formulae. Table 1 lists our results for some examples. In the table, *meta*, *peval*, and *nhm* denote the meta-programming method described in Section 2, the basic modal transformation method in Section 3, and the NHM modal transformation method in Section 4, respectively. The number of branches shows the width of the search tree, and the average proof length shows the depth of the search tree. The CPU time is given in seconds.

We used the MGTP system on one processor of the PIM/m parallel inference machine, developed at ICOT. Some examples used in the experiment are shown below. Ex1 and Ex5 are examples having multi modal operators.

Ex1. The 3 wise men problem

$$\Box_a(dot(a) \vee \Box_b\Box_c\neg dot(a)) \wedge$$
$$\Box_a\Box_b(dot(b) \vee \Box_c\neg dot(b)) \wedge$$
$$\Box_a\Box_b\Box_c(dot(a) \vee dot(b) \vee dot(c)) \wedge$$
$$\Box_a\Box_b\neg\Box_c dot(c) \wedge \Box_a\neg\Box_b dot(b) \supset \Box_a dot(a).$$

Ex2. A formula whose antecedent contains many literals

$$\Box(p_1 \wedge p_2 \wedge \cdots \wedge p_{50} \supset q \vee r) \wedge \Box p_1 \wedge \Box p_2 \wedge \cdots \wedge \Box p_{50} \supset \Box(q \vee r).$$

Ex3. A formula which produces long inference chains

$$\Box p_1 \wedge \Box(p_1 \supset p_2) \wedge \Box(p_2 \supset p_3) \wedge \cdots \wedge \Box(p_{49} \supset p_{50}) \wedge \Box(p_{50} \supset q \vee r) \supset \Box(q \vee r).$$

Ex4. A formula which may produce irrelevant branches

$$\Box(p \supset p_1 \vee p_2 \vee p_3) \wedge \Box(p \supset p_4 \vee p_5 \vee p_6) \wedge \Box(p \supset p_7 \vee p_8 \vee p_9) \wedge \Box(p \supset p_{10} \vee p_{11} \vee p_{12}) \wedge \Box(p \supset p_{13} \vee p_{14} \vee p_{15}) \wedge \Box(p \supset q \vee r) \wedge \Box p \supset \Box(q \vee r).$$

Ex5. A formula which may produce irrelevant worlds

$$\Box_a(\Box_{a1}p \wedge \Box_{a2}p \wedge \Box_{a3}p \wedge \Box_{a4}p \wedge \Box_{a5}p \supset p) \wedge$$
$$\Box_a(\Box_{a6}p \wedge \Box_{a7}p \wedge \Box_{a8}p \wedge \Box_{a9}p \wedge \Box_{a10}p \supset p) \wedge$$
$$\Box_a(\Box_{a11}p \wedge \Box_{a12}p \wedge \Box_{a13}p \wedge \Box_{a14}p \wedge \Box_{a15}p \supset p) \wedge$$
$$\Box_a(\Box_b p \supset q \vee r) \wedge \Box_a\Box_b p \supset \Box_a(q \vee r).$$

Table 1: Evaluation Results

		meta	peval	nhm
Ex1	number of branches	5	5	5
	average proof length	34	20	37
	CPU time	0.08	0.06	0.24
Ex2	number of branches	51	2	2
	average proof length	184	106	208
	CPU time	1.11	0.89	4.04
Ex3	number of branches	52	2	2
	average proof length	187	108	212
	CPU time	1.08	0.82	4.27
Ex4	number of branches	607	486	2
	average proof length	40	18	12
	CPU time	1.57	0.92	0.05
Ex5	number of branches	312	468	2
	average proof length	37	20	15
	CPU time	1.80	1.65	0.11
Ex6	number of branches	3883	486	2
	average proof length	71	36	48
	CPU time	13.78	2.39	0.45

Ex6. A formula which has antecedents with many literals and which may produce irrelevant branches
$\Box(p_1 \wedge p_2 \wedge \cdots \wedge p_{50} \supset a_1 \vee b_1 \vee c_1) \wedge \cdots \wedge \Box(p_1 \wedge p_2 \wedge \cdots \wedge p_{50} \supset a_5 \vee b_5 \vee c_5)$
$\wedge \Box(p_1 \wedge p_2 \wedge \cdots \wedge p_{50} \supset q \vee r) \wedge \Box p_1 \wedge \Box p_2 \wedge \cdots \wedge \Box p_{50} \supset \Box(q \vee r)$.

As can be seen from the table, the basic modal transformation method (peval) is generally superior to the meta-programming method (meta). In particular, for Ex6, the CPU time of the basic transformation method is reduced to 1/6 that of the meta-programming method. The reason is that the meta-programming method generates branches for every 50 atoms in the antecedents, while the basic transformation method processes conjunctive matching of the antecedents instead of generating branches.

The merit of the NHM transformation method (nhm) can be seen in the number of branches it generates. This merit becomes more important as the program size increases. For Ex4, Ex5, and Ex6, the NHM transformation method is much better than the other two methods since it does not generate irrelevant branches as shown in the table. In particular, the CPU time of the NHM transformation method is reduced to 1/30 that of the meta-programming method for Ex6.

In Ex2 and Ex3, both the basic transformation method and the NHM

transformation method generate the same number of branches, say only 2 branches, while the meta-programming method produces more than 50 branches. This is because these examples do not contain irrelevant subformulae. For Ex2 and Ex3, however, the NHM transformation method is no better than the meta-programming method in terms of CPU time. This is caused by the overhead of goal information propagation. For Ex2, as the number of input clauses and that of predicates in the NHM method are double those of the basic method, the cost of conjunctive matching increases linearly. The problem, however, can be overcome by introducing the clause indexing method which enables efficient conjunctive matching[1].

6 Concluding Remarks

We have presented the modal transformation method, that translates propositional modal formulae into input clauses of the model generation theorem prover MGTP. The main idea of the modal transformation method is partial evaluation of modal tableau rewriting rules, in particular, specialization of the π rule and the ν rule.

So far, there have been few reports on efficient strategies for modal theorem proving. Auffray et al. [AEH90] proposed a modal version of resolution strategies such as input and linear resolution. These strategies, however, impose restrictions on modal formulae, the so-called *modal Horn clauses*, while our method can be applied to any modal formula.

From the viewpoint of the modal clause transformation method, all previously proposed translation methods compute the π rule and the ν rule completely. Compared with previously proposed translation methods, the modal clause transformation method offers the following advantages.

1. Translated clauses are range-restricted. Hence, efficient theorem proving is possible, as matching is sufficient instead of full unification.

2. It is possible to restrict the invocation of the π rule that creates new worlds. We can therefore avoid generating branches irrelevant to the proof.

In this paper, we have concentrated on the modal system K in which the accessibility relation has no special property. As the proposed method can be classified as a translation method, it is obviously possible to make use of techniques for implementing the accessibility relation described in the literature [Ohl88, Non93].

Although MGTP is a parallel prover, this paper has not addressed the issue of parallelism. Our future intentions are to investigate parallel versions

[1] In fact, the NHM transformation method works better than the meta-programming method for Ex2 and Ex3 on the Prolog version of MGTP, which implements a clause indexing method.

of the modal clause transformation method by exploiting the parallelism inherent in modal theorem proving.

Modal logics have been recognized to be important in Artificial Intelligence, especially in nonmonotonic logics of belief and in accounts of abduction and belief revision based on nonstandard logic. With this regard, MGTP has been proven to be very useful for computing default negation (negation as failure) in logic programming [IKH92] and abduction based on first-order logic [IOHN93]. By combining our work with these MGTP-based approaches, we expect that more general forms of nonmonotonic and abductive reasoning based on modal logics of belief can be treated on MGTP.

Acknowledgements

We are grateful to Miyuki Koshimura for his help with the experiments and to Hiroshi Fujita for providing the MGTP compiler in Prolog. We would also like to thank Seishi Nishikawa, Tsukasa Kawaoka, Ryohei Nakano, and Shun-ichi Uchida for their support in this work.

References

[AEH90] Y. Auffray, P. Enjalbert, and J-J. Hebrard. Strategies for modal resolution: Results and problems. *J. of Automated Reasoning*, 6:1–38, 1990.

[AM86] M. Abadi and Z. Manna. Modal theorem proving. In *Proc. of the 8th Int. Conf. on Automated Deduction*, pages 172–189, 1986.

[Cat91] L. Catach. TABLEAUX: A general theorem prover for modal logics. *J. of Automated Reasoning*, 7:489–510, 1991.

[CD93] R. Caferra and S. Demri. Cooperation between direct method and translation method in non classical logics: Some results in propositional S5. In *Proc. of IJCAI'93*, pages 74—79, 1993.

[Che80] B. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, 1980.

[EF89] P. Enjalbert and L. Farinas del Cerro. Modal resolution in clausal form. *Theoretical Computer Science*, 65:1–33, 1989.

[Far86] L. Farinas del Cerro. Molog, a system that extends Prolog with modal logic. *New Generation Computing*, 4:35–50, 1986.

[FH91] H. Fujita and R. Hasegawa. A model generation theorem prover in kl1 using ramified-stack algorithm. In *Proc. of ICLP'91*, pages 535–548, 1991.

[Fit88] M. Fitting. First-order modal tableaux. *J. of Automated Reasoning*, 4:191–213, 1988.

[HM90] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *JACM*, 37:549–587, 1990.

[HOI93] R. Hasegawa, Y. Ohta, and K. Inoue. Non-horn magic sets and their relation to relevancy testing. Technical Report 834, ICOT, 1993.

[IKH92] K. Inoue, M. Koshimura, and R. Hasegawa. Embedding negation as failure into a model generation theorem prover. In *Proc. of the 11th Int. Conf. on Automated Deduction*, pages 400–415, 1992.

[IOHN93] K. Inoue, Y. Ohta, R. Hasegawa, and M. Nakashima. Bottom-up abduction by model generation. In *Proc. of IJCAI'93*, pages 102–108, 1993.

[KH91] M. Koshimura and R. Hasegawa. Modal propositional tableaux on a model generation theorem prover. In *Proc. of LPC'91*, pages 43–52, 1991.

[Kon86] Kurt Konolige. *A Deduction Model of Belief*. Morgan Kaufmann, 1986.

[Kro87] F. Kroger. Temporal logic of programs. In *EATCS Monographs on Theoretical Computer Science*, pages 1–148. Spinger-Verlag, 1987.

[MB88] R. Manthey and F. Bry. SATCHMO: A theorem prover implemented in Prolog. In *Proc. of the 9th Int. Conf. on Automated Deduction*, pages 415–434, 1988.

[Non93] A. Nonnengart. First-order modal logic theorem proving and functional simulation. In *Proc. of IJCAI'93*, pages 80—85, 1993.

[Ohl88] H. J. Ohlbach. A resolution calculus for modal logics. In *Proc. of the 9th Int. Conf. on Automated Deduction*, pages 500–516, 1988.

[Sho93] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.