CONSTRUCTING COUNTER–MODELS FOR MODAL LOGIC K4 FROM REFUTATION TREES

Abstract
In the present paper, we will introduce a system which gives an efficient way of constructing counter–models for arbitrary formulae which are unprovable in modal logic K4. Our system is obtained by modifying the method in [7], which gives us an efficient decision procedure for K4. Therefore, as a whole, our system can decide first whether a given formula $A$ is provable in K4 or not, and then gives us a counter–model for $A$ when it is not provable. Since our system can avoid loop-checking as far as possible, the above procedure can be carried out efficiently.

1. Introduction
If a given formula $A$ is not provable in a logical system, can we create a counter–model for $A$ concretely? In some systems there exists an algorithm of creating counter–models, but in other systems even if completeness theorems guarantee the existence of counter–models, we don’t have always such an algorithm. For instance, the finite model property (with the finite axiomatizability) gives us simply an algorithm of finding counter–models by checking finite models thoroughly, which is extremely inefficient.

In many systems of modal logics, it is possible to create counter–models by applying Schütte method to them (see e.g. [9]). But, this method is time–consuming, which in fact, will need exponential time at best. Thus, we want to have more efficient methods of constructing counter–models for unprovable formulas.
A reasonable and practical idea of finding counter–models in an efficient way would be to combine it with proof-search algorithms and to generate counter–models directly from failed proof-search trees. For, such proof-search trees will contain evidences of the reason of the failure, which may be useful in the construction of counter–models. A failed proof–search tree is called a refutation tree since it includes evidence to refuse a given formula.

Thus, our procedure proposed here gives us a proof when a given input formula is provable, or a counter–model for it otherwise. Moreover, we can expect that such a decision procedure will be efficient.

Effective ways of constructing counter–models for classical logic, and modal logics K and KT are known already. A loop–free proof-search procedure for intuitionistic propositional logic is given in [1] and [5]. Based on the procedure, a loop–free construction of counter–models for intuitionistic propositional logic is proposed in 1995 by [8]. A construction of counter–models for modal logic K and KT is shown by the author in [6]. Following the idea of [8], the author introduced also an efficient procedure combining the proof-search with construction of counter–models for modal logic S4 in [7]. A proof–or–refutation algorithm is a kind of decision procedure such that it outputs a proof figure if a given formula is provable, and a counter–model if otherwise.

Extending the idea in [7], we will give in this paper an efficient way of constructing counter–models for modal logic K4. Our method is based on the decision procedure with efficient loop-checking shown in [4]. In the case of intuitionistic logic, we know that there exists always a finite counter–model with a tree-form. Thus, as is done in [8], we can use structures of refutation trees in constructing counter–models. On the other hand, for modal logics, finite counter–models will contain some clusters and semi–clusters in general, and therefore we need to introduce some new idea to overcome this problem. This is done in Section 3 by introducing two kinds of blocks.

2. Decision procedure for K4

Our decision procedure for K4 is obtained by modifying the procedure given in [7]. The basic idea is in fact almost the same. But the construction of counter–models from refutation trees for modal logic K4 becomes
more complicated than S4. Because binary relations in K4 frames are not reflexive in general, for each "block" in a refutation tree we need to take a semi-cluster consisting of a cluster with a non reflexive world, instead of a single cluster.

The resulting system for proof-search procedure is presented below (Figure 1). In this paper, we will consider $A \lor B$ and $A \supset B$ as abbreviations of $\neg (\neg A \land \neg B)$ and $\neg (A \land \neg B)$, respectively.

There are three points in our system which are different from standard sequent systems. The first one is that both sides of our sequents consist of multisets of formulas rather than sequences of formulas, except ($init)_s$, ($init)_t$, ($\Box)_s$ and ($\Box)_t$ rules. In an application of these four rules, we need to check whether conditions of application (written within parentheses) are satisfied or not. In this case, we regard an expression like $\Box \Delta$ in conditions as a set but not as a multiset. For example, both sides of the inclusion $\Box \Delta \subseteq \Box \Sigma$ of the condition for ($\Box)_s$ rule must be interpreted as sets of formulas. Moreover, in ($\Box)_s$ and ($\Box)_t$, both sides of upper sequents should be treated as sets, and hence we need to remove duplications of identical formulas in upper sequents.

$\square \Gamma, p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m, \square \Delta \ (\square \Gamma \square \Sigma) (init)_s \quad (\Box \Delta \subseteq \Box \Sigma, \Box \Delta \neq \emptyset)$

$\square \Gamma, p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m \ (\square \Gamma \square \Sigma) (init)_t$

$\Gamma, A, B \rightarrow \Delta \ (\square \Pi \square \Sigma) \quad \Gamma, A \land B \rightarrow \Delta \ (\square \Pi \square \Sigma) \quad \Gamma \rightarrow A \land B \rightarrow \Delta \ (\square \Pi \square \Sigma) \quad \Gamma \rightarrow \Delta, A \land B \rightarrow \Delta \ (\square \Pi \square \Sigma) \quad (\land R)$

$\Gamma \rightarrow A, \Delta \ (\square \Pi \square \Sigma) \quad \Gamma \rightarrow \Delta, A \ (\Box \Pi \square \Sigma) \quad \Gamma \rightarrow \Delta, \neg A \ (\Box \Pi \square \Sigma) \quad (\neg R)$

$\square \Gamma, \Gamma \rightarrow A_1 \ (\square \Gamma \square \Sigma, \Box \Theta) \ldots \square \Gamma, \Gamma \rightarrow A_n \ (\square \Gamma \square \Sigma, \Box \Theta) \quad (\Box \Theta \equiv \Box A_1, \ldots, \Box A_n, \Box \Theta \cap \Box \Sigma = \emptyset, \Box \Delta \subseteq \Box \Sigma)$

$\square \Gamma, \Gamma \rightarrow A_1 \ (\square \Gamma \square \Theta) \ldots \square \Gamma, \Gamma \rightarrow A_n \ (\square \Gamma \square \Theta) \quad \square \Gamma, \Gamma \rightarrow A \ (\square \Gamma \square \Theta) \quad \square \Gamma, \Gamma \rightarrow A \ (\square \Gamma \square \Theta) \quad \square \Gamma, \Gamma \rightarrow A \ (\square \Gamma \square \Theta) \quad (\Box \Theta \equiv \Box A_1, \ldots, \Box A_n, \Box \Gamma \supseteq \Box \Pi)$

Figure 1: Decision procedure for K4
The second point is that only in (□)_s and (□)_t rules, upper sequents should be understood as ‘or’–branch, but not as ‘and’–branch. Therefore, if one of the upper sequents is provable, then the lower sequent is provable. In order to distinguish it from other rules whose upper sequents are understood as usual, we use double lines (=), instead of single lines.

The third point is that we add certain sets of formulas (called ‘histories’) to sequents, which indicate when rules for modality are applicable. To make it precise, let us consider the case for the (init)_t rule, as an example.

\[\Gamma, p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m (□Π|□Σ) (init)_t\]

Here, the pair \(⟨□Π|□Σ⟩\) is called a history. The first component \(□Π\) of the pair in \((init)_t\) rule is called a valid history, and the second component \(□Σ\) a invalid history. When a counter–model is constructed, \(□Π\) must be valid and \(□Σ\) must be invalid in the world which “corresponds” to this application of \((init)_t\) rule. Histories eliminate any redundant loop–checking in proof-search, and provide us a loop-free procedure in the construction of counter–models.

This system indeed gives us a decision procedure for K4. If a formula is provable in this system, we can construct a proof figure for the formula in our system (as a sequent calculus). Moreover, any proof-search in this system terminates always for any given sequent. To show this, suppose that we have a infinite branch in a proof-search tree. Then, in this branch (□)_t or (□)_s rule must be applied infinitely many times. However, both valid and invalid histories are subsets of all boxed formulas in the original sequent, and thus the total number is finite. Moreover, the number of formulas either in valid or in invalid history increases strictly by each application of (□)_t or (□)_s. The total number of (□)_t and (□)_s in a branch is at most the square of the number of boxed formulas. Therefore, it is impossible to apply them infinitely many times. This is a contradiction.

3. Construction of Counter–models

In this section, we will show how to construct counter–models which are K4-models, when our proof-search is failed for a given sequent.

So, suppose that \(S\) is an unprovable sequent, and \(T\) is its refutation tree. We will construct a Kripke–model from \(T\), in which \(S\) is false.
**Definition 1.** [block] A part of a refutation tree $T$, which begins with any of an initial sequent, $(\square)_a$ and $(\square)_r$ rules and ends either at the end sequent, or at an upper sequent of $(\square)_a$ rule or $(\square)_r$ rule, and which moreover contains no other $(\square)_a$ or $(\square)_r$ rules, is called a block of $T$. In other words, a block is a subtree of a refutation tree between two double lines($=$) (or, between one double line and the end sequent) which contains no other double lines in it.

**Symmetric block** is a block which begins with either $(\square)_a$ or $(init)_a$ rule.

**Transitive block** is a block which begins with $(\square)_r$ or $(init)_r$ rule.

**Initial block** is a block which begins with an initial sequent.

**Symmetric initial block** is a block which begins with $(init)_s$ initial sequent.

**transitive initial block** is a block which begins with $(init)_t$ initial sequent.

For a given refutation tree $T$, we construct a pair of a Kripke–model $M = (W, R, \models)$ and a world $w \in W$ by induction on the number of blocks in $T$ as follows.

**Case 1.** $T$ consists of a single symmetric initial block beginning with $\square \Gamma, p_1, \ldots, p_n \rightarrow q_1, \ldots, q_m, \square \Delta (\square \Gamma | \square \Sigma )$. Then, take $(M, w)$ such that $M = (W, R, \models), W = \{ w \}, R = (w, w), w \models p_1 \wedge \cdots \wedge p_n$.

**Case 2.** $T$ consists of a single transitive initial block: Similar to Case 1.

**Case 3.** $T$ ends at a symmetric block with $(\square)_s$ of the following form

\[
\ldots \square \Gamma, \Gamma \rightarrow S_i \langle \square \Gamma | \square \Sigma, \square \Theta \rangle \ldots \square \Gamma, \Gamma \rightarrow T_j \langle \square \Gamma | \square \Sigma, \square \Theta \rangle \ldots
\]

(1 $\leq i \leq m, 1 \leq j \leq l$):

Among upper sequents of this $(\square)_s$, suppose that each $\square \Gamma, \Gamma \rightarrow S_i (\square \Gamma | \square \Sigma, \square \Theta)$ is an end sequent of a symmetric block, and each $\square \Gamma, \Gamma \rightarrow T_j (\square \Gamma | \square \Sigma, \square \Theta)$ is an end sequent of a transitive block. By the hypothesis of induction, we have $(M_{S_i}, w_{S_i}), (M_{S_{T_j}}, w_{S_{T_j}}, \models)$ for each $i$, and $(M_{S_{T_j}}, w_{S_{T_j}})$, where $M_{T_j} = (W_{T_j}, R_{T_j}, \models)$ for each $j$.

Now, for $T$ we construct $(M, w)$ with $M = (W, R, \models)$ as follows: Let $W = \{ w \} \cup W_{S_i} \cup \cdots \cup W_{S_m} \cup W_{T_i} \cup \cdots \cup W_{T_j}$. Let $R$ be the transitive closure of $R'$, where $R' = \{ (w, w_{S_i}), \ldots, (w, w_{S_m}), (w, w_{T_i}), \ldots, (w, w_{T_j}) \} \cup \{ (w_{S_i}, w), \ldots, (w_{S_m}, w) \} \cup R_{S_i} \cup \cdots \cup R_{S_m} \cup R_{T_i} \cup \cdots \cup R_{T_j}$. For $\models$, let
transitive closure of $R_i$ for each $w_i \in W$ is defined in the same way as one in a model to which $w'$ belongs.

$\rightsquigarrow Sym = Sym = Tra = Tra$

\[ \cdots \cdots \cdots \cdots \] $(\square)_t \implies \circ \cdots \circ \bullet \cdots \bullet$

(Each $\circ$ corresponds to a symmetric block, and each $\bullet$ corresponds to a transitive block.)

**Case 4.** $T$ ends at a transitive block with $(\square)_t$ of the following form

$$\cdots \square \Gamma, \Gamma \rightarrow S_i (\square \Pi \square \Theta) \cdots \square \Gamma, \Gamma \rightarrow T_j (\square \Pi \square \Theta) \cdots$$

$(\square)_t (1 \leq i \leq m, 1 \leq j \leq l)$

Among upper sequents of this $(\square)_t$, suppose that each $\square \Gamma, \Gamma \rightarrow S_i (\square \Pi | \square \Sigma, \square \Theta)$ is an end sequent of a symmetric block, and each $\square \Gamma, \Gamma \rightarrow T_j (\square \Pi | \square \Sigma, \square \Theta)$ is an end sequent of a transitive block. By the hypothesis of induction, as well as the symmetric block case, we have $(M_{S_i, w_{S_i}})$ for each $i$ and $(M_{T_j, w_{T_j}})$ for each $j$.

The transitive block above corresponds to $(M, w)$, where $M = (W, R, \models)$ such that $W = \{w\} \cup W_{S_i} \cup \cdots \cup W_{S_m} \cup W_{T_1} \cup \cdots \cup W_{T_l}$, and $R$ is the transitive closure of $R' = \{(w, w_{S_1}), \cdots, (w, w_{S_m}), (w, w_{T_1}), \cdots, (w, w_{T_l})\} \cup \{(w_{S_1}, w_{S_1}), \cdots, (w_{S_1}, w_{S_m}), (w_{S_1}, w_{T_1}), \cdots, (w_{S_1}, w_{T_l}), \cdots, (w_{S_m}, w_{S_1}), \cdots, (w_{S_m}, w_{S_m}), (w_{S_m}, w_{T_1}), \cdots, (w_{S_m}, w_{T_l})\} \cup \{R_{S_i} \cup \cdots \cup R_{S_m} \cup R_{T_1} \cup \cdots \cup R_{T_l}\}$. For $\models$, let $w \models p_1 \land \cdots \land p_n$, and for any $w' \neq w$ in $W \models$ at $w'$ in this model is defined in the same way as one in a model to which $w'$ belongs.

$\rightsquigarrow Sym = Sym = Tra = Tra$

\[ \cdots \cdots \cdots \cdots \] $(\square)_t \implies \circ \cdots \circ \bullet \cdots \bullet$

**Example 1.** [counter–model] To construct a counter–model from a refutation tree, we need to look at $(\text{init})_s$, $(\text{init})_t$, $(\square)_s$ and $(\square)_t$ rules, and construct a Kripke–frame. For example,
Next, we look at the lower sequents of (init)_s, (init)_t, (□)_s and (□)_t rules, and we determine the valuation at each node. For example,

\[\begin{align*}
\Gamma, p_1 &\rightarrow q_1, □B, □C(□(□)A, □B, □C) & (init)_S \\
□(□) &\rightarrow C(□(□)A, □B, □C) & (□)_S \\
\Gamma, p_2 &\rightarrow q_2, □A, □C(□(□)A, □B) & D & (□)_T \\
□(□) &\rightarrow A(□(□)A, □B) & \Delta &\rightarrow \Sigma(\epsilon|\epsilon) \\
\end{align*}\]

Where \( D \) is:

\[\begin{align*}
\Gamma, p_3 &\rightarrow q_3, □B, □D(□(□)A, □B, □D) & (init)_S \\
\Gamma, p_4 &\rightarrow q_4, □B, □D(□(□)A, □B, □D) & (□)_T \\
\end{align*}\]

Thus, we get the following model.
Theorem 1. Let $T$ be the refutation tree for a given sequent $S$, and $(M, w)$ be the pair of a Kripke–model and a world, determined from $T$ by our method. Then $S$ is false at $w$ in $M$.

Proof. This proof goes similarly to the proof for S4 shown in [7], by using induction on the structure of semi-clusters. Here a semi–cluster means a set of worlds which consists of a cluster $C$ and a possible world $w$, and we let each world in the cluster $C$ to be a successor of the world $w$.

4. Characteristics of K4 counter–models by our method

By our method, each counter–model is represented as a tree of semi–clusters, The higher in the tree structure a semi–cluster is, the more valid boxed formulae it has, as shown in the example in Figure 2.
5. Conclusion

In this paper, we give an effective way of constructing counter-models for modal logic K4 from a proof-search procedure. Thus, we have a proof-or-refutation algorithm for modal logic K4, in total. That is, if a given formula is provable, then we obtain a proof figure for it, and moreover if otherwise, we obtain a counter-model for it.

This algorithm is implemented already on xpe[6]. One can download it from our homepage(http://www.jaist.ac.jp/~mouri). We have only documents in Japanese at this moment, but English version will be available in near future.

After we accomplished the revision of this paper, we found a paper by T. Skura [10] in a recent issue of Studia Logica. He introduced a refutation system for K4 and gave a way of constructing counter-models for K4 from refutation trees. The syntactic system given in [10] is considerably different from ours, and hence there will be no direct relation between his work and ours. Still it will be interesting to see the relationship between these two.

References


Meme Media Laboratory, Faculty of Engineering, Hokkaido University
Kita 13 Nishi 8, Kita-ku, Sapporo 060-8628, Japan
e-mail: mouri@meme.hokudai.ac.jp