A Decision Procedure for Alternation-free Two-way Modal mu-calculus

Yoshinori Tanabe[1,2], Koichi Takahashi[2],
Mitsuharu Yamamoto[3], Akihiko Tozawa[4]
and Masami Hagiya[5]

1: JST CREST, 2: AIST CVS, 3: Chiba University,

4: IBM Japan Ltd, 5: University of Tokyo

A Decision Procedure for Alternation-free Two-way Modal μ-calculus*,**

Yoshinori Tanabe[1,2], Koichi Takahashi[2], Mitsuharu Yamamoto[3], Akihiko Tozawa[4], and Masami Hagiya[5]

[1] CREST, Japan Science and Technology Agency
[2] Research Center for Verification and Semantics, National Institute of Advanced Industrial Science and Technology (AIST)
[3] Faculty of Science, Chiba University
[4] IBM Research, Tokyo Research Laboratory, IBM Japan ltd., Japan
[5] Graduate School of Information Science and Technology, University of Tokyo

Abstract. The satisfiability checking problem is known to be decidable for a variety of modal/temporal logics such as the modal μ-calculus, but effective implementation has not necessarily been developed for all such logics. In this paper, we propose a decision procedure using the tableau method for the alternation-free two-way modal μ-calculus. Although the size of the tableau set maintained in the method might be large for complex formulas, the set and the operations on it can be expressed using BDD and therefore we can implement the method in an effective way.

1 Introduction

There are various applications of satisfiability checking in modal logics. For example it has been used to synthesize a current program from a specification expressed as a temporal formula by checking satisfiability of the formula [1, 2]. The authors have proposed the abstraction method [3] that can be applied to verification problems of graph rewriting systems using satisfiability checking of temporal formulas, and applied the method to analysis of cellular automata [4]. We also applied satisfiability checking of temporal formulas to verification and analysis of programs that process XML documents, whose tree structures are naturally expressible using branching time temporal logic formulas [5].

In applications mentioned above, we regard a graph with labelled edges as a Kripke structure, and describe properties of the graph by formulas in temporal logics. For the purpose of the applications, we often need to follow edges not only in the forward direction but also in the backward direction. Therefore we need to use temporal logics that can handle both directions of edges as modalities.

* This research was supported by Core Research for Evolutional Science and Technology (CREST) Program "New High-performance Information Processing Technology Supporting Information-oriented Society" of Japan Science and Technology Agency (JST).

** This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research on Priority Areas, 16016211, 2004.

From this point of view the two-way modal μ-calculus.[6] is one of the logics we regard as powerful enough [6].

It is known that for the general two-way modal μ-calculus the satisfiability problem is decidable and its complexity is EXPTIME-complete [6]. The decision procedure is constructed by converting the problem into the emptiness problem of the language recognized by a certain alternating tree automaton on infinite trees. In order to solve the emptiness problem, complex operations are required including determinization of parity automata [7]. The algorithm described in [8] has the time complexity $2^{O(n^4 \log n)}$ with regard to the length n of the given formula and actual implementation has not been reported. In possible applications of satisfiability checking mentioned above, on the other hand, the decision procedure need to be called repeatedly. From this point finding a decision procedure that can be implemented effectively is meaningful even if it can only be applied to a subset of the logic as long as the subset is reasonably powerful for expressing problems in such applications.

In this research we focus on the alternation-free two-way modal μ-calculus as such a subset. We propose a decision procedure for checking satisfiability in the logic and describe experimental implementation of the procedure. The time complexity of the decision procedure is $2^{O(n \log n)}$, which is faster than the time complexity for the whole two-way modal μ-calculus as expected. The experimental implementation uses BDD [9]. Effectiveness of the use of BDD in verification tools are well-known. Classical examples are model checking tools such as SMV [10]. It is also applied to satisfiability checking in the experimental procedure for the basic modal logic K [11, 12], and in [13], a verification tool on the monadic second-order logic WS2S, which is closely related to temporal logics and allows an effective decision procedure. Our decision procedure is also suitable to implement with BDD since it is built with iteration of set operations on finite sets such as tableaux. The experimental implementation shows that it finishes in reasonable time for formulas in modest sizes.

As described in detail in a later section, the key point for deriving the time complexity $2^{O(n \log n)}$ is to effectively express the condition concerning loops that include both forward and backward modalities. This point is specific to two-way logics.

For analysis of programs that process XML documents as mentioned above, we need to restrict models to be binary-branching finite trees. We show that our decision procedure can also be used for this purpose, though the performance may not be very attractive. We reported a more effective decision procedure specialized to binary-branching finite tree models [14]. In this case the problem can be converted into WS2S, so we could use Mona for satisfiability checking, but experimental implementation shows that the specialized procedure performs better.

---

[6] The two-way modal μ-calculus is also known as the full modal μ-calculus. In this paper we use the word "two-way" because the word "full" usually means the entire system of a logic while we mainly deal with a subsystem of the μ-calculus.

2 Preliminaries

We denote the set of propositional constants by PC, the set of propositional variables by PV and the set of modalities by Mod. We assume that a function $\overline{\ }$: Mod $\to$ Mod is defined and that $\overline{\overline{a}} = a$ for each $a \in$ Mod.

Definition 1. *We define the set L_μ of the formulas of the alternation-free two-way modal μ-calculus.*

- $\top, \bot \in L_\mu$.
- *For propositional constant$P \in PC$, $P, \neg P \in L_\mu$. For propositional variable $X \in PV$, $X \in L_\mu$.*
- *If $\varphi_1, \varphi_2 \in L_\mu$ then $\varphi_1 \vee \varphi_2, \varphi_1 \wedge \varphi_2 \in L_\mu$. We call $\vee$ and $\wedge$ the principal operator of $\varphi_1 \vee \varphi_2$ and $\varphi_1 \wedge \varphi_2$ respectively.*
- *If $\varphi \in L_\mu$ and $a \in Mod$ then $[a]\varphi, \langle a\rangle\varphi \in L_\mu$. We call $[]$ and $\langle\rangle$ the principal operator of $[a]\varphi$ and $\langle a\rangle\varphi$ respectively.*
- *If $\varphi \in L_\mu$ and $X \in PV$ then $\mu X\varphi, \nu X\varphi \in L_\mu$. We call μ and ν the principal operator of $\mu X\varphi$ and $\nu X\varphi$ respectively.*

An occurrence of $X \in PC$ in $\varphi \in L_\mu$ is bound if it is contained in a subformula of the form of $\mu X\varphi'$ or $\nu X\varphi'$, otherwise it is free. We call $\varphi \in L_\mu$ a formula of the alternation-free two-way modal μ-calculus if the following conditions are satisfied:

- *If $\mu X\psi$ is a subformula of φ and $\nu Y\chi$ is a subformula of ψ then X does not occur freely in χ.*
- *If $\nu X\psi$ is a subformula of φ and $\mu Y\chi$ is a subformula of ψ then X does not occur freely in χ.*

We denote the set of formulas of the alternation-free two-way modal μ-calculus by L_μ^{af}.

For example, $\mu X\nu Y(P \wedge [a]Y \wedge [\overline{a}]X)$ is a formula of the two-way modal μ-calculus but it is not alternation-free since X appears in the scope of νY. On the other hand, $\mu X(\nu Y(P \wedge [b]Y) \vee \langle a\rangle X \vee \langle\overline{a}\rangle X)$ is alternation-free. This is equivalent to formula $\mathbf{E}_{a,\overline{a}}\mathbf{F}\,\mathbf{A}_b\mathbf{G}\,P$ of the two-way CTL. A formula in two-way CTL can be translated into a formula of the alternation-free two-way modal μ-calculus.

Let $F \subseteq \mathrm{L}_\mu^{\mathrm{af}}$ and S be a sequence of $\vee$, $\wedge$, $[]$, $\langle\rangle$, μ and ν. We denote by $\mathrm{PO}(F; S)$ the set of the elements of F whose principal operator is listed in S. When $\varphi = \langle a\rangle\varphi'$ or $\varphi = [a]\varphi'$, we denote the modality a by $\mathrm{mod}(\varphi)$, and the subformula φ' by $\overrightarrow{\varphi}$.

Definition 2. *A Kripke structure is a triple $\mathcal{M} = (M, R, \lambda)$ that satisfies the following conditions:*

- *M is a set.*
- *R is a function whose domain is Mod and for $a \in Mod$, $R(a) \subseteq M \times M$ and $R(\overline{a}) = R(a)^{-1}$.*
- *λ is a function whose domain is PC and for $P \in PC$, $\lambda(P) \subseteq M$.*

A valuation v is a function whose domain is PV and for $X \in PV$, $v(X) \subseteq M$. For $S \subseteq M$ we denote by $v[X \mapsto S]$ the valuation obtained from v by replacing the value of $v(X)$ with S.

Definition 3. *For $\varphi \in L_\mu^{af}$, Kripke structure $\mathcal{M} = (M, R, \lambda)$ and valuation v we define relation $\mathcal{M}, v, m \models \varphi$, or $m \models \varphi$ for short.*

- $m \models \top$ *holds for all* $m \in M$.
- $m \models \bot$ *holds for no* $m \in M$.
- *For* $P \in PC$, $m \models P \iff m \in \lambda(P)$.
- *For* $P \in PC$, $m \models \neg P \iff m \notin \lambda(P)$.
- *For* $X \in PV$, $m \models X \iff m \in v(X)$.
- $m \models \varphi_1 \vee \varphi_2 \iff m \models \varphi_1$ *or* $m \models \varphi_2$.
- $m \models \varphi_1 \wedge \varphi_2 \iff m \models \varphi_1$ *and* $m \models \varphi_2$.
- $m \models \langle a \rangle \varphi \iff$ *There exists* $m' \in M$ *such that* $(m, m') \in R(a)$ *and* $m' \models \varphi$.
- $m \models [a]\varphi \iff$ *For all* $m' \in M$ *if* $(m, m') \in R(a)$ *then* $m' \models \varphi$.
- $m \models \mu X \varphi \iff m \in \bigcap \{S \subseteq M \mid S \supseteq \{m \in M \mid \mathcal{M}, v[X \mapsto S], m \models \varphi\}\}$.
- $m \models \nu X \varphi \iff m \in \bigcup \{S \subseteq M \mid S \subseteq \{m \in M \mid \mathcal{M}, v[X \mapsto S], m \models \varphi\}\}$.

If φ is a sentence, that is all propositional variables occurring in φ is bound, the relation does not depend on the valuation so we write $\mathcal{M}, m \models \varphi$ instead of $\mathcal{M}, v, m \models \varphi$.

As an example, let $\mathrm{PC} = \{P\}$, $M = \omega$ (the set of natural numbers), $\mathrm{Mod} = \{a, \overline{a}\}$, $R(a) = \{(n, n+1) \mid n < \omega\}$ and $\lambda(P) = \{0\}$, and we consider Kripke structure $\mathcal{M} = (M, R, \lambda)$. Let $\varphi_1 = \nu X(\neg P \wedge [a]X)$ and $\varphi_2 = \mu X(P \vee [\overline{a}][\overline{a}]X)$. Then $m \models \varphi_1$ is equivalent to $m > 0$ and $m \models \varphi_2$ holds if and only if m is even.

We will introduce a few more concept to describe a decision procedure.

First, for $\varphi_1, \varphi_2 \in \mathrm{L}_\mu^{\mathrm{af}}$, if $\mathcal{M}, m \models \varphi_1 \iff \mathcal{M}, m \models \varphi_2$ holds for any Kripke structure $\mathcal{M} = (M, R, \lambda)$ and $m \in M$, we say φ_1 and φ_2 are *equivalent* and write $\varphi_1 \equiv \varphi_2$.

Next we define normal form of a formula.

Definition 4. *Formula $\varphi \in L_\mu^{af}$ that satisfies the following condition is said to be in in normal form:*

- *For any propositional variable X, either all the occurrence of X is free or there is unique subformula ψ of φ in the form of $\mu X \varphi'$ or $\nu X \varphi'$ and X does not occur outside of ψ.*
- *If $\mu X \varphi'$ or $\nu X \varphi'$ is a subformula of φ, for any occurrence of X in φ' there is a subformula of φ' that contains the occurrence and is in the form of $\langle a \rangle \varphi''$ or $[a]\varphi''$.*

For example $\varphi_1 = \mu X.[a](X \vee \mu X.\langle b \rangle X)$ is not in normal form since it does not satisfy the first condition, neither is $\varphi_2 = \mu X.X$ since it does not satisfy the second condition. On the other hand $\varphi_3 = \mu X.[a](X \vee \mu Y.\langle b \rangle Y)$ is in normal form.

It is known [15, Lemma 2.2] that for every $\varphi \in \mathrm{L}_\mu^{\mathrm{af}}$ there exists a formula $\psi \in \mathrm{L}_\mu^{\mathrm{af}}$ in normal form that is equivalent to φ. So in the rest of the paper we assume that all formulas in consideration are in normal form.

We will then define the closure of a formula and related concepts. For $\varphi = \mu X\psi$ or $\varphi = \nu X\psi$, we denote by $\exp(\varphi)$ the formula ψ with replacing all the occurrence of X with φ. We define relation $\rightarrow_{\mathrm{e}}$ on $\mathrm{L}_{\mu}^{\mathrm{af}}$ to be the least relation that satisfies the following:

- $\varphi_1 \vee \varphi_2 \rightarrow_{\mathrm{e}} \varphi_i$ $(i = 1, 2)$.
- $\varphi_1 \wedge \varphi_2 \rightarrow_{\mathrm{e}} \varphi_i$ $(i = 1, 2)$.
- $\langle a\rangle\varphi \rightarrow_{\mathrm{e}} \varphi$.
- $[a]\varphi \rightarrow_{\mathrm{e}} \varphi$.
- $\mu X\varphi \rightarrow_{\mathrm{e}} \exp(\mu X\varphi)$.
- $\nu X\varphi \rightarrow_{\mathrm{e}} \exp(\nu X\varphi)$.

The *closure* of formula $\varphi \in \mathrm{L}_{\mu}^{\mathrm{af}}$ is defined as the smallest set S that contains φ and closed under relation $\rightarrow_{\mathrm{e}}$, that is, if $\psi \in S$ and $\psi \rightarrow_{\mathrm{e}} \psi'$ then $\psi' \in S$. We denote the closure of φ by $\mathrm{cl}(\varphi)$. Let $\mathcal{D}$ be the set of strongly connected components of $\mathrm{cl}(\varphi)$ with respect to relation $\rightarrow_{\mathrm{e}}$, and we define $\mathcal{D}_{\mu} = \{D \in \mathcal{D} \mid \mathrm{PO}(D; \mu) \neq \varnothing\}$ and $\mathcal{D}_{\nu} = \{D \in \mathcal{D} \mid \mathrm{PO}(D; \nu) \neq \varnothing\}$. Some basic properties on the concepts defined so far are the following:

Proposition 1. *(1)* $\varphi \equiv \exp(\varphi)$.
(2) *$\mathrm{cl}(\varphi)$ is a finite set and its size is linear to the length of φ, that is the number of operators and propositional variables appearing in φ.*
(3) $\mathcal{D}_{\mu} \cap \mathcal{D}_{\nu} = \varnothing$.

Proof. Refer [16] for claims (1) and (2). Claim (3) follows from the fact that φ is alternation-free. □

For example consider formula $\varphi = \mu X([a]X \wedge \nu Y\langle b\rangle Y)$. If we denote the subformula $\nu Y\langle b\rangle Y$ by ψ then φ can be written as $\mu X([a]X \wedge \psi)$. With these, $\exp(\varphi)$ is $[a]\varphi \wedge \psi$, and the closure $\mathrm{cl}(\varphi)$ is $\{\varphi, [a]\varphi \wedge \psi, [a]\varphi, \psi, \langle b\rangle\psi\}$. Relation $\rightarrow_{\mathrm{e}}$ contains $\varphi \rightarrow_e [a]\varphi \wedge \psi \rightarrow_e [a]\varphi \rightarrow_e \varphi$. There are two strongly connected components, namely $D_1 = \{\varphi, [a]\varphi \wedge \psi, [a]\varphi\}$ and $D_2 = \{\psi, \langle b\rangle\psi\}$. D_1 belongs to $\mathcal{D}_{\mu}$ while D_2 belongs to $\mathcal{D}_{\nu}$.

Let Lit be the set of propositional constants and their negations that are members of $\mathrm{cl}(\varphi)$. We call the set $\mathrm{Lit} \cup \mathrm{PO}(\mathrm{cl}(\varphi); \langle\rangle, [])$ the *lean* of φ after Pan et al [11]. Since we only consider formulas in normal form, every "path" in $(\mathrm{cl}(\varphi), \rightarrow_{\mathrm{e}})$ reaches to the lean after finite steps and during the path we see only $\vee$, $\wedge$, μ and ν as principal operators.

3 The Decision Procedure

3.1 Overview

A simple decision procedure for satisfiability checking in CTL, which is a subsystem of the two-way modal μ-calculus, is well-known [17]. In the case of CTL, by expanding modal formulas (for example, formula $\mathbf{EF}\,\varphi$ is expanded to $\varphi \vee \mathbf{EXEF}\,\varphi$), any formula can be regarded as a boolean combination of basic propositions and formulas in the form of $\mathbf{EX}\varphi$ or $\mathbf{AX}\varphi$. We consider a tableau

which consists of nodes each of which is a function mapping basic propositions and formulas $\mathbf{EX}\varphi$ and $\mathbf{AX}\varphi$ to truth values. Starting with the full tableau consisting of all nodes, we repeatedly dispose "inconsistent" nodes. Among the types of inconsistency, the most important one is on "eventuality." An example situation is that the formula $\mathbf{EF}\,\varphi$ is "true" at a node but it is not reachable to a node at which φ is "true." In the case of CTL, this type of inconsistency can be judged relatively easily, because while processing the formula $\mathbf{EF}\,\varphi$ we do not need to process other formulas.

In the case of the modal μ-calculus, however, the situation is not so simple. While formulas are boolean combinations of basic propositions, $\langle a\rangle\varphi$ and $[a]\varphi$ (that is, the elements of the lean of the top level formula) as in the case of CTL, during processing expansion of an μ operator we may encounter ν operators. Then it is not possible to judge if there is inconsistency for the μ operator by simply checking a finite set of nodes since the encountered ν operators also need to be taken into account.

We can avoid this difficulty if we focus on the alternation-free fragment of the two-way modal μ-calculus. Although two or more formulas may be involved in the inconsistency checking unlike the case of CTL, it is guaranteed that ν-operators do not appear during expansion of μ-operators.

But there are still issues that arise from the fact we have two directions of modalities. Consider a formula $\varphi = \mu X([a]X \wedge [\overline{a}]X)$. Suppose that there are two nodes n_1 and n_2 in the tableau, and $[a]\varphi$ and $[\overline{a}]\varphi$ are true at both n_1 and n_2. Can we put an edge labelled by a from node n_1 to node n_2? At a first glance it seems all right since φ is calculated to be true at both n_1 and n_2. Unfortunately, however, this does not work. In order for n_1 to satisfy $[a]\varphi$, n_2 must satisfy φ, which imposes a requirement for n_2 to satisfy $[\overline{a}]\varphi$, which depends on n_1 to satisfy φ that requires $[a]\varphi$ at n_1, returning to the start point. We need a mechanism to avoid such an edge.

To achieve the mechanism we introduce at each node n a binary relation $E(\varphi, \psi)$ for two formulas φ and ψ that are true at n. The intention of $E(\varphi, \psi)$ is that "it is not prohibited that an expansion sequence stating with φ reaches ψ." Nodes in the tableau must express not only true/false of the formulas in the lean but also the relation E, that is, different Es yield different nodes. From the intention E must be a transitive and irreflexive relation and the following property is requested: if there is an edge labelled by a from node n_1 to n_2 and if $E(\varphi, [\overline{a}]\psi)$ holds at node n_2 then $E([a]\varphi, \psi)$ holds at node n_1. There is also an requirement regarding to $\langle\rangle$-formulas, which we do not discuss here. See the relation LoopSafe in Section 3.2 for the exact definition.

This way leads to a correct decision procedure but we employ one more trick for better complexity. The size of the lean is proportional to the length n of the formula for which the procedure decides satisfiability. A naive implementation would encode the relation E as a subset of the direct product of two copies of the lean. Then the number of nodes in the tableau would be $2^{\mathcal{O}(n^2)}$, which means the overall complexity is $2^{\mathcal{O}(n^2)}$ and the number of BDD variables required in an implementation using BDDs is $\mathcal{O}(n^2)$. To reduce the number we encode a node

n by a function f whose domain is the lean and whose range is some appropriate $m = \{0, \ldots, m-1\}$. The encoding is such that if φ is true at n then $f(\varphi) > 0$ else $f(\varphi) = 0$, and if $E(\varphi_1, \varphi_2)$ holds, then $f(\varphi_2) \leq f(\varphi_1)$ holds. With this encoding the number of the nodes is reduced to $2^{\mathcal{O}(n \log m)}$. The number m depends on $D \in \mathcal{D}_\mu$. If formulas in the form of $\langle a \rangle \psi$ or $[a]\psi$ and formulas in the form of $\langle \overline{a} \rangle \psi$ or $[\overline{a}]\psi$ do not appear in common in D, we take $m = 2$. If all $D \in \mathcal{D}_\mu$ satisfies the condition the number of the nodes is $2^{\mathcal{O}(n)}$. Otherwise m is the number of such formulas appearing in D plus one. The worst case complexity is $2^{\mathcal{O}(n \log n)}$.

The alternation-free two-way modal μ-calculus contains a subsystem (for example CTL) whose time complexity is known to be EXPTIME-complete [17]. Therefore the complexity of the procedure cannot be less than $2^{\mathcal{O}(n)}$.

3.2 The Decision Procedure

In this subsection we give a decision procedure to judge whether there is a Kripke structure $\mathcal{M} = (M, R, \lambda)$ and $m \in M$ such that $\mathcal{M}, m \models \varphi_{\mathrm{I}}$ for a given sentence $\varphi_{\mathrm{I}} \in \mathrm{L}_\mu^{\mathrm{af}}$ using tableau method.

We assume $\mathcal{D}_\mu \neq \varnothing$ without loss of generality, since if it is not the case we can consider $\varphi_{\mathrm{I}} \vee \mu X \langle a \rangle X$ instead of φ_{I}. Note that $\mu X \langle a \rangle X$ is equivalent to $\bot$.

We introduce some notation. Suppose $D \in \mathcal{D}_\mu$.

- $\mathrm{PC}_{\mathrm{I}} = \mathrm{PC} \cap \mathrm{cl}(\varphi_{\mathrm{I}})$.
- $\mathrm{Lean} = \mathrm{PC}_{\mathrm{I}} \cup \mathrm{PO}(\mathrm{cl}(\varphi_{\mathrm{I}}); \langle\rangle, [])$
- $\mathrm{BMod}_D = \{a \in \mathrm{Mod} \mid$ There exists $\varphi, \psi \in \mathrm{PO}(D; \langle\rangle, [])$ such that $\mathrm{mod}(\varphi) = a$ and $\mathrm{mod}(\psi) = \overline{a}.\}$.
- $\mathrm{BForm}_D = \{\varphi \in \mathrm{PO}(D; \langle\rangle, []) \mid \mathrm{mod}(\varphi) \in \mathrm{BMod}_D\}$.

Definition 5. *An φ_I-type is a function $t : Lean \to \omega$ satisfying the following conditions: for $\varphi \in Lean$,*

- *If $\varphi \in D \cap Lean$ for $D \in \mathcal{D}_\mu$, $t(\varphi) \leq |BForm_D| + 1$*
- *otherwise, $t(\varphi) \leq 1$*

where $|A|$ is the number of a finite set A. We denote the set of all φ_I-type by T_I.

In our decision procedure, φ_{I}-types are the nodes of the tableau. We define a total order $<_{\mathrm{L}}$ on ω by $1 <_{\mathrm{L}} 2 <_{\mathrm{L}} \ldots <_{\mathrm{L}} 0$. For $t \in T$, a function $\overline{t} : \mathrm{cl}(\varphi_{\mathrm{I}}) \to \omega$ is defined as follows:

- For $\varphi \in \mathrm{Lean}$, $\overline{t}(\varphi) = t(\varphi)$.
- For $P \in \mathrm{PC}_{\mathrm{I}}$, $\overline{t}(\neg P) = 1 - t(P)$.
- $\overline{t}(\varphi_1 \vee \varphi_2) = \min_{\mathrm{L}}\{\overline{t}(\varphi_1), \overline{t}(\varphi_2)\}$.
- $\overline{t}(\varphi_1 \wedge \varphi_2) = \max_{\mathrm{L}}\{\overline{t}(\varphi_1), \overline{t}(\varphi_2)\}$.
- $\overline{t}(\mu X \varphi) = \overline{t}(\exp(\mu X \varphi))$.
- $\overline{t}(\nu X \varphi) = \overline{t}(\exp(\nu X \varphi))$.

where $\min_{\mathrm{L}}$ and $\max_{\mathrm{L}}$ are the minimum and maximum value with respect to the order $<_{\mathrm{L}}$ respectively. For $t \in T$ and $\varphi \in \mathrm{cl}(\varphi_{\mathrm{I}})$ we define relation $t \Vdash \varphi$ as $\overline{t}(\varphi) \neq 0$. For $t, t' \in T$ and $\varphi_1, \varphi_2 \in \mathrm{PO}(\mathrm{cl}(\varphi_{\mathrm{I}}), \langle\rangle, [])$, relation $\mathrm{LoopSafe}(t, t', \varphi_1, \varphi_2)$ holds if the following two conditions are satisfied:

- $\overline{t'}(\varphi_2) \leq_{\mathrm{L}} \overline{t'}(\overrightarrow{\varphi_1}) <_{\mathrm{L}} 0 \implies \overline{t}(\overrightarrow{\varphi}_2) <_{\mathrm{L}} \overline{t}(\varphi_1)$

– $\overline{t}(\varphi_1) \leq_{\mathrm{L}} \overline{t}(\overrightarrow{\varphi}_2) <_{\mathrm{L}} 0 \implies \overline{t'}(\overrightarrow{\varphi}_1) <_{\mathrm{L}} \overline{t'}(\varphi_2)$

For $t, t' \in T$ and $\langle a\rangle\varphi \in \mathrm{cl}(\varphi_{\mathrm{I}})$, relation $t \xrightarrow{\langle a\rangle\varphi} t'$ holds if the following conditions are satisfied:

– $t' \Vdash \varphi$.
– For $[a]\psi \in \mathrm{cl}(\varphi_{\mathrm{I}})$ if $t \Vdash [a]\psi$ then $t' \Vdash \psi$.
– For $[\overline{a}]\psi \in \mathrm{cl}(\varphi_{\mathrm{I}})$ if $t' \Vdash [\overline{a}]\psi$ then $t \Vdash \psi$.
– For all $D \in \mathcal{D}_\mu$,
  • For all $[a]\psi_1, [\overline{a}]\psi_2 \in \mathrm{BForm}_D$ it holds $\mathrm{LoopSafe}(t, t', [a]\psi_1, [\overline{a}]\psi_2)$.
  • If $\langle a\rangle\varphi \in \mathrm{BForm}_D$ holds then for all $[\overline{a}]\psi_2 \in \mathrm{BForm}_D$, it holds $\mathrm{LoopSafe}(t, t', \langle a\rangle\varphi, [\overline{a}]\psi_2)$.

We introduce some notation:

– For $t \in T_{\mathrm{I}}$, $\mathrm{sat}_{\mathrm{T}}(t) = \{\varphi \in \mathrm{Lean} \mid t \models \varphi\}$
– For $T \subseteq T_{\mathrm{I}}$ and $D \in \mathcal{D}_\mu$,
  $\mathrm{TS}(T, D) = \{(t, S) \mid t \in T,\ S \subseteq D \cap \mathrm{sat}_{\mathrm{T}}(t)$, for any $\varphi \in S$ and $\psi \in D \cap \mathrm{Lean}$ if $t(\psi) < t(\varphi)$ then $\psi \in S.\}$
– For $(t, S) \in \mathrm{TS}(T, D)$, $t_S \in T$ is defined as follows: for $\varphi \in \mathrm{Lean}$,
$$t_S(\varphi) = \begin{cases} 0 & \text{if } \varphi \in D \setminus S \\ t(\varphi) & \text{otherwise} \end{cases}$$
– For $T \subseteq T_{\mathrm{I}}$ and $V \subseteq \mathrm{TS}(T, D)$ we define $\mathrm{Step}_D(V, T) \subseteq \mathrm{TS}(T, D)$:
  $(t, S) \in \mathrm{Step}_D(V, T)$
  $\overset{\text{def}}{\iff}$ For any $\langle a\rangle\psi \in \mathrm{sat}_{\mathrm{T}}(t)$ there exists $(t', S') \in V$ such that:
  • $t \xrightarrow{\langle a\rangle\psi} t'$ holds.
  • For any $[a]\varphi \in S$ (the modality a should be in common), ${t'}_{S'} \Vdash \varphi$.
  • If $\langle a\rangle\psi \in S$, ${t'}_{S'} \Vdash \psi$.

Suppose $T \subseteq T_{\mathrm{I}}$ and $D \in \mathcal{D}_\mu$. For $j < \omega$ we define $V_j(D, T) \subseteq \mathrm{TS}(T, D)$, or V_j for short.

– $V_0 = \{(t, \varnothing) \mid t \in T\}$
– $V_{j+1} = V_j \cup \mathrm{Step}_D(V_j, T)$

Each V_j is a subset of a finite set T_{I} and $(V_j \mid j < \omega)$ is an increasing sequence with the inclusion relation. Therefore there exists $J = J(D, T)$ such that if $j \geq J$ then $V_j = V_J$. Then we define $T_k \subseteq T_{\mathrm{I}}$ for $k < \omega$.

– $T_0 = T_{\mathrm{I}}$
– $T_{k+1} = \mathrm{Con}_\Diamond(T_k) \cap \mathrm{Con}_\mu(T_k)$

where

– $\mathrm{Con}_\Diamond(T) = \{t \in T \mid$ For all $\langle a\rangle\varphi \in \mathrm{sat}_{\mathrm{T}}(t)$ there exists $t' \in T$ such that $t \xrightarrow{\langle a\rangle\varphi} t'.\ \}$
– $\mathrm{Con}_\mu(T) = \{t \in T \mid$ For all $D \in \mathcal{D}_\mu$, $(t, D \cap \mathrm{sat}_{\mathrm{T}}(t)) \in V_J\}$

Since $(T_k \mid k < \omega)$ is a decreasing sequence with the inclusion relation, there exists $K < \omega$ such that if $k \geq K$ then $T_k = T_K$.

Theorem 1. *For a sentence $\varphi \in L^{af}_\mu$ the following two conditions are equivalent.*
(1) φ_I is satisfiable, that is, there is a Kripke structure $\mathcal{M} = (M, R, \lambda)$ and $m \in M$ such that $\mathcal{M}, m \models \varphi_I$ holds.
(2) There is $t \in T_K$ such that $t \Vdash \varphi_I$.

Proof. Refer to Sections 4.2 and 4.3. □

Theorem 1 gives a decision procedure for checking satisfiability of the alternation-free two-way modal μ-calculus sentence. We start from $T_0 = T_{\mathrm{I}}$ and calculate sequentially $T_1, T_2, \dots$ until we get the same T_K compared with the previous stage, then we check if $t \Vdash \varphi_{\mathrm{I}}$ holds for each t in a finite set T_K. Note that every step can be done by checking elements in finite sets. Complexity can be calculated by counting the size of such finite sets. For example the number of the iteration of the main loop $(T_0, T_1, \dots)$ is evaluated by the size of T_{I}, which is less than n^n. Calculating the other loops as well shows that the time complexity of this decision procedure is $2^{\mathcal{O}(n \log n)}$.

3.3 Finite Binary Tree Model

When we apply the satisfiability checking procedure to XML problems, we usually pose the following restrictions:

- The set of modality is fixed: $\mathrm{Mod} = \{1, 2, \overline{1}, \overline{2}\}$.
- A model of a formula must form a finite binary tree. More precisely, (M, R, λ) is a finite binary tree model (or FBTM for short) if M is finite and $(M, R(1) \cup R(2))$ is a tree and for all $m \in M$ and $j \in \{1, 2\}$, there is at most one $m_j \in M$ such that $(m, m_j) \in R(j)$.

Definition 6.

(1) An FBTM $\mathcal{M} = (M, R, \lambda)$ satisfies φ if $\mathcal{M}, r \models \varphi$ where r is the root of the tree M.

(2) We denote by rootAx the formula $\mu X([1]X \wedge [2]X) \wedge [\overline{1}]\bot \wedge [\overline{2}]\bot$.

(3) For a formula φ, we denote by φ^{FBT} the formula φ with replacing all occurrences of $\langle a \rangle \psi$ by $\langle a \rangle \top \wedge [a](\psi^{FBT})$, where a is either 1, 2, $\overline{1}$ or $\overline{2}$.

We can use the decision procedure in Section 3.2 for deciding whether a formula is satisfied by an FBTM or not:

Proposition 2. *A formula φ is satisfied by some FBTM if and only if rootAx$\wedge$ φ^{FBT} is satisfiable.*

Proof. The "only if" part is almost immediate. To prove the "if" part, we assume $(M, R, \lambda), m_0 \models \mathrm{rootAx} \wedge \varphi^{\mathrm{FBT}}$. We construct a tree model (T, S, λ') and a function $f : T \to M$. Add the root t_0 of T with $f(t_0) = m_0$. Suppose we constructed a node t of T. For $j \in \{1, 2\}$, if there is $m' \in M$ with $(f(t), m') \in R(j)$ then pick one such m' and add a node t' to T with $f(t') = m'$. Finally we define $\lambda'(P) = \{t \in T \mid f(t) \in \lambda(P)\}$.

There is no infinite branch in T since if there were such B then $\{f(t) \mid t \in B\}$ would be a counterexample for $\mu X([1]X \wedge [2]X)$ at m_0. By König's lemma, T is finite. To show $(T, S, \lambda') \models \varphi'$, it is enough to show that for all formula ψ and $t \in T$, $(T, S, \lambda'), t \models \psi$ holds if $(M, R, \lambda), f(t) \models \psi^{\mathrm{FBT}}$ holds, which can easily been shown by induction on the construction of ψ. □

4 Correctness

4.1 Properties on Kripke Structures

In this subsection we state two basic properties on Kripke structures and formulas in the alternation-free two-way modal μ-calculus without proof. They will be used to prove correctness of the decision procedure in the following subsections. Let $\mathcal{M} = (M, R, \lambda)$ a Kripke structure. We denote the set $\{(\varphi, m) \in \mathrm{cl}(\varphi_{\mathrm{I}}) \times M \mid m \models \varphi\}$ by $\mathrm{Sat}(\mathcal{M})$.

Let $Q \subseteq \mathrm{cl}(\varphi_{\mathrm{I}}) \times M$. We call a function F a *choice function* on Q if its domain is the set $\{(\varphi, m) \in Q \mid \varphi \in \mathrm{PO}(\mathrm{cl}(\varphi_{\mathrm{I}}); \vee, \langle\rangle)$ and satisfies the following:

- $F(\varphi_1 \vee \varphi_2, m) \in \{\varphi_1, \varphi_2\}$
- $F(\langle a\rangle\varphi, m) \in M$, $(m, F(\langle a\rangle\varphi, m)) \in R(a)$,

We define relation R_F on $\mathrm{cl}(\varphi_{\mathrm{I}}) \times M$ to be the least one that satisfies the following:

- If $\varphi = \varphi_1 \vee \varphi_2$ and $(\varphi, m) \in Q$ then $(\varphi, m)\, R_F\, (F(\varphi, m), m)$.
- $(\varphi_1 \wedge \varphi_2, m)\, R_F\, (\varphi_i, m) \quad (i = 1, 2)$.
- $(\mu X\varphi, m)\, R_F\, (\exp(\mu X\varphi), m)$.
- $(\nu X\varphi, m)\, R_F\, (\exp(\nu X\varphi), m)$.
- If $(\langle a\rangle\varphi, m) \in Q$ then $(\langle a\rangle\varphi, m)\, R_F\, (\varphi, F(\langle a\rangle\varphi, m))$.
- If $(m, n) \in R(a)$ then $([a]\varphi, m)\, R_F\, (\varphi, n)$.

Proposition 3. *If $Q \subseteq cl(\varphi_I) \times M$ and a choice function F on Q satisfies the following three conditions, $Q \subseteq Sat(\mathcal{M})$ holds.*

(1) For $P \in PC$, if $(P, m) \in Q$ then $m \in \lambda(P)$. If $(\neg P, m) \in Q$ then $m \notin \lambda(P)$.

(2) Q is closed under R_F, that is, $(\varphi, m) \in Q$ and $(\varphi, m)\, R_F\, (\psi, n)$ implies $(\psi, n) \in Q$.

(3) If $D \in \mathcal{D}_\mu$ and $Q_D = (D \times M) \cap Q$ then ${R_F}^{-1} \restriction (Q_D \times Q_D)$ is a well-founded relation on Q_D, that is, there is no sequence $(q_i \mid i < \omega)$ such that $q_i \in Q_D$ and $q_i R_F q_{i+1}$ holds for all $i < \omega$.

We fix $D \in \mathcal{D}_\mu$ until the end of this subsection. Let us denote the class of ordinals by On. For $\alpha \in \mathrm{On}$ we define $U_\alpha \subseteq \mathrm{Sat}(\mathcal{M})$ as the least set that satisfies the following:

- $\{(\varphi, m) \in \mathrm{Sat}(\mathcal{M}) \mid \varphi \notin D\} \subseteq U_0$
- If $(\varphi_1, m) \in U_\alpha$ or $(\varphi_2, m) \in U_\alpha$ then $(\varphi_1 \vee \varphi_2, m) \in U_\alpha$.
- If $(\varphi_1, m) \in U_\alpha$ and $(\varphi_2, m) \in U_\alpha$ then $(\varphi_1 \wedge \varphi_2, m) \in U_\alpha$.
- If $(\exp(\mu X\varphi), m) \in U_\alpha$ then $(\mu X\varphi, m) \in U_\alpha$.
- If there is $m' \in M$ and $\beta < \alpha$ such that $(\varphi, m') \in U_\beta$ and $(m, m') \in R(a)$ then $(\langle a\rangle\varphi, m) \in U_\alpha$.
- If for all $m' \in M$ such that $(m, m') \in R(a)$ there exists $\beta < \alpha$ with $(\varphi, m') \in U_\beta$ then $([a]\varphi, m) \in U_\alpha$.

Proposition 4. $Sat(\mathcal{M}) = \bigcup_{\alpha \in On} U_\alpha$

For $(\varphi, m) \in \mathrm{Sat}(\mathcal{M})$ we denote by $\mathrm{rank}(\varphi, m)$ by the least ordinal α such that $(\varphi, m) \in U_\alpha$.

4.2 Completeness

In this subsection we prove that the decision procedure is complete, that is, (1) $\Longrightarrow$ (2) of Theorem 1. Take a Kripke structure $\mathcal{M} = (M, R, \lambda)$ with $m_{\mathrm{I}} \in M$ such that $m_{\mathrm{I}} \models \varphi_{\mathrm{I}}$. For $m \in M$ we define $g(m) \in T_{\mathrm{I}}$ as follows. For $\varphi \in$ Lean:

$$g(m)(\varphi) = \begin{cases} 0 & \text{if } m \not\models \varphi \\ 1 & \text{if } m \models \varphi \text{ and } \varphi \notin \bigcup \mathcal{D}_\mu \\ 1 + k_\varphi & \text{if } m \models \varphi \text{ and } \varphi \in \bigcup \mathcal{D}_\mu \end{cases}$$

where $k_\varphi = \left|\{\mathrm{rank}(\psi, m) \mid \psi \in \mathrm{BForm}_D,\ \mathrm{rank}(\psi, m) \leq \mathrm{rank}(\varphi, m)\}\right|$ with $\varphi \in D \in \mathcal{D}_\mu$. It is clear from the definition that $g(m) \in T_{\mathrm{I}}$ and that $m \models \varphi \iff g(m) \Vdash \varphi$. We have three propositions. Proposition 5 can be proved using the definition of g and Proposition 6 can be shown using Proposition 5 twice. We leave the details to the reader.

Proposition 5. *Suppose $D \in \mathcal{D}_\mu$, $\varphi \in \mathit{BForm}_D$, $\psi \in D$, $m \models \varphi$ and $m \models \psi$. Then $\mathit{rank}(\psi, m) < \mathit{rank}(\varphi, m)$ implies $\overline{g(m)}(\psi) < g(m)(\varphi)$.*

Proposition 6. *Suppose $m, m' \in M$, $(m, m') \in R(a)$, $m \models \langle a \rangle \varphi$, $m' \models \varphi$ and $\mathit{rank}(\langle a \rangle \varphi, m) > \mathit{rank}(\varphi, m')$. Then $g(m) \xrightarrow{\langle a \rangle \varphi} g(m')$ holds.*

Proposition 7. *$g(m) \in T_k$ holds for any $m \in M$ and $k < \omega$.*

Proof. We prove the proposition by induction on k. Since it trivially holds for $k = 0$, we assume it holds for k and show it holds for $k + 1$.

By the induction hypothesis $g(m) \in T_k$. It can easily been checked that $g(m) \in \mathrm{Con}_\Diamond(T_k)$ using Proposition 6.

In order to show $g(m) \in \mathrm{Con}_\mu(T_k)$, take $D \in \mathcal{D}_\mu$. For $\alpha \in \mathrm{On}$ let $S(\alpha, m) = \{\varphi \in \mathrm{Lean} \cap D \mid m \models \varphi \text{ and } \mathrm{rank}(\varphi, m) < \alpha\}$. It is clear by definition of $g(m)$ that $(g(m), S(\alpha, m)) \in \mathrm{TS}(T, D)$ holds. It is possible to show that for any $\alpha \in \mathrm{On}$, $(g(m), S(\alpha, m)) \in V_J$ where $J = J(D, T_k)$. We leave the details to the reader. Then we take α large enough (for example larger than the cardinality of M), and we have $S(\alpha, m) = D \cap \mathrm{sat}_{\mathrm{T}}(g(m))$. □

Using this proposition, completeness is almost clear: since $m_{\mathrm{I}} \models \varphi_{\mathrm{I}}$, we have $g(m_{\mathrm{I}}) \Vdash \varphi_{\mathrm{I}}$ as we already noticed. From Proposition 7, $g(m_{\mathrm{I}}) \in T_K$ holds.

4.3 Soundness

In this subsection we prove that the decision procedure is sound, that is, (2) $\Longrightarrow$ (1) of Theorem 1.

Take $t_{\mathrm{I}} \in T_K$ with $t_{\mathrm{I}} \Vdash \varphi_{\mathrm{I}}$ in the condition (2) of Theorem 1. Let $L = \{(t, S, D) \mid D \in \mathcal{D}_\mu, (t, S) \in V_J = V_{J(D,T_K)}(D, T_K)\}$ and let $\tau : \mathcal{D}_\mu \to \mathcal{D}_\mu$ be a cyclic permutation on $\mathcal{D}_\mu$. Also pick $D_0 \in \mathcal{D}_\mu$. We construct a finite branching tree (but possibly infinite) $\hat{T}$ with elements of L as the label of each node as follows: Prepare a node n_{I} labelled by $(t_I, D_0 \cap \mathrm{sat}_{\mathrm{T}}(t_I), D_0)$ and make it the root of tree $\hat{T}$. This node has not yet been "processed."

Then pick an unprocessed node n whose position is one of the shallowest. Let (t, S, D) be the label of n. For each $\varphi \in \mathrm{PO}(\mathrm{cl}(\varphi_{\mathrm{I}}); \langle\rangle)$ such that $t \Vdash \varphi$, we create a node $n' = \mathrm{Succ}(n, \varphi)$ as a child of node n. If $S \neq \varnothing$, let $\varphi = \langle a \rangle \psi$ and let $j = \min\{j \leq J \mid \varphi \in (t, S) \in V_{j+1}\}$. Since $(t, S) \in \mathrm{Step}_D(V_j, T_K)$, take $(t', S') \in V_j$ that satisfies the three conditions in the definition of $\mathrm{Step}_D(V_j, T_K)$ and make it the label of node n'. If $S = \varnothing$, take any $t' \in T_K$ such that $t \xrightarrow{\varphi} t'$ (such one exists since $t \in \mathrm{Con}_{\Diamond}(T_K)$) and make $(t', \tau(D) \cap \mathrm{sat}_{\mathrm{T}}(t'), \tau(D))$ the label of node n'. That completes the construction of $\hat{T}$.

We denote the label of node n by $l(n)$. When $l(n) = (t, S, D)$, we denote t, S and D by $t(n)$, $S(n)$ and $D(n)$ respectively. Let $j(n) = \min\{j < \omega \mid (t(n), S(n)) \in V_j(D(n), T_K)\}$. The following proposition is clear from the construction of $\hat{T}$.

Proposition 8. *Suppose $(n_k \mid k < \omega)$ is an enumeration of a postfix of an infinite branch of $\hat{T}$, that is for any $k < \omega$ there exists φ such that $n_{k+1} = \mathit{Succ}(n_k, \varphi)$.*
(1) For any $k < \omega$ there is $k' > k$ such that $S(n_{k'}) = \varnothing$.
(2) For any $D \in \mathcal{D}_\mu$ there exists $k < \omega$ such that $D(n_k) = D$ and $S(n_k) = D \cap \mathit{sat}_T(t(n_k))$

We define a Kripke structure $\mathcal{M} = (M, R, \lambda)$ as follows: M is the set of all nodes of tree $\hat{T}$. $R(a) = \{(n_1, n_2) \mid n2 = \mathrm{Succ}(n_1, \langle a \rangle \varphi)$ for some $\langle a \rangle \varphi$ or $n1 = \mathrm{Succ}(n_2, \langle \overline{a} \rangle \varphi)$ for some $\langle \overline{a} \rangle \varphi\}$. And for $P \in \mathrm{PC}$, $\lambda(P) = \{n \in M \mid t(n) \Vdash P\}$. We will show $\mathcal{M}, n_{\mathrm{I}} \models \varphi_{\mathrm{I}}$. Let $Q = \{(\varphi, n) \mid t(n) \Vdash \varphi\}$ and we will show $Q \subseteq \mathrm{Sat}(\mathcal{M})$ by checking the three conditions in Proposition 3, where the choice function F is defined as $F(\varphi_1 \vee \varphi_2, n) = \varphi_1 \iff (\overline{t(n)}(\varphi_1) <_{\mathrm{L}} \overline{t(n)}(\varphi_2))$ or $(\overline{t(n)}(\varphi_1) = \overline{t(n)}(\varphi_2)$ and $t(n)_{S(n)} \Vdash \varphi_1)$, $F(\langle a \rangle \varphi, n) = \mathrm{Succ}(n, \langle a \rangle \varphi)$. Conditions (1) and (2) among the three can easily be checked.

In the rest of this subsection we prove condition (3). Suppose, on the contrary, it does not hold. There are $D \in \mathcal{D}_\mu$ and sequence $((\varphi_i, n_i) \mid i < \omega)$ satisfying $\varphi_i \in D$, $n_i \in M$, $(\varphi_i, n_i) \in Q$ and $(\varphi_i, n_i)\, R_F\, (\varphi_{i+1}, n_{i+1})$ $(i < \omega)$.

Proposition 9. *Suppose $i \leq j$, $n_i = n_j$, $t = t(n_i) = t(n_j)$. Then $\bar{t}(\varphi_j) \leq_L \bar{t}(\varphi_i)$ holds. Furthermore if there exists k such that $i < k < j$ and $n_i \neq n_k$, $\bar{t}(\varphi_j) <_L \bar{t}(\varphi_i)$ holds.*

This proposition can be proved by induction on $j - i$ from the fact that adjacent nodes of the tree has the relation LoopSafe. We leave the details to the reader.

Since the formulas are in normal forms the sequence cannot stay at a node infinitely long: for each $i < \omega$ there exists $j > i$ such that $n_j \neq n_i$. By Proposition 9, for each node $n \in M$ there are only finitely many i such that $n_i = n$. Therefore there is a subsequence $(n_{i(k)} \mid k < \omega)$ that forms a postfix of an infinite branch of tree $\hat{T}$. There is variation on how to pick $i(k)$ but we take the largest one, that is, if $i > i(k)$ then $n_i \neq n_{i(k)}$.

Take $k < \omega$ such that $D(n_{i(k)}) = D$ and $S(n_{i(k)}) = D \cap \mathrm{sat}_{\mathrm{T}}(n_{i(k)})$ by Proposition 8(2). Clearly $t(n_{i(k)})_{S(n_{i(k)})} \Vdash \varphi_{i(k)}$ We can check that for any $k' >$

k, $t(n_{i(k')})_{S(n_{i(k')})} \Vdash \varphi_{i(k')}$ by induction on k'. But this is impossible for k' with $S(n_{i(k')}) = \varnothing$, which exists by Proposition 8(1), since $\varphi_{i(k')} \in D \cap \mathrm{Lean}$. That completes the proof of soundness of the decision procedure.

5 An Experimental Implementation

We have implemented the decision procedure proposed in Section 3.2. Since the main part of the procedure repeatedly computes subsets of some fixed set, implementation using BDD [9] is suitable.

Table 1 shows the results for a few formulas. l is the length of the formula, *Sat* shows whether the formula is satisfiable or not, v is the number of the BDD variables, n is the number of the BDD nodes, and *Time* shows the execution (elapsed) time in milliseconds.

The formula `lap`n, where n is a natural number, is $p_0 \wedge \bigwedge_{i=1}^{n} \nu X((p_{i-1} \rightarrow \mu Y(p_i \vee \langle a \rangle Y)) \wedge [a]X) \wedge \neg \mu Z((p_n \wedge \mu W(p_0 \vee W)) \vee \langle \overline{a} \rangle Z)))$. Its intention is "At the start point p_0 holds and if p_{i-1} holds one can find a point where p_i holds by following arrows labelled by a. But one cannot find a point where p_n holds and connected to a point where p_0 holds with reverse arrows labelled by a." No model satisfies this formula. The formulas `loop`n and `tree`n are constructed in a similar fashion. There is a $D \in \mathcal{D}_\mu$ such that the size of BForm_D is $2n$ in the formula `loop`n, while BForm_D is an empty set in the other formulas for all $D \in \mathcal{D}_\mu$. The formula `tree`n are satisfiable but they do not have finite models, while the other formulas are unsatisfiable.

The experimental implementation is written in Java with Java BDD 1.0, which calls the BuDDy library. Values for the tuning parameters are: nodenum = 2^{20}, maxincrease = 2^{18}, cachesize = variable, cacheratio = 1/64.0. Our experiments have been performed on a Pentium 4 2.4 GHz machine with 512 megabytes of RAM, running Microsoft Windows XP, Sun Java Development Kit version 1.5.0.

Table 1. Experiments

Formula	l	Sat	v	n	Time
`lap8`	73	no	45	$5,9 \times 10^4$	500
`lap15`	129	no	80	3.8×10^6	26625
`loop4`	38	no	61	1.6×10^5	516
`loop7`	59	no	113	4.1×10^6	43016
`tree8`	48	yes	32	6.9×10^3	343
`tree16`	96	yes	64	1.4×10^6	5875

6 Conclusion and Future Work

In this study we proposed an effective decision procedure for checking satisfiability of formulas in the alternation-free two-way modal μ-calculus and reported an experimental implementation of the procedure.

In this section we describe two directions for future work. The first direction is to expand the range of logics where the method is applicable. A natural extension of the two-way modal μ-calculus is μ-LGF, which extends the decidable subsystem, LGF (Loosely Guarded Fragment), of the first-order predicate logic with fixed point operators. A general decision procedure is already known for μ-LGF [18]. It is a complex procedure using alternating tree automata as in the two-way modal μ-calculus. Therefore it is meaningful to find an appropriate subsystem of μ-LGF to which the method in this study can be applied. An obvious attempt would be to define the alternation-free μ-LGF, in which we restrict fixed operators not to alternate. However, it does not work well since in the subsystem, witnesses for the least fixed point operator are not necessarily finite sets. Remember that this finiteness is the key property to apply our method. Finding an appropriate subsystem in which witnesses are always finite is an issue.

The second direction is to apply the decision procedure to verification problems. As mentioned in the introduction, we already employed a decision procedure for judging satisfiability of two-way CTL formulas for analyzing properties of cellular automata. Replacing it with the decision procedure in this study will make more properties verifiable. We also have a plan to apply it to software model checking [19]. In the predicate abstraction framework, on which many software model checking tools are based, satisfiability checking for properties on numbers is used to obtain a small state space that abstracts an original huge state space [20, 21]. Temporal logics such as the two-way modal μ-calculus has enough power to describe properties of a heap when we regard a heap as a Kripke structure with the "points-to" relation as the transition relation. Therefore the decision procedure developed in this study may be useful to build an abstract state space for properties on a heap.

References

1. Emerson, E.A., Clarke, E.M.: Using branching-time temporal logic to synthesize synchronization skeletons. Science of Computer Programming **2**(3) (1982) 241–266
2. Manna, Z., Wolper, P.: Synthesis of communicating processes from temporal logic specifications. ACM Transactions on Programming Languages and Systems **6**(1) (1984) 68–93
3. Takahashi, K., Hagiya, M.: Abstraction of graph transformation using temporal formulas. In: Supplemental Volume of the 2003 International Conference on Dependable Systems and Networks (DNS-2003). (2003) W–65 to W–66
4. Hagiya, M., Takahashi, K., Yamamoto, M., Sato, T.: Analysis of synchronous and asynchronous cellular automata using abstraction by temporal logic. In: FLOPS2004: The Seventh Functional and Logic Programming Symposium. Volume 2998 of Lecture Notes in Computer Science. (2004) 7–21

5. Tozawa, A.: On binary tree logic for XML and its satisifiability test. In: Sixth Workshop on Programming and Programming Language (PPL2004). (2004)
6. Vardi, M.Y.: Reasoning about the past with two-way automata. In: Automata, Languages and Programming, 25th International Colloquium, ICALP'98. Volume 1443 of Lecture Notes in Computer Science. (1998) 628–641
7. Safra, S.: On the complexity of omega-automata. In: Proceedings of the 29th Annual Symposium on Foundations of Computer Science, FoCS '88, IEEE Computer Society Press (1988) 319–327
8. Grädel, E., Thomas, W., Wilke, T., eds.: Automata, Logics, and Infinite Games: A Guide to Current Research. In Grädel, E., Thomas, W., Wilke, T., eds.: Automata, Logics, and Infinite Games. Volume 2500 of Lecture Notes in Computer Science., Springer (2002)
9. Bryant, R.E.: Symbolic boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys **24**(3) (1992) 293–318
10. McMillan, K.: Symbolic Model Checking. Kluwer Academic Publ. (1993)
11. Pan, G., Sattler, U., Vardi, M.Y.: BDD-based decision procedures for K. In: Proceedings of the 18th International Conference on Automated Deduction, Springer-Verlag (2002) 16–30
12. Pan, G., Vardi, M.Y.: Optimizing a BDD-based modal solvar. In: 19th International Conference on Automated Deduction. Volume 2741 of Lecture Notes in Computer Science. (2003) 75–89
13. Henriksen, J.G., Jensen, J.L., Jørgensen, M.E., Klarlund, N., Paige, R., Rauhe, T., Sandholm, A.: Mona: Monadic second-order logic in practice. In: Proceedings of the First International Workshop on Tools and Algorithms for Construction and Analysis of Systems. Volume 1019 of Lecture Notes in Computer Science., Springer-Verlag (1995) 89–110
14. Tozawa, A., Tanabe, Y., Hagiya, M.: Experiments on global type checking and termination checking for XML transducer. (Submitted)
15. Niwinski, D., Walukiewicz, I.: Games for the μ-calculus. Theoretical Computer Science **163**(1,2) (1996) 99–116
16. Kozen, D.: Results on the propositional μ-calculus. Theoretical Computer Science **27** (1983) 333–354
17. Emerson, E.A.: Temporal and modal logic. In: Handbook of theoretical computer science (vol. B): formal models and semantics, Elsevier Science Publishers B.V. (1990) 995–1072
18. Grädel, E.: Guarded fixed point logics and the monadic theory of countable trees. Theoretical Computer Science **288** (2002) 129–152
19. Yamamoto, M., Tanabe, Y., Takahashi, K., Hagiya, M.: Abstraction of graph transformation systems by temporal logic and its verification. (Submitted)
20. Ball, T., Rajamani, S.K.: Automatically validating temporal safety properties of interfaces. In: Proceedings of the 8th international SPIN workshop on Model checking of software, Springer-Verlag New York, Inc. (2001) 103–122
21. Henzinger, T.A., Jhala, R., Majumdar, R., Sutre, G.: Lazy abstraction. In: Proceedings of the 29th Annual Symposium on Principles of Programming Languages, ACM Press (2002) 58–70

無交代 2 方向様相 μ 計算の決定手続き (in English)
(算譜科学研究速報)
発行日：2005 年 6 月 15 日
編集・発行：独立行政法人産業技術総合研究所関西センター尼崎事業所
システム検証研究センター
同連絡先：〒661-0974 兵庫県尼崎市若王寺 3-11-46
e-mail：informatics-inquiry@m.aist.go.jp

A Decision Procedure for Alternation-free Two-way Modal mu-calculus
(Programming Science Technical Report)
June 15, 2005
Research Center for Verification and Semantics (CVS)
AIST Kansai, Amagasaki Site
National Institute of Advanced Industrial Science and Technology (AIST)
3-11-46 Nakouji, Amagasaki, Hyogo, 661-0974, Japan
e-mail: informatics- inquiry@m.aist.go.jp