Proving $\forall\mu$-Calculus Properties with SAT-Based Model Checking*

Bow-Yaw Wang

Institute of Information Science, Academia Sinica,
Taipei, Taiwan

Abstract. In this paper, we present a complete bounded model checking algorithm for the universal fragment of μ-calculus. The new algorithm checks the completeness of bounded proof of each property on the fly and does not depend on prior knowledge of the completeness thresholds. The key is to combine both local and bounded model checking techniques and use SAT solvers to perform local model checking on finite Kripke structures. Our proof-theoretic approach works for any property in the specification logic and is more general than previous work on specific properties. We report experimental results to compare our algorithm with the conventional BDD-based algorithm.

1 Introduction

Due to the limitation of BDD-based model checking on large designs, SAT-based bounded model checking has become a supplementary verification technique in recent years [1, 2]. Different from model checking [3, 4], bounded model checking focuses on catching design flaws within a bounded number of steps, and therefore does not guarantee the design to be free from errors. Naturally, one wonders whether bounded model checking can be extended to be complete.

There is a bound (called *completeness threshold*) such that the absence of flaws within the completeness threshold implies the satisfiability of the property [1, 5]. One often uses over-approximations of the completeness threshold in practice since computing the exact value is hard. But redundant computation incurred by approximations may impede the performance. Promising alternatives are available for checking linear properties, where the completeness of bounded model checking can also be determined dynamically [6–9]. However, the dynamic completeness criteria for branching-time properties are still missing.

In this paper, we propose a new framework for proving temporal properties by bounded model checking. Similar to [6–9], our algorithm determines the completeness of bounded model checking on the fly to avoid redundant computation. We use the universal fragment of propositional μ-calculus as the formalism for property specification. With the standard embedding [10–12], linear- and fragments of branching-time temporal logics are subsumed by our framework. Our

* This work was supported in part by NSC grand NSC 93-2213-E-001-012-.

F. Wang (Ed.): FORTE 2005, LNCS 3731, pp. 113–127, 2005.

technique therefore opens up opportunities for developing new complete bounded model checking algorithms.

The key concept is to combine bounded and local model checking techniques. Local model checking (also known as tableau-based model checking) tries to find a proof for the property by exploring neighboring states [13–15]. The proof search in local model checking algorithms is not unlike those of bug hunting in bounded model checking: a flaw is nothing but a "local" proof of the negation of the given property. The completeness of the proof rules in local model checking ensures that a flaw can always be found in finite models, should one exist.

We therefore propose an algorithm that reduces the proof search in local model checking to Boolean satisfiability. Since the negation of any formula in the universal fragment of μ-calculus belongs to the existential fragment of μ-calculus, we look for design flaws by finding proofs for arbitrary formula in the fragment. For any formula in the fragment, we construct a Boolean formula for it. The satisfiability of the Boolean formula is shown to be equivalent to the existence of a bounded proof in local model checking. Additionally, we show that the unsatisfiability of a similar Boolean formula implies the absence of proofs. The latter formula allows our algorithm to check the completeness criterion dynamically. Since the criterion is proof-theoretic, it is valid for *all* properties in the specification logic. Our technique gives a proof-theoretic interpretation of the completeness criteria and is more general than those in [6, 7, 9].

A major advantage of our technique is to verify many more properties by the use of standard encodings. For instance, $\forall$CTL [10, 12] and the universal fragment of Fair CTL [16] can be verified by embedding them into the universal fragment of μ-calculus. Our framework gives a unified theory of completeness criteria, which cannot be found in previous works. Additionally, the verification of linear-time temporal logic can be reduced to checking fairness constraints by the automata-theoretic technique [11]. Our technique is also applicable for linear properties.

The remainder of this paper is organized as follows. After discussing related work in Section 1.1, preliminaries are given in Section 2. Section 3 recalls the local model checking proof rules. The main technical results are shown in Section 4. Experimental results are presented in Section 5. Finally, in Section 6, we present our conclusions and discuss the future work.

1.1 Related Work

The inductive method was originally proposed as a heuristic for proving properties in bounded model checking. Later, it was improved and made complete for safety [6, 7] and liveness [17] properties. In the complete inductive method, if the induction proves the property or the completeness criterion is met, the algorithm reports that the property is satisfied. Otherwise, it looks for design flaws within the current bound.

A more direct approach for LTL model checking is reported in [9]. The authors give characterizations for LTL formulae of the form $\neg Gp$, $\neg FG\neg p$, and $\neg Fp$. Using the automata-theoretic technique developed in [11], the LTL model

checking problem is reduced to verifying $FG\neg p$ and solved in [9]. For special cases such as Gp and Fp, [9] shows how to verify these properties directly.

State traversal can be simulated by exploiting conflict analysis in SAT solvers as well [8]. Given two conflicting Boolean formulae A and B, an interpolant P of A and B is a formula that is implied by A but conflicts with B. If A represents the initial states and B represents the set of states that violate the property, their interpolants can be understood as under-approximations of "bad" states. The interpolation is then combined with bounded model checking to verify linear temporal properties in [8].

The reduction of proof search in local model checking to satisfiability can also be found in [18, 19], in which the authors reduce the local model checking problem to Presburger arithmetic for infinite-state systems. Due to the undecidability of the μ-calculus model checking problem on infinite-state systems, the completeness of the algorithms in [18, 19] is not the main concern of the authors. For the invariant and inevitable properties on finite-state systems, the present work extends and subsumes the complete algorithms in [20].

To the best of our knowledge, estimating the completeness threshold is still required in order to prove fragments of branching-time temporal logics in bounded model checking [1]. Ideally, one would like to apply the techniques in [6–9] to develop similar on-the-fly completeness criteria for branching-time temporal logics. However, the techniques used in [6–9] are based essentially on closely examining paths of interest. It is unclear whether the approach would work for branching-time temporal logics. Additionally, our proof-theoretic approach gives general completeness criteria for fragments of branching-time temporal logics, not only particular temporal properties.

2 Preliminaries

We use the universal fragment of μ-calculus as the specification logic for temporal properties [21]. A μ-calculus formula ψ is defined recursively as follows.

- Propositional variables (PV): $X, Y, Z, \ldots$;
- Atomic propositions (AP): $p, q, \ldots$;
- Boolean operators: $\neg\psi$, $\psi \wedge \psi'$;
- The modal existential next-state operator: $\Diamond\psi$;
- The least fixed-point operator: $\mu X.\psi$, where the bound propositional variable X occurs positively in ψ.

As usual, derived operators such as the disjunctive operator $\psi \vee \psi'$ ($\equiv \neg(\neg\psi \wedge \neg\psi')$), the modal universal next-state operator $\Box\psi$ ($\equiv \neg\Diamond\neg\psi$) and, the greatest fixed-point operator $\nu X.\psi$ ($\equiv \neg\mu X.\neg\psi[\neg X/X]$, where $\neg\psi[\neg X/X]$ is obtained by substituting $\neg X$ for X in $\neg\psi$) are used. A μ-calculus formula ψ is *normal* if all negations only apply to atomic propositions. The universal fragment of μ-calculus (denoted $\forall\mu$-calculus) formulae are those without modal existential next-state operators in their normal forms. Similarly, $\exists\mu$-calculus formulae are those without modal universal next-state operators. By α-conversion, it suffices

to consider μ-calculus formula ψ whose nested bound propositional variables are distinct.

Let $\mathbb{B} = \{\mathsf{false}, \mathsf{true}\}$ be the Boolean domain and $\mathbb{N}$ the natural numbers (non-negative integers). A *state* (denoted by $\bar{r}, \bar{s}, \bar{t}, \ldots$) is a Boolean vector of size $n > 0$. Let V be a set of Boolean variables, and $\bar{u}, \bar{v}, \bar{w} \in V^n$ be vectors of Boolean variables of size n. Equivalently, we may think of a state as a *valuation* $[\![\bar{u}]\!]\rho$ for $\bar{u}$, where $\rho \in V \to \mathbb{B}$ is an assignment of Boolean variables. A *Kripke structure* is a tuple $K = (\mathbb{B}^n, I, \to, L)$, where $I \subseteq \mathbb{B}^n$ is the set of initial states, $\to \subseteq \mathbb{B}^n \times \mathbb{B}^n$ is the total transition relation, and $L : \mathbb{B}^n \to 2^{AP}$ is the labeling function that maps each state to the atomic propositions satisfied in that state. We write $\bar{s} \to \bar{t}$ for $(\bar{s}, \bar{t}) \in \to$.

Let $\epsilon \in PV \to 2^{\mathbb{B}^n}$ be an environment for propositional variables. Given a propositional variable X and a set of states R, the environment $\epsilon[X \mapsto R]$ assigns X to R, but keeps other propositional variables Y assigned to $\epsilon(Y)$. The semantic function $[\psi]\epsilon \subseteq \mathbb{B}^n$ for the μ-calculus formula ψ and the environment ϵ is defined as follows.

$$\begin{aligned}
[X]\epsilon &= \epsilon(X) \\
[p]\epsilon &= \{\bar{s} \in \mathbb{B}^n : p \in L(\bar{s})\} \\
[\neg\psi]\epsilon &= \mathbb{B}^n \setminus [\psi]\epsilon \\
[\psi \wedge \psi']\epsilon &= [\psi]\epsilon \cap [\psi']\epsilon \\
[\Diamond\psi]\epsilon &= \{\bar{s} \in \mathbb{B}^n : \exists \bar{t} \in \mathbb{B}^n . \bar{s} \to \bar{t} \text{ and } \bar{t} \in [\psi]\epsilon\} \\
[\mu X.\psi]\epsilon &= \bigcap \{R \subseteq \mathbb{B}^n : [\psi](\epsilon[X \mapsto R]) \subseteq R\}.
\end{aligned}$$

The characteristic functions of p, I, and $\to$ are denoted by χ_p, χ_I, and $\chi_\to$ respectively. Let $\bar{u}$ and $\bar{u}'$ be vectors of Boolean variables representing current and next states respectively. Then $\chi_p(\bar{u})$ is satisfied by ρ if and only if $[\![\bar{u}]\!]\rho$ is a state satisfying the atomic proposition p. Similarly, $\chi_I(\bar{u})$ is satisfied by an assignment ρ if and only if the state $[\![\bar{u}]\!]\rho$ is an initial state, and $\chi_\to(\bar{u}, \bar{u}')$ is satisfied by ρ if and only if the state $[\![\bar{u}]\!]\rho$ is followed by $[\![\bar{u}']\!]\rho$ in K.

Let ψ be a μ-calculus formula, $K = (\mathbb{B}^n, I, \to, L)$ a Kripke structure and $\bar{s}$ a state. We write $K, \bar{s} \models \psi$ if $\bar{s} \in [\psi]\emptyset$; if $K, \bar{s}_0 \models \psi$ for all initial states $\bar{s}_0 \in I$, we denote it by $K \models \psi$. The model checking problem is to determine whether $K \models \psi$.

In [13–15], several tableau-based μ-calculus model checking algorithms were developed. The proof rules in [13, 14] were simplified in [22, 15] by extending fixed point operators to:

$$\sigma X\{\bar{r}_0 \cdots \bar{r}_m\}\Phi,$$

where σ can be either of the fixed point operators and $\bar{r}_0, \ldots, \bar{r}_m$ are states. Intuitively, $\bar{r}_0, \ldots, \bar{r}_m$ record visited states in the fixed-point formulae. The semantics of the new operators are defined accordingly:

$$\begin{aligned}
[\mu X\{\bar{r}_0 \cdots \bar{r}_m\}\psi]\epsilon &= \bigcap \{R \subseteq \mathbb{B}^n : [\psi](\epsilon[X \mapsto R]) \setminus \{\bar{r}_0 \cdots \bar{r}_m\} \subseteq R\} \\
[\nu X\{\bar{r}_0 \cdots \bar{r}_m\}\psi]\epsilon &= \bigcup \{R \subseteq \mathbb{B}^n : R \subseteq [\psi](\epsilon[X \mapsto R]) \cup \{\bar{r}_0 \cdots \bar{r}_m\}\}.
\end{aligned}$$

The *extended μ-calculus* uses extended fixed point operators instead. Note that $\sigma X\{\}\psi \equiv \sigma X.\psi$; hence, any μ-calculus formula can be transformed into an equivalent extended μ-calculus formula syntactically.

3 Proof Rules

Different from global model checking algorithms in [10, 12, 4], the algorithms developed in [13–15, 22] search for a proof for the given μ-calculus property at an initial state by exploring the Kripke structure locally. It is noted that the worst-case complexity of the tableau-based algorithms remains the same as the conventional algorithms [13]. However, the proof-theoretic algorithms would be more efficient if the property could be proved locally.

Figure 1 shows the proof rules for $\exists\mu$-calculus model checking. Given a Kripke structure K, a state $\bar{s}$, and a μ-calculus formula ψ, a *judgment* is of the form $K, \bar{s} \vdash \psi$. Given a judgment, a *proof* is a tree constructed according to the proof rules in Figure 1. Note that the rules ($\neg\neg$), ($\vee$L), ($\vee$R), ($\neg\vee$), ($\wedge$), ($\neg\wedge$L), ($\neg\wedge$R), ($\Diamond$), ($\neg\Box$), (σ-Unroll), and ($\neg\sigma$-Unroll) reduce the current judgment to one or more judgments to be justified later. We therefore say a proof is *full* if all of its leaves are instances of the rules (AP), ($\neg$AP), (ν-Term), or ($\neg\mu$-Term).

Since we are interested in constructing Boolean formulae for $\exists\mu$-calculus in this work, Figure 1 omits the corresponding rules for the universal modal operator, which are given in [13–15]. The full proof rules are sound and complete for finite Kripke structures:

Theorem 1. *([13–15]) Let $K = (\mathbb{B}^n, I, \rightarrow, L)$ be a Kripke structure, $\bar{s} \in \mathbb{B}^n$, and ψ a μ-calculus formula. Then*

$$K, \bar{s} \vdash \psi \text{ has a full proof if and only if } K, \bar{s} \models \psi.$$

4 Proof Search by SAT

To motivate our reduction of proof search to Boolean satisfiability, consider the safety property AGp. Suppose a flaw satisfying $EF\neg p (\equiv \neg AGp \equiv \mu X\{\}\neg p \vee \Diamond X)$ is found in one step. The corresponding Boolean formula generated by one of the complete inductive methods in [6] is

$$\chi_I(\bar{v}_0) \wedge \chi_{\rightarrow}(\bar{v}_0, \bar{v}_1) \wedge \neg\chi_p(\bar{v}_1) \wedge \bigwedge_{0 \le i < j \le 1} \bar{v}_i \neq \bar{v}_j. \tag{1}$$

Let the satisfying Boolean assignment be ρ. The following full proof for the judgment $K, [\![\bar{v}_0]\!]\rho \vdash \mu X\{\}\neg p \vee \Diamond X$ can be constructed by the proof rules in Figure 1,

$$\frac{p \in L(\bar{s})}{K, \bar{s} \vdash p}\ (\text{AP}) \qquad \frac{p \notin L(\bar{s})}{K, \bar{s} \vdash \neg p}\ (\neg\text{AP})$$

$$\frac{K, \bar{s} \vdash \psi}{K, \bar{s} \vdash \neg\neg\psi}\ (\neg\neg)$$

$$\frac{K, \bar{s} \vdash \psi}{K, \bar{s} \vdash \psi \vee \psi'}\ (\vee\text{L}) \qquad \frac{K, \bar{s} \vdash \psi'}{K, \bar{s} \vdash \psi \vee \psi'}\ (\vee\text{R})$$

$$\frac{K, \bar{s} \vdash \neg\psi \quad K, \bar{s} \vdash \neg\psi'}{K, \bar{s} \vdash \neg(\psi \vee \psi')}\ (\neg\vee)$$

$$\frac{K, \bar{s} \vdash \psi \quad K, \bar{s} \vdash \psi'}{K, \bar{s} \vdash \psi \wedge \psi'}\ (\wedge)$$

$$\frac{K, \bar{s} \vdash \neg\psi}{K, \bar{s} \vdash \neg(\psi \wedge \psi')}\ (\neg\wedge\text{L}) \qquad \frac{K, \bar{s} \vdash \neg\psi'}{K, \bar{s} \vdash \neg(\psi \wedge \psi')}\ (\neg\wedge\text{R})$$

$$\frac{K, \bar{t} \vdash \psi \quad \bar{s} \to \bar{t}}{K, \bar{s} \vdash \Diamond\psi}\ (\Diamond) \qquad \frac{K, \bar{t} \vdash \neg\psi \quad \bar{s} \to \bar{t}}{K, \bar{s} \vdash \neg\Box\psi}\ (\neg\Box)$$

$$\frac{\bar{s} \in \{\bar{r}_0 \cdots \bar{r}_m\}}{K, \bar{s} \vdash \nu X\{\bar{r}_0 \cdots \bar{r}_m\}\psi}\ (\nu\text{-Term})$$

$$\frac{K, \bar{s} \vdash \psi[\nu X\{\bar{r}_0 \cdots \bar{r}_m \bar{s}\}\psi/X] \quad \bar{s} \notin \{\bar{r}_0 \cdots \bar{r}_m\}}{K, \bar{s} \vdash \nu X\{\bar{r}_0 \cdots \bar{r}_m\}\psi}\ (\nu\text{-Unroll})$$

$$\frac{K, \bar{s} \vdash \neg\psi[\nu X\{\bar{r}_0 \cdots \bar{r}_m \bar{s}\}\psi/X] \quad \bar{s} \notin \{\bar{r}_0 \cdots \bar{r}_m\}}{K, \bar{s} \vdash \neg\nu X\{\bar{r}_0 \cdots \bar{r}_m\}\psi}\ (\neg\nu\text{-Unroll})$$

$$\frac{\bar{s} \in \{\bar{r}_0 \cdots \bar{r}_m\}}{K, \bar{s} \vdash \neg\mu X\{\bar{r}_0 \cdots \bar{r}_m\}\psi}\ (\neg\mu\text{-Term})$$

$$\frac{K, \bar{s} \vdash \neg\psi[\mu X\{\bar{r}_0 \cdots \bar{r}_m \bar{s}\}\psi/X] \quad \bar{s} \notin \{\bar{r}_0 \cdots \bar{r}_m\}}{K, \bar{s} \vdash \neg\mu X\{\bar{r}_0 \cdots \bar{r}_m\}\psi}\ (\neg\mu\text{-Unroll})$$

$$\frac{K, \bar{s} \vdash \psi[\mu X\{\bar{r}_0 \cdots \bar{r}_m \bar{s}\}\psi/X] \quad \bar{s} \notin \{\bar{r}_0 \cdots \bar{r}_m\}}{K, \bar{s} \vdash \mu X\{\bar{r}_0 \cdots \bar{r}_m\}\psi}\ (\mu\text{-Unroll})$$

Fig. 1. Proof Rules

where Γ and Δ stand for $[\![\bar{v}_1]\!]\rho \notin \{[\![\bar{v}_0]\!]\rho\}$ and $[\![\bar{v}_0]\!]\rho \to [\![\bar{v}_1]\!]\rho$ respectively:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \notin L([\![\bar{v}_1]\!]\rho)}{K, [\![\bar{v}_1]\!]\rho \vdash \neg p}\ (\neg\text{AP})}{K, [\![\bar{v}_1]\!]\rho \vdash \neg p \vee \Diamond\mu X\{[\![\bar{v}_0]\!]\rho[\![\bar{v}_1]\!]\rho\}\neg p \vee \Diamond X}\ (\vee\text{L}) \quad \Gamma}{K, [\![\bar{v}_1]\!]\rho \vdash \mu X\{[\![\bar{v}_0]\!]\rho\}\neg p \vee \Diamond X}\ (\mu\text{-Unroll}) \quad \Delta}{K, [\![\bar{v}_0]\!]\rho \vdash \Diamond\mu X\{[\![\bar{v}_0]\!]\rho\}\neg p \vee \Diamond X}\ (\Diamond)$$
$$\dfrac{K, [\![\bar{v}_0]\!]\rho \vdash \neg p \vee \Diamond\mu X\{[\![\bar{v}_0]\!]\rho\}\neg p \vee \Diamond X}{K, [\![\bar{v}_0]\!]\rho \vdash \mu X\{\}\neg p \vee \Diamond X}\ (\mu\text{-Unroll})}\ (\vee\text{R})$$

It is easy to see that the Boolean formula $\chi_{\rightarrow}(\bar{v}_0, \bar{v}_1)$ in (1) corresponds to the second antecedent of the rule ($\Diamond$), and the formula $\bigwedge_{0 \leq i < j \leq 1} \bar{v}_i \neq \bar{v}_j$ to the second antecedent of the rule (μ-Unroll). Finally, the antecedent of rule ($\neg$AP) is discharged by the satisfiability of $\neg\chi_p(\bar{v}_1)$. Roughly, there is a Boolean subformula for each application of the proof rule ($\neg$AP), ($\Diamond$), and (μ-Unroll) respectively. We generalize the idea and construct a Boolean formula for each rule in Figure 1 so that the satisfiability of the Boolean formula is equivalent to the existence of subproofs.

A syntactic extension of μ-calculus formulae is needed in the following presentation. Consider the formula $\sigma X\{\bar{r}_0 \cdots \bar{r}_m\}\psi$, where $\bar{r}_0 \cdots \bar{r}_m$ are states. Since states $\bar{r}_i$'s are denoted by variable vectors $\bar{v}_i$'s to be determined by SAT solvers, we allow the syntactic extension $\sigma X\{\bar{v}_0 \cdots \bar{v}_m\}\psi$ in our construction. Formulae constructed by Boolean operators, modal operators, and the syntactic extension of fixed point operators are called *schematic μ-calculus* formulae. If ρ is an assignment to Boolean variables, define

$$
\begin{aligned}
[\![p]\!]\rho &= p \\
[\![X]\!]\rho &= X \\
[\![\neg\varphi]\!]\rho &= \neg[\![\varphi]\!]\rho \\
[\![\varphi \vee \varphi']\!]\rho &= [\![\varphi]\!]\rho \vee [\![\varphi']\!]\rho \\
[\![\varphi \wedge \varphi']\!]\rho &= [\![\varphi]\!]\rho \wedge [\![\varphi']\!]\rho \\
[\![\Diamond\varphi]\!]\rho &= \Diamond[\![\varphi]\!]\rho \\
[\![\Box\varphi]\!]\rho &= \Box[\![\varphi]\!]\rho \\
[\![\sigma X\{\bar{v}_0 \cdots \bar{v}_m\}\varphi]\!]\rho &= \sigma X\{[\![\bar{v}_0]\!]\rho \cdots [\![\bar{v}_m]\!]\rho\}[\![\varphi]\!]\rho.
\end{aligned}
$$

The mapping $[\![\bullet]\!]\rho$ assigns states to variable vectors appearing in a schematic μ-calculus formula and thereby yielding an extended μ-calculus formula. We say an extended μ-calculus formula ψ is an *instance* of a schematic μ-calculus formula φ if there is an assignment ρ such that $[\![\varphi]\!]\rho = \psi$.

Let $K = (\mathbb{B}^n, I, \rightarrow, L)$ be a Kripke structure, $\bar{u} \in V^n$ and $d \in \mathbb{N}$. Figure 2 shows the translation rules to construct a Boolean formula $\Theta_K(\bar{u}, \varphi, d)$ for any schematic $\exists\mu$-calculus formula φ. Intuitively, the vector of Boolean variables $\bar{u}$ corresponds to the current state, φ the sub-property to be fulfilled at the current state, and d the bound of unrolling. The translation ensures that the satisfiability of the Boolean formula $\Theta_K(\bar{u}, \varphi, d)$ witnessed by the assignment ρ is equivalent to the existence of proof for $[\![\varphi]\!]\rho$ at state $[\![\bar{u}]\!]\rho$. For Boolean and next-state modal operators, consider the rule ($\neg\Box$) as an example. If there is a proof for $[\![\neg\Box\varphi]\!]\rho$ at state $[\![\bar{u}]\!]\rho$, then there is a proof for $[\![\neg\varphi]\!]\rho$ at state $[\![\bar{u}']\!]\rho$ for some $[\![\bar{u}']\!]\rho$ with $[\![\bar{u}]\!]\rho \rightarrow [\![\bar{u}']\!]\rho$. The corresponding Boolean formula is therefore $\chi_{\rightarrow}(\bar{u}, \bar{u}') \wedge \Theta_K(\bar{u}', \neg\varphi, d)$. Other rules can be derived similarly.

For proof of correctness, note that the unrolling of fixed-point subformulae increases the length of a formula. Induction on the lengths of formulae would not work. The following definition is needed in our doubly-inductive proof:

$$\Theta_K(\bar{u}, \nu X\{\bar{v}_0 \dots \bar{v}_m\}\varphi, d) =$$
$$\begin{cases} (\bigwedge_{k=0}^{m} \bar{u} \neq \bar{v}_k) \Leftrightarrow c_i & \text{if } d = 0 \\ (\bigvee_{k=0}^{m} \bar{u} = \bar{v}_k) \vee \Theta_K(\bar{u}, \varphi[\nu X\{\bar{v}_0 \dots \bar{v}_m \bar{u}\}\varphi/X], d-1) & \text{if } d \neq 0 \end{cases}$$
where $c_i \in V$ is a fresh Boolean variable

$$\Theta_K(\bar{u}, \neg\nu X\{\bar{v}_0 \dots \bar{v}_m\}\varphi, d) =$$
$$\begin{cases} (\bigwedge_{k=0}^{m} \bar{u} \neq \bar{v}_k) \wedge c_i & \text{if } d = 0 \\ (\bigwedge_{k=0}^{m} \bar{u} \neq \bar{v}_k) \wedge \Theta_K(\bar{u}, \neg\varphi[\nu X\{\bar{v}_0 \dots \bar{v}_m \bar{u}\}\varphi/X], d-1) & \text{if } d \neq 0 \end{cases}$$
where $c_i \in V$ is a fresh Boolean variable

$$\Theta_K(\bar{u}, \mu X\{\bar{v}_0 \dots \bar{v}_m\}\varphi, d) =$$
$$\begin{cases} (\bigwedge_{k=0}^{m} \bar{u} \neq \bar{v}_k) \wedge c_i & \text{if } d = 0 \\ (\bigwedge_{k=0}^{m} \bar{u} \neq \bar{v}_k) \wedge \Theta_K(\bar{u}, \varphi[\mu X\{\bar{v}_0 \dots \bar{v}_m \bar{u}\}\varphi/X], d-1) & \text{if } d \neq 0 \end{cases}$$
where $c_i \in V$ is a fresh Boolean variable

$$\Theta_K(\bar{u}, \neg\mu X\{\bar{v}_0 \dots \bar{v}_m\}\varphi, d) =$$
$$\begin{cases} (\bigwedge_{k=0}^{m} \bar{u} \neq \bar{v}_k) \Leftrightarrow c_i & \text{if } d = 0 \\ (\bigvee_{k=0}^{m} \bar{u} = \bar{v}_k) \vee \Theta_K(\bar{u}, \neg\varphi[\mu X\{\bar{v}_0 \dots \bar{v}_m \bar{u}\}\varphi/X], d-1) & \text{if } d \neq 0 \end{cases}$$
where $c_i \in V$ is a fresh Boolean variable

$$\Theta_K(\bar{u}, p, d) = \chi_p(\bar{u})$$
$$\Theta_K(\bar{u}, \neg p, d) = \neg\chi_p(\bar{u})$$
$$\Theta_K(\bar{u}, \neg\neg\varphi, d) = \Theta_K(\bar{u}, \varphi, d)$$
$$\Theta_K(\bar{u}, \varphi \wedge \varphi', d) = \Theta_K(\bar{u}, \varphi, d) \wedge \Theta_K(\bar{u}, \varphi', d)$$
$$\Theta_K(\bar{u}, \neg(\varphi \wedge \varphi'), d) = \Theta_K(\bar{u}, \neg\varphi, d) \vee \Theta_K(\bar{u}, \neg\varphi', d)$$
$$\Theta_K(\bar{u}, \varphi \vee \varphi', d) = \Theta_K(\bar{u}, \varphi, d) \vee \Theta_K(\bar{u}, \varphi', d)$$
$$\Theta_K(\bar{u}, \neg(\varphi \vee \varphi'), d) = \Theta_K(\bar{u}, \neg\varphi, d) \wedge \Theta_K(\bar{u}, \neg\varphi', d)$$
$$\Theta_K(\bar{u}, \Diamond\varphi, d) = \chi_{\rightarrow}(\bar{u}, \bar{u}') \wedge \Theta_K(\bar{u}', \varphi, d)$$
where $\bar{u}' \in V^n$ is a vector of fresh Boolean variables

$$\Theta_K(\bar{u}, \neg\Box\varphi, d) = \chi_{\rightarrow}(\bar{u}, \bar{u}') \wedge \Theta_K(\bar{u}', \neg\varphi, d)$$
where $\bar{u}' \in V^n$ is a vector of fresh Boolean variables

Fig. 2. Translation Rules

Definition 1. *Let Γ be a full proof. The* unrolling depth *of a leaf is the number of unrolling rules applied along the path from the root of Γ to the leaf. The unrolling depth of Γ is the maximum over the unrolling depths of all leaves.*

Since the proof of $\neg(\psi \vee \psi')$ is established by the proofs of $\neg\psi$ and $\neg\psi'$, naive structural induction is not applicable in the inner induction. Instead, the following ordering of extended μ-calculus formulae is used:

Definition 2. *Let ψ be an extended μ-calculus formula, then define*

$$\omega(p) = \omega(X) = \omega(\sigma X\{\bar{r}_0 \cdots \bar{r}_m\}\psi) = 0$$
$$\omega(\neg\psi) = \omega(\Diamond\psi) = \omega(\Box\psi) = \omega(\psi) + 1$$
$$\omega(\psi \vee \psi') = \omega(\psi \wedge \psi') = \max(\omega(\psi), \omega(\psi')) + 1$$

Since the function $\omega(\bullet)$ can be extended to schematic μ-calculus formulae straightforwardly, we abuse the notation and write $\omega(\varphi)$ when φ is a schematic μ-calculus formula as well.

Our results can be demonstrated in three steps. First, we consider proofs without unrolling fixed-point subformulae (Lemma 1 and 2). Using atomic propositions and fixed-point subformulae as the basis of inner induction, it can be shown that the existence of proofs is equivalent to the satisfiability of a Boolean formula ($\Omega_K(\bar{u}, \psi, d)$ in Theorem 2). Finally, the unsatisfiability of another Boolean formula ($\Lambda_K(\bar{u}, \psi, d)$ in Theorem 3) can be shown to imply the absence of proofs.[1]

Lemma 1. *Consider any schematic $\exists\mu$-calculus formula φ recursively constructed by $\neg\neg\varphi', \varphi' \wedge \varphi'', \neg(\varphi' \wedge \varphi''), \varphi' \vee \varphi'', \neg(\varphi' \vee \varphi''), \Diamond\varphi'$, or $\neg\Box\varphi'$. Let $\bar{u} \in V^n$ be a vector of Boolean variables and $d \in \mathbb{N}$. Suppose*

- *for all φ' with $\omega(\varphi') < \omega(\varphi)$, if $\Theta_K(\bar{u}, \varphi', d)$ is satisfied by some Boolean assignment ρ', then there is a full proof of unrolling depth d for $\psi' = [\![\varphi']\!]\rho'$ at $\bar{s}' = [\![\bar{u}]\!]\rho'$; and*
- *$\Theta_K(\bar{u}, \varphi, d)$ is satisfied by some Boolean assignment ρ.*

Then, there is a full proof of unrolling depth d for $\psi = [\![\varphi]\!]\rho$ at $\bar{s} = [\![\bar{u}]\!]\rho$.

Lemma 2. *Consider any extended $\exists\mu$-calculus formula ψ recursively constructed by $\neg\neg\psi', \psi' \wedge \psi'', \neg(\psi' \wedge \psi''), \psi' \vee \psi'', \neg(\psi' \vee \psi''), \Diamond\psi'$, or $\neg\Box\psi'$. Let φ be a schematic $\exists\mu$-calculus formula, $\bar{u} \in V^n$ a vector of Boolean variables, and $d \in \mathbb{N}$. Suppose*

- *ψ is an instance of φ;*
- *for all ψ' with $\omega(\psi') < \omega(\psi)$, if there is a full proof of unrolling depth d for ψ' at $\bar{s}'$ and ψ' is an instance of φ', then $\Theta_K(\bar{u}, \varphi', d)$ is satisfied by some Boolean assignment ρ' with $[\![\varphi']\!]\rho' = \psi'$ and $[\![\bar{u}]\!]\rho' = \bar{s}'$; and*
- *there is a full proof of unrolling depth d for ψ at $\bar{s}$.*

Then, $\Theta(\bar{u}, \varphi, d)$ is satisfied by some Boolean assignment ρ with $[\![\varphi]\!]\rho = \psi$ and $[\![\bar{u}]\!]\rho = \bar{s}$.

Lemmas 1 and 2 establish the correspondence between the satisfiability of Boolean formulae and proofs without further unrolling. The following lemma states that the required schematic μ-calculus formula φ in Lemma 2 does indeed exist.

Lemma 3. *Given a proof of an $\exists\mu$-calculus formula at state $\bar{s}_0$, if a judgment $K, \bar{s} \vdash \psi$ occurs in the proof, there is a schematic $\exists\mu$-calculus formula φ such that ψ is an instance of φ.*

For the translation of fixed-point formulae, consider $\nu X\{\bar{v}_0 \cdots \bar{v}_m\}\varphi$ as an example. If there is a proof of unrolling depth d for $[\![\nu X\{\bar{v}_0 \cdots \bar{v}_m\}\varphi]\!]\rho$ at $[\![\bar{u}]\!]\rho$,

[1] For the proofs of technical results, please see [23].

then either $[\![\bar{u}]\!]\rho = [\![\bar{v}_k]\!]\rho$ for some $0 \leq k \leq m$, or $[\![\bar{u}]\!]\rho \neq [\![\bar{v}_k]\!]\rho$ for all $0 \leq k \leq m$ and there is a proof of unrolling depth $d-1$ for $[\![\varphi[\nu X\{\bar{v}_0 \cdots \bar{v}_m \bar{u}\}\varphi/X]]\!]\rho$ at $[\![\bar{u}]\!]\rho$. Thus, we have

$$\Theta_K(\bar{u}, \nu X\{\bar{v}_0 \cdots \bar{v}_m\}\varphi, d) = \bigvee_{k=0}^{m} \bar{u} = \bar{v}_k \vee \Theta_K(\bar{u}, \varphi[\nu X\{\bar{v}_0 \cdots \bar{v}_m \bar{u}\}\varphi/X], d-1).$$

Now suppose the number of unrolling has reached the limit ($d = 0$). The proof of $[\![\nu X\{\bar{v}_0 \cdots \bar{v}_m\}\varphi]\!]\rho$ may be full at $[\![\bar{u}]\!]\rho$, or need be justified by further unrolling. In the translation rule

$$\Theta_K(\bar{u}, \nu X\{\bar{v}_0 \cdots \bar{v}_m\}\varphi, 0) = (\bigwedge_{k=0}^{m} \bar{u} \neq \bar{v}_k) \Leftrightarrow c_i,$$

the fresh variable c_i indicates which of the two cases occurs. If c_i is set to false, then $\bigvee_{k=0}^{m} \bar{u} = \bar{v}_k$ must be true and the proof would be full at $[\![\bar{u}]\!]\rho$. On the other hand, if c_i is true, it implies that $\bigwedge_{k=0}^{m} \bar{u} \neq \bar{v}_k$. The proof need be justified by further unrolling.

We call the fresh Boolean variable c_i used in the translation of $\sigma X\{\bar{v}_0 \cdots \bar{v}_m\}\varphi$ (or $\neg\sigma X\{\bar{v}_0 \cdots \bar{v}_m\}\varphi$) an *expansion variable*. The following theorem states that the existence of proofs and the satisfiability of certain Boolean formulae are equivalent.

Theorem 2. *Let $\bar{u}$ be a vector of Boolean variables, $d \in \mathbb{N}$, ψ an $\exists\mu$-calculus formula, and $c_0, \ldots, c_\ell$ the expansion variables in $\Theta_K(\bar{u}, \psi, d)$. Define $\Omega_K(\bar{u}, \psi, d)$ to be*

$$\Theta_K(\bar{u}, \psi, d) \wedge \bigwedge_{i=0}^{\ell} \neg c_i.$$

- *If $\Omega_K(\bar{u}, \psi, d)$ is satisfied by ρ, then there is a full proof of unrolling depth d for ψ at $\bar{s} = [\![\bar{u}]\!]\rho$.*
- *If there is a full proof of unrolling depth d for ψ at $\bar{s}$, then $\Omega_K(\bar{u}, \psi, d)$ is satisfied by ρ with $[\![\bar{u}]\!]\rho = \bar{s}$.*

With a predetermined completeness threshold $\mathcal{CT}$ for the $\exists\mu$-calculus formula ψ and Kripke structure K, the satisfiability of $\Omega_K(\bar{u}, \psi, \mathcal{CT})$ is equivalent to the existence of a full proof for ψ by Theorem 2. Hence, we have a complete algorithm for $\forall\mu$-calculus properties using completeness thresholds. Since $\forall\mu$-calculus is more expressive than $\forall$CTL, our construction subsumes those in [1].

Determining exact completeness thresholds, however, is hard. We prefer an algorithm that does not use completeness thresholds, but determines the completeness of proofs on the fly. Recall that the expansion variables c_i's are false in Theorem 2. This indicates that proofs do not need further unrolling. If we assume the subproofs of all unjustified fixed-point subformulae indeed exist by setting expansion variables to true, the unsatisfiability of the modified Boolean

formula implies the absence of proof with additional unrolling. The following theorem gives us a completeness criterion in the flavor of [6, 9]:

Theorem 3. *Let $\bar{u}$ be a vector of Boolean variables, $d \in \mathbb{N}$, ψ an $\exists\mu$-calculus formula, and $c_0, \ldots, c_\ell$ the expansion variables in $\Theta_K(\bar{u}, \psi, d)$. Define $\Lambda_K(\bar{u}, \psi, d)$ to be*

$$\Theta_K(\bar{u}, \psi, d) \wedge \bigwedge_{i=0}^{\ell} c_i.$$

If there is a full proof of unrolling depth greater than d for ψ at the state $\bar{s}$, then $\Lambda_K(\bar{u}, \psi, d)$ is satisfied by some Boolean assignment ρ with $[\![\bar{u}]\!]\rho = \bar{s}$

Theorems 2 and 3 are summarized by the algorithm in Figure 3. The algorithm searches proofs incrementally. In each iteration, it first checks whether there is a full proof. If so, it reports "$K, [\![\bar{u}]\!]\rho \vdash \neg\psi$" where ρ is a satisfying assignment. Otherwise, it checks whether full proofs may exist with more unrolling. If not, it reports "ψ is satisfied." Else, the loop is repeated by incrementing the number of unrolling. Observe that the expansion variable c_i forces the condition $\bigwedge_{k=0}^{m} \bar{u} \neq \bar{v}_k$ in $\Theta_K(\bar{u}, \psi, d)$ to be satisfied for each unrolling of fixed-point subformula. Since the number of states is finite, $\bigwedge_{k=0}^{m} \bar{u} \neq \bar{v}_k$ will be unsatisfiable after a finite number of unrolling. By Theorem 3, we conclude that there is no full proof.

Analysis. By generalizing the formula $\sigma X\{\}(\Diamond X \vee \sigma Y\{\}\Diamond(Y \vee X))$, it is easy to see that our construction requires $O(n2^d)$ Boolean variables in general. However,

```
Let ψ be an ∀μ-calculus formula
d ← 0
loop
   if I(ū) ∧ Ω_K(ū, ¬ψ, d) is satisfied by ρ then
      report "K, [[ū]]ρ ⊢ ¬ψ"
   if I(ū) ∧ Λ_K(ū, ¬ψ, d) is unsatisfiable then
      reports "ψ is satisfied"
   d ← d + 1
end
```

Fig. 3. An Algorithm for Checking ∀μ-Calculus Properties

if we consider ∀CTL properties, it can be shown that our algorithm requires $O(nd^\kappa)$ Boolean variables where κ is the maximal depth of nested temporal operators in the ∀CTL property.

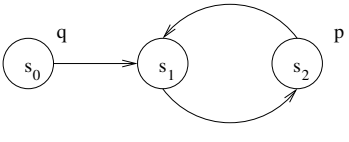

Fig. 4. A Simple Kripke Structure

As an example, consider the sample Kripke structure in Figure 4. The labels p and q denote $L(s_0) = \{q\}$, $L(s_2) = \{p\}$, but $L(s_1) = \emptyset$. Let Ψ stand for $\nu Y\{\}p \wedge \Box Y$. Suppose we wish to check whether $\neg q \vee (\mu X\{\}\Psi \vee \Box X)$ is satisfied by the Kripke structure. The corresponding Boolean formula for $\neg(\neg q \vee (\mu X\{\}\Psi \vee \Box X))$ is: (for detailed derivation, please see [23])

$$\begin{aligned}
&\Theta_K(\bar{u}, \neg(\neg q \vee (\mu X\{\}\Psi \vee \Box X)), 2)) \\
&= \chi_q(\bar{u}) \wedge (\neg\chi_p(\bar{u}) \vee (\chi_{\rightarrow}(\bar{u}, \bar{w}) \wedge (\bar{w} \neq \bar{u} \wedge c))) \wedge \\
&\quad (\chi_{\rightarrow}(\bar{u}, \bar{v}) \wedge (\bar{v} = \bar{u} \vee (c' \wedge (\chi_{\rightarrow}(\bar{v}, \bar{x}) \wedge ((\bar{x} \neq u \wedge \bar{x} \neq \bar{v}) \Leftrightarrow c'')))))
\end{aligned}$$

It is easy to see that there is no satisfying assignment for $\Omega_K(\bar{u}, \neg(\neg q \vee (\mu X\{\}\Psi \vee \Box X)), 2)$. By Theorem 2, there is no counterexample at unrolling depth 2. On the other hand, take the assignment ρ, where $[\![\bar{u}]\!]\rho = s_0$, $[\![\bar{v}]\!]\rho = s_1$, $[\![\bar{x}]\!]\rho = s_2$, and $[\![c]\!]\rho = [\![c']\!]\rho = [\![c'']\!]\rho = \mathsf{true}$. It is straightforward to verify that ρ is a satisfying assignment for $\Lambda_K(\bar{u}, \neg(\neg q \vee (\mu X\{\}\Psi \vee \Box X)), 2)$. Hence there may be a full proof of unrolling depth greater than 2 for $\neg(\neg q \vee (\mu X\{\}\Psi \vee \Box X))$ by Theorem 3.

5 Experimental Results

We are interested in the analysis of an n-process agreement protocol. Initially, process i has a random local bit v_i. All processes collect and distribute information with one another concurrently. At the end of the protocol, they will have the same value assigned to their local bits. In case of system failure, the faulty process stops updating its local bit nor exchanging information with others.

In addition to the local bit v_i, a program counter pc_i is used to indicate the current status (normal, failed, or decided) of process i. Firstly, we are interested in knowing whether all processes have agreed on their private bits when they all make their decisions. We therefore check that the following predicate is indeed invariant in the protocol:

$$good_n \stackrel{\triangle}{=} (\bigwedge_{i=1}^{n} pc_i = decided) \Rightarrow ((\bigwedge_{i=1}^{n} v_i) \vee (\bigwedge_{i=1}^{n} \neg v_i))$$

Secondly, we verify the following CTL property in the protocol:

$$up_n \stackrel{\triangle}{=} AG(v_1 \Rightarrow AF((\bigwedge_{i=1}^{n} pc_i = decided) \Rightarrow (\bigwedge_{i=1}^{n} v_i)))$$

The property up_n states that if the local bit of process 1 is true, then all computation paths will eventually make all local bits to be true when all processes decide. It is impossible to turn them back to be false in the protocol.

Thirdly, we verify that either all processes decide their local bits or some of them have failure almost surely along all computation. It can be specified by the following LTL formula:

$$ltl\_stable_n \stackrel{\triangle}{=} \Diamond(\Box((\bigwedge_{i=1}^{n} pc_i = decided) \vee (\bigvee_{i=1}^{n} pc_i = failed)))$$

In other words, no process can stay in a normal but undecided state forever. A weaker but similar property can be specified in Fair CTL. We now consider fair paths where no process is in the failed state infinitely often ($\Psi = \bigwedge_{i=1}^{n} F^{\infty}(pc_i \neq failed)$ in [10]). We would like to know whether all parties will decide their local bits eventually for all computation. In FCTL, we can specify the property as follows.

$$fctl\_stable_n \stackrel{\triangle}{=} A_{\Psi} F \bigwedge_{i=1}^{n} pc_i = decided$$

It is straightforward to rewrite the properties $good_n$, up_n, and $fctl\_stable_n$ in $\forall\mu$-calculus by standard encoding. For the LTL property ltl_stable_n, we apply the technique reported in [12, 24] and verify the existence of fair paths satisfying a $\forall\mu$-calculus formula. Observe that the completeness criteria for these properties are uniformly obtained by our framework. Once the property is rewritten as a $\forall\mu$-calculus formula, our proof-theoretic technique is able to verify it by any SAT solver.

Figure 5 compares the performance of our algorithm with the conventional BDD-based μ-calculus model checking algorithm. In our experiments, we use the CUDD package (release 2.4.0) with the sifting algorithm to implement the BDD-based algorithm. The zchaff SAT solver (release November 15th, 2004) is used as our SAT solver. All experiments were conducted on a Linux workstation (Pentium 4 2.8GHz with 2 GB memory).

Our experiments show that BDD-based algorithms perform consistently for different properties. If the BDD model representation can be built, these four properties can be verified with similar cost. On the other hand, the performance of SAT-based algorithm differs significantly in these properties. This is due to the fact that our algorithm requires different number of variables for these properties. It therefore does not perform so uniformly for various properties.

	$good_n$		up_n		ltl_stable_n		$fctl\_stable_n$	
n	BDD	SAT	BDD	SAT	BDD	SAT	BDD	SAT
3	0.13	0.77	0.19	0.99	0.2	2.33	0.11	7.36
4	1.2	1.44	1.06	4.09	2.80	10.76	1.36	31.00
5	11.91	3.86	10.55	9.45	17.21	29.12	12.95	112.82
6	17.44[2]	9.43	15.09[2]	34.94	15.19[2]	61.12	9.69[2]	232.63
7	timeout	18.23	timeout	75.86	timeout	157.21	timeout	553.02

(verification time in seconds)

Fig. 5. Experimental Results

[2] CUDD runs out of time with the sifting algorithm. The data is obtained without dynamic variable ordering.

For the invariant property $good_n$, our SAT-based algorithm is better than BDD-based algorithm for $n \geq 5$. For branching-time properties (up_n, ltl_stable_n, and $fctl\_stable_n$), the BDD-based algorithm cannot finish in 10 minutes for $n = 7$. With our algorithm, we are able to verify all these properties within 10 minutes. Surprisingly, our SAT-based algorithm performs better than BDD-based algorithm for some branching-time properties in this experiment.

6 Conclusion and Future Work

A complete SAT-based $\forall\mu$-calculus model checking algorithm is presented in the paper. Unlike previous works on proving branching-time temporal logics, our algorithm does not depend on completeness thresholds. Instead, it determines the completeness of proofs on the fly. The novelty of the new algorithm is that it combines both local and bounded model checking, and essentially reduces proof search in local model checking to Boolean satisfiability.

Our technique uses a proof-theoretic approach to develop completeness criteria. We feel our technique may give new insights into devising complete SAT-based model checking algorithms. Currently, it is unclear whether induction or interpolation can be applied in our framework. It would be interesting to have proof-theoretic interpretations of these heuristics as well.

Our experimental results suggest that our algorithm may perform better than a typical BDD-based model checker in some cases. In the future, we would like to conduct more experiments to support our preliminary findings.

Acknowledgments. The author would like to thank anonymous reviewers for their constructive comments and suggestions in improving the paper.

References

1. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In Cleaveland, W.R., ed.: Tools and Algorithms for the Construction and Analysis of Systems. Volume 1579 of LNCS., Springer-Verlag (1999) 193–207
2. Biere, A., Cimatti, A., Clarke, E.M., Fujita, M., Zhu, Y.: Symbolic model checking using SAT procedures instead of BDDs. In: Proceedings of the 36th Design Automation Conference (DAC' 99), New York, ACM Press (1999) 317–320
3. Emerson, E., Clarke, E.: Using branching-time temporal logic to synthesize synchronization skeletons. Science of Computer Programming **2** (1982) 241–266
4. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press, Cambridge, Massachusetts (1999)
5. Clarke, E., Kroening, D., Ouaknine, J., Strichman, O.: Completeness and complexity of bounded model checking. In Steffen, B., Levi, G., eds.: Verification, Model Checking, and Abstract Interpretation. Volume 2937 of LNCS., Springer-Verlag (2004) 85–96
6. Sheeran, M., Singh, S., Stålmarck, G.: Checking safety properties using induction and a SAT-solver. In Jr., W.A.H., Johnson, S.D., eds.: Formal Methods in Computer-Aided Design. Volume 1954 of LNCS., Springer-Verlag (2000) 108–125

7. Leonardo de Moura, H.R., Sorea, M.: Bounded model checking and induction: From refutation to verification. In Jr., W.A.H., Somenzi, F., eds.: Computer Aided Verification. Volume 2725 of LNCS., Springer Verlag (2003) 14–26
8. McMillan, K.L.: Interpolation and sat-based model checking. In Jr., W.A.H., Somenzi, F., eds.: Computer-Aided Verification. Volume 2725 of LNCS., Springer Verlag (2003) 1–13
9. Awedh, M., Somenzi, F.: Proving more properties with bounded model checking. In Alur, R., Peled, D.A., eds.: Computer Aided Verification. Volume 3114 of LNCS., Springer Verlag (2004) 96–108
10. Emerson, E.A., Lei, C.L.: Efficient model-checking in fragments of the propositional mu-calculus. In: Proceedings First Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press (1986) 267–278
11. Vardi, M., Wolper, P.: An automata-theoretic approach to automatic program verification. In: Proceedings First Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press (1986) 332–344
12. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, L.J.: Symbolic model checking: 10^{20} states and beyond. Information and Computation **98** (1992) 142–170
13. Cleaveland, R.: Tableau-based model checking in the propositional mu-calculus. Acta Informatica **27** (1989) 725–747
14. Stirling, C., Walker, D.: Local model checking in the modal mu-calculus. Theoretical Computer Science **89** (1991) 161–177
15. Andersen, H.R., Stirling, C., Winskel, G.: A compositional proof system for the modal μ-calculus. In: Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science, Paris, France, IEEE Computer Society Press (1994) 144–153
16. Emerson, E., Lei, C.: Modalities for model-checking: Branching time logic strikes back. In: Proceedings of the 12th ACM Symposium on Principles of Programming Languages, ACM Press (1985) 84–96
17. Schuppan, V., Biere, A.: Efficient reduction of finite state model checking to reachability analysis. Software Tools for Technology Transfer **5** (2004) 185–204
18. Schuele, T., Schneider, K.: Global vs. local model checking: A comparison of verification techniques for infinite state systems. In: International Conference on Software Engineering and Formal Methods (SEFM), Beijing, IEEE Computer Society Press (2004)
19. Schuele, T., Schneider, K.: Bounded local model checking. private communication (2005)
20. Wang, B.Y.: Unbounded model checking with sat - a local model checking approach. unpublished manuscript (2004)
21. Kozen, D.: Results on the propositional μ-calculus. Theoretical Computer Science **27** (1983) 333–354
22. Winskel, G.: A note on model checking the modal nu-calculus. Theoretical Computer Science **83** (1991) 157–167
23. Wang, B.Y.: Proving $\forall\mu$-calculus properties with sat-based model checking. Technical Report TR-IIS-05-003, Institute of Information Science, Academia Sinica (2005) http://www.iis.sinica.edu.tw/LIB/TechReport/tr2005/tr05003.pdf.
24. Clarke, E., Grumberg, O., Hamaguchi, K.: Another look at LTL model checking. In Dill, D.L., ed.: Computer Aided Verification. Volume 818 of LNCS., Springer-Verlag (1994) 415–428