Model Checking with SAT-Based Characterization of ACTL Formulas [⋆]

Wenhui Zhang
Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
P.O.Box 8718, Beijing 100080, China
zwh@ios.ac.cn

Abstract. Bounded semantics of LTL with existential interpretation and that of ECTL (the existential fragment of CTL), and the characterization of these existentially interpreted properties have been studied and used as the theoretical basis for SAT-based bounded model checking [2, 18]. This has led to a lot of successful work with respect to error detection in the checking of LTL and ACTL (the universal fragment of CTL) properties by satisfiability testing. Bounded semantics of LTL with the universal interpretation and that of ACTL, and the characterization of such properties by propositional formulas have not been successfully established and this hinders practical verification of valid universal properties by satisfiability checking. This paper studies this problem and the contribution is a bounded semantics for ACTL and a characterization of ACTL properties by propositional formulas. Firstly, we provide a simple bounded semantics for ACTL without considering the practical aspect of the semantics, based on converting a Kripke model to a model (called a k-model) in which the transition relation is captured by a set of k-paths (each path with k transitions). This bounded semantics is not practically useful for the evaluation of a formula, since it involves too many paths in the k-model. Then the technique is to divide the k-model into submodels with a limited number of k-paths (which depends on k and the ACTL property to be verified) such that if an ACTL property is true in every such model, then it is true in the k-model as well. This characterization can then be used as the basis for practical verification of valid ACTL properties by satisfiability checking. A simple case study is provided to show the use of this approach for both verification and error detection of an abstract two-process program written as a first order transition system.

1 Introduction

Bounded semantics of LTL with existential interpretation (called existential LTL hereafter) and that of ECTL (the existential fragment of CTL), and the characterization of these existentially interpreted properties have been studied and used as the theoretical basis for SAT-based bounded model checking [2, 18]. This has lead to a lot of successful work with respect to error detection in the checking of LTL and ACTL (the

[⋆] Supported by the National Natural Science Foundation of China under Grant No. 60573012 and 60421001, and the National Grand Fundamental Research 973 Program of China under Grant No. 2002cb312200.

universal fragment of CTL) properties by satisfiability testing [1]. It is considered as a complementary technique to BDD-based model checking [3, 5] for combating the state explosion problem, esp. for effective error detection [19].

Bounded semantics of LTL with the universal interpretation and that of ACTL, and the characterization of such properties by propositional formulas have not been successfully established and this hinders practical verification of valid universal properties by satisfiability checking. Bounded semantics of existential LTL and that of ECTL, and the characterization of such properties are consistent with the fact that the witness of the properties can be searched within a fragment of the valid paths. For witness of existential LTL properties φ, one path of the form $u \cdot v^{\omega}$ is sufficient. In some cases, a finite path u may be sufficient, while in the general case, it is sufficient to find a finite path $u \cdot v$ such that there is a transition from the last element of v to the first element of v, and show that $u \cdot v^{\omega} \models \varphi$ holds. For an ECTL property, a witness may consist of several paths. Some may need to have a loop, while others may not need to have a loop. For simplicity, we assume that all these paths are of the same length. A path of length $k+1$ is called a k-path (a path with k transitions). Then the number of k-paths needed for witnessing an ECTL formula depends on the number k and the structure of the formula. For instance, for $EGEF\varphi$ with φ being a propositional formula, the number of k-paths needed is $k+1$.

The problem of characterization of universally interpreted properties lies in that it looks difficult to reason about all involved paths of a model, since the number of such paths is too big. This paper studies this problem and the contribution is a bounded semantics for ACTL and a characterization of ACTL properties by propositional formulas. Firstly, we provide a simple bounded semantics for ACTL without considering the practical aspect of the semantics, based on converting a Kripke model to a model (called a k-model) in which the transition relation is captured by a set of k-paths (each path with k transitions). Although this bounded semantics is not practically useful for the evaluation of a formula, it serves as the basis for further development. Then the technique is to divide the k-model into submodels with a limited number of k-paths (which also depends on k and the ACTL property to be verified) such that if an ACTL property is true in every such model, then it is true in the k-model as well. This characterization can then be used as the basis for practical verification of valid ACTL properties by satisfiability checking. A simple case study is provided to show the use of this approach for both verification and error detection of an abstract two-process program written as a first order transition system.

The contents of this papers is as follows. Section 2 presents background knowledge on CTL. Section 3 presents a bounded semantics for ACTL. Section 4 presents a further development of the bounded semantics of ACTL. Section 5 presents a characterization of the problem for model checking ACTL properties by propositional formulas. Section 6 presents the verification approach and a case study to show the use of this characterization for verification of abstract programs with respect to ACTL properties and for error detection of incorrect abstract programs. Section 7 proposes a combined verification approach in light of a discussion with respect to related work. Finally, we present concluding remarks in Section 8.

2 Computation Tree Logic (CTL)

Computation tree logic is a propositional branching-time temporal logic [10] introduced by Emerson and Clarke as a specification language for finite state systems. In this section, preliminary knowledge on CTL, including the syntax and the semantics of CTL and the definition of ACTL and ECTL, is presented.

Let AP be a set of proposition symbols. The set of CTL formulas is defined as follows:

– Every member of AP is a CTL formula.
– The logical connectives of CTL are: $\neg$, $\wedge$, and $\vee$.
  If φ and ψ are CTL formulas, then so are $\neg\varphi$, $\varphi \wedge \psi$, and $\varphi \vee \psi$.
– The temporal operators are
  EX, ER, EU, AX, AR, and AU.
  If φ and ψ are CTL formulas, then so are:
  $EX\ \varphi$, $E(\varphi\ R\ \psi)$, $E(\varphi\ U\ \psi)$, $AX\ \varphi$, $A(\varphi\ R\ \psi)$, and $A(\varphi\ U\ \psi)$.

In addition to the logical connectives, $\varphi \rightarrow \psi$ may be used as an abbreviation of $\neg\varphi \vee \psi$. In addition to the temporal operators, $AF\varphi, AG\varphi, EF\varphi, EG\varphi$ may be used as abbreviations of respectively $A(true\, U\, \varphi)$, $A(false\, R\, \varphi)$, $E(true\, U\, \varphi)$, and $E(false\, R\, \varphi)$.

A model for CTL formulas is a Kripke structure $M = \langle S, T, I, L \rangle$ where S is a set of states, $T \subseteq S \times S$ is a transition relation which is total, $I \subseteq S$ is a set of initial states and $L : S \rightarrow 2^{AP}$ is a labeling function that maps each state of S to a set of propositions that are assumed to be true at that state. A sequence $\pi = \pi_0\pi_1 \cdots$ of S is a path of M, if $T(\pi_i, \pi_{i+1})$ holds for all $i \geq 0$.

Definition 1. *Let M be a model, s a state, p a proposition symbol, φ and ψ CTL formulas. $M, s \models \varphi$ denotes that φ is true at the state s in M. Let π be a path of M. The relation $\models$ is defined as follows:*

$M, s \models p$ *iff*	$p \in L(s)$.
$M, s \models \neg\varphi$ *iff*	$M, s \not\models \varphi$
$M, s \models \varphi \wedge \psi$ *iff*	$(M, s \models \varphi)$ *and* $(M, s \models \psi)$
$M, s \models \varphi \vee \psi$ *iff*	$(M, s \models \varphi)$ *or* $(M, s \models \psi)$
$M, s \models EX\varphi$ *iff*	$\exists\pi.(\pi_0 = s \wedge M, \pi_1 \models \varphi)$
$M, s \models E(\varphi U \psi)$ *iff*	$\exists\pi.(\pi_0 = s \wedge \exists k \geq 0.(M, \pi_k \models \psi \wedge \forall j < k.(M, \pi_j \models \varphi)))$
$M, s \models E(\varphi R \psi)$ *iff*	$\exists\pi.(\pi_0 = s \wedge (\forall j \geq 0.(M, \pi_j \models \psi) \vee$ $\exists k \geq 0.((M, \pi_k \models \varphi) \wedge \forall j \leq k.(M, \pi_j \models \psi))))$

$M, s \models AX\varphi$ *iff* $\forall\pi.(\pi_0 = s \rightarrow M, \pi_1 \models \varphi)$
$M, s \models A(\varphi U \psi)$ *iff* $\forall\pi.(\pi_0 = s \rightarrow \exists k \geq 0.(M, \pi_k \models \psi \wedge \forall j < k.(M, \pi_j \models \varphi)))$
$M, s \models A(\varphi R \psi)$ *iff* $\forall\pi.(\pi_0 = s \rightarrow (\forall j \geq 0.(M, \pi_j \models \psi) \vee$ $\exists k \geq 0.((M, \pi_k \models \varphi) \wedge \forall j \leq k.(M, \pi_j \models \psi))))$

A CTL formula is in negation normal form (NNF), if the symbol $\neg$ is applied only to proposition symbols. Every formula can be transformed into an equivalent formula in NNF.

The sublogic ACTL is the subset of CTL formulas that can be transformed into NNF formulas not containing the temporal operators EX, EF, EG, EU, ER. Dually, the sublogic ECTL is the subset of CTL formulas that can be transformed into NNF formulas not containing the temporal operators AX, AF, AG, AU, AR.

Definition 2. *Let φ be an ACTL formula. φ is true in M, denoted $M \models \varphi$, iff φ is true at all initial states of M.*

3 Bounded Semantics of ACTL

Since every ACTL formula can be transformed into an equivalent formula in NNF, we only consider formulas of the form $\varphi \vee \psi$, $\varphi \wedge \psi$, $AX\varphi$, $A(\varphi R \psi)$, $A(\varphi U \psi)$ constructed from propositions and the negation of propositions. Therefore, in the following, a formula refers to such an ACTL formula unless otherwise stated. In this section, a bounded semantics of ACTL is presented. One of the particular aspects of this semantics is the use of the condition $eqs(\pi)$ for stating that there are same (or equal) states appearing in different positions in the path π. Note that in the semantics of existential LTL and that of ECTL, a condition indicating that there is a transition from the last state of π to some state already in π is used [2, 18]. This condition is not useful for the construction of the bounded semantics for ACTL. The reason is that we need the property of $eqs(\pi)$ stated below for reasoning about the correctness of Lemma 1 and Lemma 2. Details are explained in the sequel.

For simplicity, we fix the model under consideration to be $M = \langle S, T, I, L \rangle$, and in the sequel, M refers to this model, unless otherwise stated. Let $k \geq 0$. A k-paths of M is a path $\pi = \pi_0 \cdots \pi_k$ of M where $\pi_i \in S$ for $i = 0, ..., k$ and $(\pi_i, \pi_{i+1}) \in T$ for $i = 0, ..., k-1$. The k-model for M is a structure $M_k = \langle S, Ph_k, I, L \rangle$ where Ph_k is the set of all different k-paths of M. Let $|\pi|$ be the length of π. We have the following definition of $eqs(\pi)$.

$$eqs(\pi) := \bigvee_{i=0}^{|\pi|-1} \bigvee_{j=i+1}^{|\pi|-1} \pi_i = \pi_j.$$

If π is a prefix of π', then $eqs(\pi) \rightarrow eqs(\pi')$.

Definition 3 (Bounded Semantics of ACTL). *Let M_k be the k-model of M, s a state, p a proposition symbol, φ and ψ ACTL formulas. $M_k, s \models_k \varphi$ denotes that φ is true at* *the state s in M_k. Let $\pi = \pi_0 \cdots \pi_k$ be a path in Ph_k. Let $[n]$ denote the set $\{0, ..., n\}$. The relation $\models_k$ is defined as follows:*

$M_k, s \models_k p$ *iff* $p \in L(s)$
$M_k, s \models_k \neg p$ *iff* $p \notin L(s)$
$M_k, s \models_k \varphi \wedge \psi$ *iff* $(M_k, s \models_k \varphi)$ *and* $(M_k, s \models_k \psi)$
$M_k, s \models_k \varphi \vee \psi$ *iff* $(M_k, s \models_k \varphi)$ *or* $(M_k, s \models_k \psi)$.
$M_k, s \models_k AX\varphi$ *iff* $k \geq 1$ *and* $\forall \pi.(\pi_0 = s \rightarrow M_k, \pi_1 \models_k \varphi)$
$M_k, s \models_k A(\varphi U \psi)$ *iff* $\forall \pi.(\pi_0 = s \rightarrow \exists i \in [k].(M_k, \pi_i \models_k \psi \wedge \forall j \in [i-1].(M_k, \pi_j \models_k \varphi)))$
$M_k, s \models_k A(\varphi R \psi)$ *iff* $\forall \pi.(\pi_0 = s \rightarrow ((eqs(\pi) \wedge \forall j \in [k].(M_k, \pi_j \models_k \psi)) \vee$ $\exists i \in [k].((M_k, \pi_i \models_k \varphi) \wedge \forall j \in [i].(M_k, \pi_j \models_k \psi))))$

For the soundness of this definition, we need to know that if $M, s \models \varphi$ then there is a finite $k \geq 0$ such that $M_k, s \models_k \varphi$, and vice versa.

Let k_M be the number of reachable states of M. For $k' = k_M$, since $k' \geq 1$ and $eqs(\pi)$ are satisfied for every $\pi \in Ph_{k'}$, we have $M, s \models \varphi$ implies $M_{k'}, s \models_{k'} \varphi$ by restricting every path in M to be k'-path. On the other hand, an infinite path π of M has a k'-path as its prefix. A property is true on π, if it is true on such a prefix, unless it is a global property, i.e., a property of the form $A(\varphi R \psi)$ such that φ does not hold in any state of π and ψ must hold in all states of π, and therefore a prefix is not sufficient for showing the truth of $\varphi R \psi$. Assume this situation occurs and $A(\varphi R \psi)$ holds in the bounded semantics. We want to show that $\varphi R \psi$ also holds on such a path π. For the first, the situation implies that ψ is true on every state of every k'-path of which the set of states is a subset of that of π. For the second, the set of states of all these k'-paths with the start state π_0 covers the set of states of π. These two conditions guarantee that ψ is true on every state of π and therefore $\varphi R \psi$ holds on π. Summing up the above discussion, we have $M_{k'}, s \models_{k'} \varphi$ implies $M, s \models \varphi$. On the other hand, paths in a k-model with more than k_M transitions can be shortened to paths with k_M transitions[1] without affecting the satisfiability of ACTL formulas in the model. Formally, we have

Lemma 1. *Let $k \geq k_M$. $M, s \models \varphi$ iff $M_k, s \models_k \varphi$.*

This assures in some sense the soundness of the semantics, in addition, we need to have some kind of continuity of the truth values of $M_k, s \models_k \varphi$ for a sequence of values of k. Fortunately, the property of $eqs(\pi)$ allows us to prove that if an ACTL property holds on the k-model, it also holds on the $(k+1)$-model (which is a model with longer paths). Formally:

[1] The number of reachable states for k_M is an over-approximation. A smaller number is usually sufficient. The least such number is called the completeness threshold, Computation of such numbers has been studied in e.g. [13].

Lemma 2. *If $M_k, s \models_k \varphi$, then $M_{k+1}, s \models_{k+1} \varphi$.*

This means that if $M_k, s \models_k \varphi$ holds for $k = k_M$, then there is a $k' \leq k$ such that for all $k'' \geq k'$, $M_{k''}, s \models_{k''} \varphi$ holds, and for all $k'' < k'$, $M_{k''}, s \models_{k''} \varphi$ does not hold. Combining Lemma 1 and Lemma 2, we obtain

Theorem 1 (Soundness). *$M, s \models \varphi$ iff there is some $k \leq k_M$ such that $M_k, s \models_k \varphi$.*

Definition 4. *Let φ be an ACTL formula. φ is true in the k-model M_k, denoted $M_k \models_k \varphi$, iff φ is true at all initial states of the model M_k.*

Following Theorem 1 and Lemma 2, we have the following theorem.

Theorem 2. *$M \models \varphi$ iff there is some $k \leq k_M$ such that $M_k \models_k \varphi$ holds.*

4 Refining the Bounded Semantics

The bounded semantics of ACTL is not directly useful as a method for checking whether an ACTL formula holds in the model, since the number of k-paths in the k-model is large. An over-approximation of the number is $(k_M)^{k+1}$, while the exact number is difficult to compute. In the case of ECTL, if a witness exists, we only need to find a small subset (depends on k and the property to be verified) of k-paths in the k-model to certify the existence of a witness. in the case of ACTL, the number of involved k-paths for certification of the property is necessarily large. The technique is then to divide the k-model into submodels with a limited number of paths (which also depends on k and the property to be verified) and prove that if such an ACTL property is true in every such model, then it is true in the k-model as well. The details are explained in the sequel. We first define the concept of submodels.

Definition 5 (Submodels). *Let $M_k = \langle S, Ph_k, I, L\rangle$ be the k-model of M. $M_k^{'} = \langle S, Ph_k^{'}, I, L\rangle$ is a submodel of M_k, if $Ph_k^{'} \subseteq Ph_k$. We write $M_k^{'} \leq M_k$ for this relation.*

Similarly, if $M_k^{'}$ and $M_k^{''}$ are two submodels, $M_k^{'} \leq M_k^{''}$ iff $Ph_k^{'} \subseteq Ph_k^{''}$. The number of k-paths in a submodel $M_k^{'}$ is denoted by $|M_k^{'}|$. We call a submodel $M_k^{'}$ with n k-paths a (k, n)-submodel. Note that in a (k, n)-submodel, we do not require the n k-paths in the submodel be different.

A state s in a submodel $M_k^{'}$ satisfies a formula φ, denoted by $M_k^{'}, s \models_k \varphi$, is defined just like the definition of $M_k, s \models_k \varphi$ (cf. Definition 3), except that $\forall\pi$ means $\forall\pi \in Ph_k^{'}$ instead of $\forall\pi \in Ph_k$. We have the following property of submodels.

Proposition 1. *If $M_k^{'} \leq M_k^{''}$, then $M_k^{''}, s \models_k \varphi$ implies $M_k^{'}, s \models_k \varphi$.*

Based on this proposition, we obtain:

Proposition 2. *Let $M_k^{'}, M_k^{''}$ be respectively a (k, n)-submodel and a (k, m)-submodel. If $M_k^{'}, s_1 \not\models_k \varphi$ or $M_k^{''}, s_2 \not\models_k \psi$, then there is a $(k, max(m, n))$-submodel $M_k^{'''}$ such that $M_k^{'''}, s_1 \not\models_k \varphi$ or $M_k^{'''}, s_2 \not\models_k \psi$.*

We may combine submodels. Let $M_k^{'}, M_k^{''}$ be two submodels. Then $M_k^{'} \cup M_k^{''}$ is the submodel M_k^* with $Ph_k^* = Ph_k' \cup Ph_k''$ and the other components remain the same.

Proposition 3. *Let $M_k^{'}, M_k^{''}$ be two submodels. If $M_k^{'}, s_1 \not\models_k \varphi$ and $M_k^{''}, s_2 \not\models_k \psi$, then $M_k^{'} \cup M_k^{''}, s_1 \not\models_k \varphi$ and $M_k^{'} \cup M_k^{''}, s_2 \not\models_k \psi$.*

A consequence of this proposition is that if there is a (k, n)-submodel $M_k^{'}$ and a (k, m)-submodel $M_k^{''}$ such that $M_k^{'}, s_1 \not\models_k \varphi$ and $M_k^{''}, s_2 \not\models_k \psi$, then there is a $(k, m+n)$-submodel $M_k^{'''}$ such that $M_k^{'''}, s_1 \not\models_k \varphi$ and $M_k^{'''}, s_2 \not\models_k \psi$.

In the following, we analyze how many paths are needed in submodels such that we can conclude if an ACTL property is true in every such submodel of the k-model, then it is true in the k-model. Let φ be a propositional formula.

> For every s, if for every $(k, 0)$-submodel $M_k^{'}$ (there is actually only one $(k, 0)$-submodel), $M_k^{'}, s \models_k \varphi$ holds, then $M_k, s \models_k \varphi$.

This is because propositional property does not depend on k-paths[2]. This fact serves as the basis for reasoning about composed formulas. Suppose that we have now the following two assumptions (which are needed for the following inductive construction):

1. For every s, if for every (k, n)-submodel $M_k^{'}$, $M_k^{'}, s \models_k \varphi$ holds, then $M_k, s \models_k \varphi$.
2. For every s, if for every (k, m)-submodel $M_k^{''}$, $M_k^{''}, s \models_k \psi$ holds, then $M_k, s \models_k \psi$.

We then consider the composed ACTL formulas. Let $z = max(m, n)$. According to Proposition 1 and the two assumptions, we have, for every s,

– if for every (k, z)-submodel $M_k^{'''}$, $M_k^{'''}, s \models_k \varphi$ holds, then $M_k, s \models_k \varphi$.
– if for every (k, z)-submodel $M_k^{'''}$, $M_k^{'''}, s \models_k \psi$ holds, then $M_k, s \models_k \psi$.

Combining these two statements, we obtain:

> For every s, if for every $(k, max(m, n))$-submodel M_k^*, $M_k^*, s \models_k \varphi \wedge \psi$ holds, then $M_k, s \models_k \varphi \wedge \psi$.

For disjunction, we consider the validity of $M_k, s \models_k \varphi \vee \psi$. Suppose that $M_k, s \models_k \varphi \vee \psi$ does not hold. Then none of $M_k, s \models_k \varphi$ and $M_k, s \models_k \psi$ holds. According to assumption 1 and assumption 2, there is a (k, n)-submodel $M_k^{'}$ such that $M_k^{'}, s \not\models_k \varphi$ and there is a (k, m)-submodel $M_k^{''}$, such that $M_k^{''}, s \not\models_k \psi$. Combining these two submodels, we obtain a $(k, m+n)$-submodel $M_k^{'''}$ such that $M_k^{'''}, s \not\models_k \varphi$ and $M_k^{'''}, s \not\models_k \psi$. Then $M_k^{'''}, s \not\models_k \varphi \vee \psi$. By turning the direction of reasoning, we obtain:

> For every s, if for every $(k, m+n)$-submodel M_k^*, $M_k^*, s \models_k \varphi \vee \psi$ holds, then $M_k, s \models_k \varphi \vee \psi$.

[2] The number of different (k, n)-submodels is limited by m^n where m is the number of different k-paths.

For temporal formulas of the form $AX\varphi$, suppose that $M_k, s \models_k AX\varphi$ does not hold. Then there is a k-path $P_1 = \pi_0\pi_1 \cdots \pi_k$ with $\pi_0 = s$ such that $M_k, \pi_1 \models_k \varphi$ does not hold. According to assumption 1, there is a (k, n)-submodel $M_k^{'}$ such that $M_k^{'}, \pi_1 \not\models_k \varphi$. Extending $M_k^{'}$ with P_1, we obtain a $(k, n+1)$-submodel $M_k^{''}$ such that $M_k^{''}, s \not\models_k AX\varphi$. By turning the direction of reasoning, we obtain

For every s, if for every $(k, n+1)$-submodel M_k^*, $M_k^*, s \models_k AX\varphi$ holds, then $M_k, s \models_k AX\varphi$.

For temporal formulas of the form $A(\varphi U\psi)$, suppose that $M_k, s \models_k A(\varphi U\psi)$ does not hold. Then there is a k-path $P_1 = \pi_0\pi_1 \cdots \pi_k$ with $\pi_0 = s$ such that either (1) $M_k, \pi_i \models_k \psi$ does not hold for all $0 \leq i \leq k$ or

- (2a) $M_k, \pi_0 \models_k \psi$ does not hold, and
- (2b) for each $j < k$, if $M_k, \pi_i \models_k \varphi$ holds for all $0 \leq i \leq j$, then $M_k, \pi_{j+1} \models_k \psi$ does not hold.

According to assumption 1 and assumption 2,

- With condition (1), there is a (k, m)-submodel M_k^i such that $M_k^i, \pi_i \not\models_k \psi$ for each $0 \leq i \leq k$.
  According to Proposition 3, we may combine the $k+1$ submodels, and obtain that there is a $(k, (k+1)m)$-submodel M_k^* such that $M_k^*, \pi_i \not\models_k \psi$ for each $0 \leq i \leq k$.
- With condition (2a), there is a (k, m)-submodel M_k^0 such that $M_k^0, \pi_0 \not\models_k \psi$.
- With condition (2b), for each $j < k$, there is a (k, n)-submodel $M_k^{'i}$ such that $M_k^{'i}, \pi_i \not\models_k \varphi$ for some $0 \leq i \leq j$, or there is a (k, m)-submodel M_k^{j+1} such that $M_k^{j+1}, \pi_{j+1} \not\models_k \psi$.
  According to Proposition 2, we obtain that for each $j < k$, there is a $(k, max(m, n))$-submodel $M_k^{''}$ such that $M_k^{''}, \pi_i \not\models_k \varphi$ for some $0 \leq i \leq j$, or $M_k^{''}, \pi_{j+1} \not\models_k \psi$.
  According to Proposition 3, we obtain that there is a $(k, k \cdot max(m, n))$-submodel $M_k^{'*}$ such that for each $j < k$, $M_k^{'*}, \pi_i \not\models_k \varphi$ for some $0 \leq i \leq j$, or $M_k^{'*}, \pi_{j+1} \not\models_k \psi$.

Since condition (2a) and condition (2b) are to be satisfied at the same time, we need a $(k, k \cdot max(m, n) + m)$-submodel to cover condition (2). Since condition (1) is an alternative to condition (2), and $(k+1) \cdot m \leq (k, k \cdot max(m, n) + m)$, a $(k, k \cdot max(m, n) + m)$-submodel is sufficient to cover both conditions. Take the path P_1 into consideration, we have a $(k, k \cdot max(m, n) + m + 1)$-submodel M_k^{**} such that $M_k^{**}, s \models_k A(\varphi U\psi)$ does not hold. By turning the direction of reasoning, we obtain

For every s, if for every $(k, k \cdot max(m, n) + m + 1)$-submodel M_k^*, $M_k^*, s \models_k A(\varphi U\psi)$ holds, then $M_k, s \models_k A(\varphi U\psi)$.

Similar arguments can be applied to temporal formulas of the form $A(\varphi R\psi)$. Because the semantics of $A(\varphi R\psi)$ involves the condition $eqs(\pi)$, an analysis of $eqs(\pi)$ is needed. Otherwise, the reasoning is similar to that of the case of $A(\varphi U\psi)$.

Suppose that $M_k, s \models_k A(\varphi R\psi)$ does not hold. Then there is a k-path $P_1 = \pi_0\pi_1 \cdots \pi_k$ with $\pi_0 = s$ such that

– (1) $eqs(\pi)$ does not hold or $M_k, \pi_j \models_k \psi$ does not hold for some $0 \leq j \leq k$, and
– (2) for each $j \leq k$, if $M_k, \pi_i \models_k \psi$ holds for all $0 \leq i \leq j$, then $M_k, \pi_j \models_k \varphi$ does not hold.

Condition (1) can be divided into 2 subcases: (1a) $eqs(\pi)$ does not hold and (1b) $eqs(\pi)$ holds. According to assumption 1 and assumption 2,

– With condition (1b), we have that $M_k, \pi_j \models_k \psi$ does not hold for some $0 \leq j \leq k$. Then there is a (k, m)-submodel $M_k^{'}$ such that $M_k^{'}, \pi_j \not\models_k \psi$ for some $0 \leq j \leq k$.
– With condition (2), for each $j \leq k$, there is a (k, m)-submodel $M_k^{'i}$ such that $M_k^{'i}, \pi_i \not\models_k \psi$ for some $0 \leq i \leq j$, or there is a (k, n)-submodel M_k^{j} such that $M_k^{j}, \pi_j \not\models_k \varphi$.
Then there is a $(k, (k+1) \cdot max(m, n))$-submodel $M_k^{''}$ such that for each $j \leq k$, $M_k^{''}, \pi_i \not\models_k \varphi$ for some $0 \leq i \leq j$, or $M_k^{''}, \pi_j \not\models_k \varphi$.

Applying the similar argument as that in the case of $A(\varphi U \psi)$, we obtain that in case $eqs(\pi)$ holds (i.e., we have condition (1b) and condition (2)), there is a $(k, (k+1) \cdot max(m, n) + m + 1)$-submodel M_k^* such that $M_k^*, s \not\models_k A(\varphi R \psi)$. In case $eqs(\pi)$ does not holds, there is a $(k, (k+1) \cdot max(m, n) + 1)$-submodel $M_k^{'''}$ such that $M_k^{'''}, s \not\models_k A(\varphi R \psi)$. According to Proposition 1, there is a $(k, (k+1) \cdot max(m, n) + m + 1)$-submodel M_k^* such that $M_k^*, s \not\models_k A(\varphi R \psi)$ also in the this case. Therefore, by turning the direction of reasoning, we obtain

For every s, if for every $(k, (k+1) \cdot max(m, n) + m + 1)$-submodel M_k^*, $M_k^*, s \models_k A(\varphi R \psi)$ holds, then $M_k, s \models_k A(\varphi R \psi)$.

The above reasoning leads to the following definition of the necessary number of paths in such submodels.

Definition 6. *Let φ be an ACTL formula. $n_k(\varphi)$ is defined as follows.*

$n_k(p)$	$= 0 \ if \ p \in AP$
$n_k(\neg p)$	$= 0 \ if \ p \in AP$
$n_k(\varphi \wedge \psi)$	$= max(n_k(\varphi), n_k(\psi))$
$n_k(\varphi \vee \psi)$	$= n_k(\varphi) + n_k(\psi)$
$n_k(AX\varphi)$	$= n_k(\varphi) + 1$
$n_k(A(\varphi R \psi))$	$= (k+1) \cdot max(n_k(\varphi), n_k(\psi)) + n_k(\psi) + 1$
$n_k(A(\varphi U \psi))$	$= k \cdot max(n_k(\varphi), n_k(\psi)) + n_k(\psi) + 1$

For verifying φ, we divide the k-model into submodels with $n_k(\varphi)$ paths. This leads to the following lemma.

Lemma 3. *$M_k, s \models_k \varphi$ iff for every $(k, n_k(\varphi))$-submodel M_k^*, $M_k^*, s \models_k \varphi$ holds.*

This lemma can be proved by structural induction on φ based on the above analysis. Combining Theorem 1, we obtain

Theorem 3. *$M, s \models \varphi$ iff there is some $k \leq k_M$ such that for every $(k, n_k(\varphi))$-submodel M_k^*, $M_k^*, s \models_k \varphi$ holds.*

Similar to Definition 4, we can define the relation $M_k^* \models_k \varphi$ for a submodel M_k^* and a formula φ. Then we obtain

Theorem 4. *$M \models \varphi$ iff there is some $k \leq k_M$ such that for every $(k, n_k(\varphi))$-submodel M_k^*, $M_k^* \models_k \varphi$ holds.*

5 SAT-Based Characterization of ACTL

Let $k \geq 0$. Let N_k be the number of different k-paths of M. Let $u_{i,0}, ..., u_{i,k}$ be a finite sequence of state variables for each $i \in \{1, ..., N_k\}$. The sequence $u_{i,0}, ..., u_{i,k}$ is intended to be used as a representation of a path of M_k.

Definition 7. *Let $k \geq 0$.*

$$P_k(i) := \bigwedge_{j=0}^{k-1} T(u_{i,j}, u_{i,j+1})$$

Every assignment to the set of state variables $\{u_{i,0}, ..., u_{i,k}\}$ satisfying $P_k(i)$ represents a valid k-path of M. The sequence $u_{i,0}, ..., u_{i,k}$ is then called a symbolic path of M_k. Let a be an assignment to $u_{i,0}, ..., u_{i,k}$ for $i \in \{1, ..., N_k\}$. Then the value assigned to $u_{i,j}$, denoted $a(u_{i,j})$, represents a state of M. Conversely, for each state $s \in S$ of M, we use $u(s)$ to represent that u has already been assigned a value representing the state s. The difference between a state variable u and $u(s)$ is that the latter has a fixed assignment and therefore cannot be assigned new values.

Definition 8 (Transition Relation). *Let $k \geq 0$. Let $0 \leq b \leq N_k$.*

$$[[M]]_k^b := \bigwedge_{i=1}^{b} P_k(i)$$

This is a collection of $P_k(l)$ for $l = 1, ..., b$. Let $p \in AP$ be a proposition symbol and $p(u)$ represent the propositional formula representing the states in which p is true according to L of M. State it differently, we have that $p(u)$ is true when u is assigned the truth value representing a state s such that p holds on s. Let $e_k(i)$ denote that there are same states appearing in different positions in path $P_k(i)$. Formally,

$$e_k(i) := \bigvee_{x=0}^{k-1} \bigvee_{y=x+1}^{k} u_{i,x} = u_{i,y}.$$

This definition corresponds to the definition of $eqs(\pi)$ for a k-path $\pi = u_{i,0}u_{i,1} \cdots u_{i,k}$.

Definition 9 (Translation of ACTL formulas). *Let $k \geq 0$. Let u be a state variable and φ be an ACTL formula. The encoding $[[\varphi, u]]_k^b$ is defined as follows.*

$$
\begin{array}{ll}
\hline
[[p, u]]_k^b & = p(u) \\
[[\neg p, u]]_k^b & = \neg p(u) \\
[[\varphi \vee \psi, u]]_k^b & = [[\varphi, u]]_k^b \vee [[\psi, u]]_k^b \\
[[\varphi \wedge \psi, u]]_k^b & = [[\varphi, u]]_k^b \wedge [[\psi, u]]_k^b \\
\hline
[[AX\varphi, u]]_k^b & = \bigwedge_{i=1}^{b}(u = u_{i,0} \rightarrow [[\varphi, u_{i,1}]]_k^b) \\
[[A(\varphi R \psi), u]]_k^b & = \bigwedge_{i=1}^{b}(u = u_{i,0} \rightarrow \\
 & \quad \bigvee_{j=0}^{k}([[\varphi, u_{i,j}]]_k^b \wedge \bigwedge_{t=0}^{j}[[\psi, u_{i,t}]]_k^b) \vee \bigwedge_{j=0}^{k}[[\psi, u_{i,j}]]_k^b \wedge e_k(i)) \\
[[A(\varphi U \psi), u]]_k^b & = \bigwedge_{i=1}^{b}(u = u_{i,0} \rightarrow \bigvee_{j=0}^{k}([[\psi, u_{i,j}]]_k^b \wedge \bigwedge_{t=0}^{j-1}[[\varphi, u_{i,t}]]_k^b)) \\
\hline
\end{array}
$$

where $[[\varphi, u_{i,j}]]_0^b$ denotes $false$ for $j > 0$.

$[[\varphi, u_{i,j}]]_0^b$ may occur when $k = 0$. We may only consider the cases with $k \geq 1$ in the definition. But we choose to allow $k = 0$ for avoiding situations where we may need to explicitly mention $k = 0$ as a special case.

Definition 10. $[[M, \varphi, u]]_k^b := [[M]]_k^b \rightarrow [[\varphi, u]]_k^b$.

$[[M, \varphi, u(s)]]_k^b$ encodes $M_k', s \models \varphi$, in the sense that a model of $[[M, \varphi, u(s)]]_k^b$ satisfying $[[M]]_k^b$ yields a (k, b)-submodel M_k' such that $M_k', s \models \varphi$. This means that if there is no falsifying assignments, then every (k, b)-submodel M_k' satisfies $M_k', s \models \varphi$. On the other hand, a falsifying assignment of $[[M, \varphi, u(s)]]_k^b$ yields a (k, b)-submodel M_k'' such that $M_k'', s \not\models \varphi$.

Lemma 4. *$[[M, \varphi, u(s)]]_k^b$ is valid iff for every (k, b)-submodel M_k', $M_k', s \models \varphi$.*

According to Lemma 3, we obtain $M_k, s \models \varphi$ iff $[[M, \varphi, u(s)]]_k^{n_k(\varphi)}$ is valid. Then with Theorem 1, we obtain

Theorem 5. *$M, s \models \varphi$ iff there is some $k \leq k_M$ such that $[[M, \varphi, u(s)]]_k^{n_k(\varphi)}$ is valid.*

Definition 11. $[[M, \varphi]]_k^b := I(u) \wedge [[M]]_k^b \rightarrow [[\varphi, u]]_k^b$.

$I(u)$ restricts the potential values of u to be the initial states of M. $[[M, \varphi]]_k^b$ is valid iff for each s of the initial states, $[[M]]_k^b \rightarrow [[\varphi, u(s)]]_k^b$ is valid. According to Lemma 4 and Lemma 3, we obtain $M_k \models \varphi$ iff $[[M, \varphi]]_k^{n_k(\varphi)}$ is valid. Then with Theorem 2, we obtain

Theorem 6. *$M \models \varphi$ iff there is some $k \leq k_M$ such that $[[M, \varphi]]_k^{n_k(\varphi)}$ is valid.*

6 Bounded Verification and Case Study

Bounded verification of valid ACTL properties can be based on theorem 6. For minimizing the number of propositions used in the SAT formulas, we base the verification on the following corollary where the variable u (implicitly) in Theorem 6 is replaced by $u_{1,0}$ which is already in $[[M]]_k^n$ (when $n \geq 1$). Let

$$[[M, \varphi]]_k^* := I(u_{1,0}) \wedge [[M]]_k^{n_k(\varphi)} \rightarrow [[\varphi, u_{1,0}]]_k^{n_k(\varphi)}.$$

Corollary 1. *$M \models \varphi$ iff there is a $0 \leq k \leq k_M$ such that $[[M, \varphi]]_k^*$ is valid.*

The verification approach is as follows. For a given model M and an ACTL formula φ,

– Start with $k = 0$;
– If $[[M, \varphi]]_k^*$ is valid, report that the property holds;
– Increase k, if $k \leq k_M$, go to the first "if"-test;
– Report that the property does not hold.

6.1 Case Study

We first present the tool for the case study. There are mainly two steps for the verification: one is the generation of a CNF formula and the other is the checking of the formula. The tool first converts the Boolean representation of the initial state and the transition relation of the abstract program and the property (to be checked) to the CNF formula and then call the satisfiability checker BOSCH[3] for checking the CNF formula. If the formula is satisfiable, an assignment that makes the formula satisfiable is presented. This can be used for error detection as demonstrated in Section 6.3.

We now present the abstract program (this is taken from [23] in which the program was used for the illustration of the verification of LTL properties) and the properties to be verified. Let a, b be variables of enumeration type which have respectively the domain $\{s_0, ..., s_3\}$ and $\{t_0, ..., t_3\}$. Let x, y, t be variables of Boolean type. The program consists of two processes: A and B with the following specification in a first order transition system [17]:

Process A:	
$a = s_0$	$\longrightarrow (y, t, a) := (1, 1, s_1)$
$a = s_1 \wedge (x = 0 \vee t = 0)$	$\longrightarrow (a) := (s_2)$
$a = s_2$	$\longrightarrow (y, a) := (0, s_3)$
$a = s_3$	$\longrightarrow (y, t, a) := (1, 1, s_1)$
Process B:	
$b = t_0$	$\longrightarrow (x, t, b) := (1, 0, t_1)$
$b = t_1 \wedge (y = 0 \vee t = 1)$	$\longrightarrow (b) := (t_2)$
$b = t_2$	$\longrightarrow (x, b) := (0, t_3)$
$b = t_3$	$\longrightarrow (x, t, b) := (1, 0, t_1)$

Let the initial state be $a = s_0 \wedge b = t_0 \wedge x = y = t = 0$. We consider two properties:

– A liveness property: process A or process B will at some future point (including the current one) pass a critical region, i.e. $AF(a = s_3 \vee b = s_3)$.
– A mixed property: at any point, process A or process B will at some future point (including the current one) pass a critical region, i.e. $AGAF(a = s_3 \vee b = s_3)$.

[3] A tool based on DPLL and developed based on parts of the code of a tool presented in [25].

Let boolean variables a_0 and a_1 represent the variable a such that $a_0 = i \wedge a_1 = j$ meaning $a = s_{2i+j}$, and b_0 and b_1 represent b such that $b_0 = i \wedge b_1 = j$ meaning $b = t_{2i+j}$. Then each state is represented by a tuple $(a_0, a_1, b_0, b_1, x, y, t)$.

Let $V = \{a_0, a_1, b_0, b_1, p, q, r\}$. The system can be represented by boolean formulas as follows:

$$
\begin{array}{l}
I(a_0, a_1, b_0, b_1, x, y, t) \\
\equiv x = 0 \wedge y = 0 \wedge t = 0 \wedge a_0 = 0 \wedge a_1 = 0 \wedge b_0 = 0 \wedge b_1 = 0 \\
T(a_0, a_1, b_0, b_1, x, y, t, a_0', a_1', b_0', b_1', x', y', t') \\
\equiv a_0 = 0 \wedge a_1 = 0 \wedge y' = 1 \wedge t' = 1 \wedge a_1' = 1 \wedge same(V \setminus \{y, t, a_1\}) \vee \\
\quad a_0 = 0 \wedge a_1 = 1 \wedge (x = 0 \vee t = 0) \wedge a_0' = 1 \wedge a_1' = 0 \wedge same(V \setminus \{a_0, a_1\}) \vee \\
\quad a_0 = 1 \wedge a_1 = 0 \wedge y' = 0 \wedge a_1' = 1 \wedge same(V \setminus \{y, a_1\}) \vee \\
\quad a_0 = 1 \wedge a_1 = 1 \wedge y' = 1 \wedge t' = 1 \wedge a_0' = 0 \wedge same(V \setminus \{y, t, a_0\}) \\
\quad b_0 = 0 \wedge b_1 = 0 \wedge x' = 1 \wedge t' = 0 \wedge b_1' = 1 \wedge same(V \setminus \{x, t, b_1\}) \vee \\
\quad b_0 = 0 \wedge b_1 = 1 \wedge (y = 0 \vee t = 1) \wedge b_0' = 1 \wedge b_1' = 0 \wedge same(V \setminus \{b_0, b_1\}) \vee \\
\quad b_0 = 1 \wedge b_1 = 0 \wedge x' = 0 \wedge b_1' = 1 \wedge same(V \setminus \{x, b_1\}) \vee \\
\quad b_0 = 1 \wedge b_1 = 1 \wedge x' = 1 \wedge t' = 0 \wedge b_0' = 0 \wedge same(V \setminus \{x, t, b_0\}) \vee loop\_action
\end{array}
$$

where $same(X)$ represents $v_1' = v_1 \wedge \cdots \wedge v_n' = v_n$ for the set of propositions $X = \{v_1, ..., v_n\}$, and loop_action is a transition enabled if none of the other transitions is applicable, and the effect of this transition is that the values of the state variables are kept unchanged in the next state.

The formula $(a = s_3 \vee b = t_3)$ is the same as the following.

$$(a_0 = 1 \wedge a_1 = 1 \vee b_0 = 1 \wedge b_1 = 1)$$

Let us denote this formula by ψ. Then the two properties are as follows.

$$
\begin{array}{l}
(1)\ M \models AF\psi \\
(2)\ M \models AGAF\psi
\end{array}
$$

6.2 Checking Correctness

For $M \models AF\psi$, we want to know whether $[[M, AF\psi]]_k^*$ is valid for some k. We check the satisfiability of the negation of $[[M, AF\psi]]_k^*$ for $k = 0, 1, 2, ...$, until the formula is unsatisfiable, or the completeness threshold k_M is reached. By making trivial simplifications, transforming the formula into CNF format, and using the tool BOSCH for satisfiability checking, we obtain that the CNF formula is satisfiable for $k = 0, 1, 2, 3$ and it is unsatisfiable for $k = 4$. This proves $M \models AF\psi$. Table 1 shows the experimental data of this verification on a Sun Blade 1000 with 750 MHz and 512 MB. The number of variables includes the number of variables representing the states and that of auxiliary variables used in the transformation of the formula into CNF. The time used by BOSCH for satisfiability checking is negligible.

For $M \models AGAF\psi$, we check the satisfiability of the negation of $[[M, AGAF\psi]]_k^*$ for $k = 0, 1, 2, ...$. We obtain that it is satisfiable for $k = 0, 1, ..., 9$ and it is unsatisfiable for $k = 10$. This proves $M \models AGAF\psi$. Table 2 shows the experimental data of this verification for $k = 0, 3, 6, 9, 10$. The time is that (in seconds) used by BOSCH for satisfiability checking.

Property	k	Variables	Clauses	Time	SAT
$AF\psi$	0	9	14	0.0	yes
	1	26	157	0.0	yes
	2	43	302	0.0	yes
	3	60	449	0.0	yes
	4	77	598	0.0	no

Table 1. Experimental data for verification of $AF\psi$

Property	k	Variables	Clauses	Time	SAT
$AGAF\psi$	0	23	25	0.0	yes
	3	185	1000	0.0	yes
	6	410	2146	0.1	yes
	9	698	3463	0.2	yes
	10	808	3940	5.0	no

Table 2. Experimental data for for verification of $AGAF\psi$

6.3 Error Detection

Suppose that we have an erroneous program where the transition rule

$$a = s_1 \wedge (x = 0 \vee t = 0) \rightarrow (a) := (s_2)$$

is wrongly written as

$$a = s_1 \wedge (x = 0 \vee t = 1) \rightarrow (a) := (s_2).$$

Then the two properties do not hold in this modified program. Let us denote this program (its equivalent Kripke structure) by M'. Then we need to check the properties up to the threshold $k_{M'}$. With Proposition 2, we may use an over approximation of $k_{M'}$. For instance, we may use 17, which is the number of reachable states of M', as the over approximation.

The inputs to the satisfiability checker are satisfiable with each k up to and including 17, and this certifies that the properties do not hold in this program, i.e.

$$M' \not\models AF\psi$$
$$M' \not\models AGAF\psi$$

Table 3 and Table 4 show the experimental data of error detection with respect to the properties $AF\psi$ and $AGAF\psi$ with $k = 0, 4, 8, 12, 16, 17$, respectively.

For error location, the path information produced by BOSCH can be used for the analysis of the problem of the program. For the property $AGAF\psi$, the path information is shown in Table 5, where $a = s_i$ iff $a_0 = i/2 \wedge a_1 = i\%2$ and $b = t_i$ iff $b_0 = i/2 \wedge b_1 = i\%2$. By looking at the path information, we find two paths. The first path has a loop at state 06 (the sixth state in the path) and the second path has a loop at state 02. The first path satisfies $AF\psi$ and the second one does not. Further, we can relate the first state of the second path to the first state of the first path, and this means that the second path starts at the first state of the first path, and this path has an execution sequence that looks like a deadlock. By analyzing the program, we know that there is a deadlock (not ending with a state satisfying ψ) and therefore the program does not satisfy $AGAF\psi$.

Property	k	Variables	Clauses	Time	SAT
$AF\psi$	0	9	14	0.0	yes
	4	77	598	0.0	yes
	8	145	1214	0.0	yes
	12	213	1862	0.0	yes
	16	281	2542	0.0	yes
	17	298	2717	0.0	yes

Table 3. Experimental data for error detection w. r. t. $AF\psi$

Property	k	Variables	Clauses	Time	SAT
$AGAF\psi$	0	23	25	0.0	yes
	4	253	1363	0.0	yes
	8	595	3005	0.2	yes
	12	1049	4951	0.4	yes
	16	1615	7201	0.9	yes
	17	1774	7811	1.1	yes

Table 4. Experimental data for error detection w. r. t. $AGAF\psi$

$Path$	$State$	a_0	a_1	b_0	b_1	x	y	t
1	00	0	0	0	0	0	0	0
1	01	0	1	0	0	0	1	1
1	02	1	0	0	0	0	1	1
1	03	1	1	0	0	0	0	1
1	04	1	1	0	1	1	0	0
1	05	1	1	1	0	1	0	0
1	06	0	1	1	0	1	1	1
1	07	1	0	1	0	1	1	1
1	08	1	1	1	0	1	0	1
1	$09 = 06$	0	1	1	0	1	1	1
1	$\cdots$	$\cdots\cdots\cdots\cdots\cdots\cdots\cdots$						
2	01	0	1	0	0	0	1	1
2	02	0	1	0	1	1	1	0
2	$03 = 02$	0	1	0	1	1	1	0
2	$\cdots$	$\cdots\cdots\cdots\cdots\cdots\cdots\cdots$						

Table 5. Path information for error detection w. r. t. $AGAF\psi$

6.4 Complexity and Discussion

The complexity of $[[M, \varphi]]_k^*$ depends on M, k and $n_k(\varphi)$. For a given k, the number of propositional variables involved in $[[M, \varphi]]_k^*$ is $(k + 1) \cdot n_k(\varphi)$ where $n_k(\varphi) = 2^{O(\log(k) \cdot |\varphi|)}$. This means that the number of propositional variables could be exponential in the length (in practice, in the nesting depth of AR and AU) of φ. We expect that for practical applications, the nesting depth of AR and AU of a formula is small. Then the efficiency depends very much on k which is bounded by the number of reachable states (or more accurately, the diameter) of M.

When a small k is sufficient for the verification, the advantage of this approach is clear. In such cases, it could be much more efficient than BDD based approaches. we provide an example illustrating this advantage.

Let $p_0, ..., p_{n-2}, q, r$ be variables of the domain $\{0, 1\}$ and $\oplus$ be the function: addition modulo 2. Let the system be consist of n processes. A, B and C_i for $i = 0, ..., n-3$ (each is a sequential process which executed in parallel to each other with the interleaving semantics) with the following specification:

$A: \ r = r \oplus 1; p_0 = p_0 \oplus 1$
$B: \ p_{n-2} = p_{n-2} \oplus 1; q = q \oplus 1$
$C_i: p_i = p_i \oplus 1; p_{i+1} = p_{i+1} \oplus 1; q = q \oplus 1$

Let the initial state be $p_i = 0$ and $q = r = 1$.

Let $\varphi = AXA(qU(p_0 \vee p_2 \vee \cdots \vee p_{n-2}))$ for an even number n. For verifying $\varphi(n)$, we first transform the problem to CNF formula, then use zChaff, an implementation of the Chaff algorithm [16] for verification. For $n = 4, 6, 8, 10, 12$, the property is verified when k reaches respectively $2, 3, 4, 5, 6$. The verification times by zChaff for $n = 4, 6, 8, 10, 12$ are shown in Table 6.

Property	k	Time (s)	Variables	Clauses	SAT
$\varphi(4)$	2	0.01	139	1254	no
$\varphi(6)$	3	0.03	278	4077	no
$\varphi(8)$	4	0.11	465	9522	no
$\varphi(10)$	5	0.42	700	18456	no
$\varphi(12)$	6	1.15	983	31746	no

Table 6. Experimental data for verification by zChaff

For comparison, we have carried out the same verification task using SMV (release 2.5.4.3), an implementation of the symbolic model checking technique [15]. The verification times for $n = 4, 6, 8, 10, 12$ are shown in Table 7.

Table 6 and Table 7 show clear advantage of using this bounded verification approach over the BDD based verification approach for this example.

6.5 Summary

The case study shows that this approach can be used to both verification of correct properties and error detection, and the comparison has illustrated that when a small k

Property	Time (s)	BDD nodes	Memory (KB)
$\varphi(4)$	0.06	6092	1245.184
$\varphi(6)$	0.96	14545	1376.256
$\varphi(8)$	12.97	111981	2949.120
$\varphi(10)$	192.01	888025	15335.424
$\varphi(12)$	6596.34	6135235	99287.040

Table 7. Experimental data for verification by SMV

is sufficient for the verification, the advantage is clear. In such cases, it could be much more efficient than BDD based approaches. For error detection, in addition to identifying that there is an error, error paths may also be produced. Creating and analyzing tree-like counter examples have also been studied in many papers including [8, 20]. In our work, the counterexample may be created and presented as a set of k-paths. Although we may use this approach for error detection, it needs to reach a completeness threshold for k. This is usually not very efficient. This approach can be combined with that presented in [18] for error detection. This is to be discussed in the next section.

7 A Combined Verification Approach

Bounded model checking based on SAT (satisfiability checking) has first been introduced as a complementary technique to BDD-based symbolic model checking of LTL properties [2]. This idea has then been used for checking ACTL properties [18]. The characterization, denoted here by $[[\varphi, u_{i,j}]]_k^{*,b}$, is based on a bounded semantics for ECTL and the encoding of ECTL formulas as follows.

$$
\begin{array}{ll}
\hline
[[p, u]]_k^{*,b} & = p(u) \\
[[\neg p, u]]_k^{*,b} & = \neg p(u) \\
[[\varphi \vee \psi, u]]_k^{*,b} & = [[\varphi, u]]_k^{*,b} \vee [[\psi, u]]_k^{*,b} \\
[[\varphi \wedge \psi, u]]_k^{*,b} & = [[\varphi, u]]_k^{*,b} \wedge [[\psi, u]]_k^{*,b} \\
[[EX\varphi, u]]_k^{*,b} & = \bigvee_{i=1}^{b}(u = u_{i,0} \wedge [[\varphi, u_{i,1}]]_k^{*,b}) \\
\hline
[[EG\varphi, u]]_k^{*,b} & = \bigvee_{i=1}^{b}(u = u_{i,0} \wedge \bigwedge_{j=0}^{k}[[\varphi, u_{i,j}]]_k^{*,b} \wedge \bigwedge_{j=0}^{k} T(u_{i,k}, u_{i,j})) \\
[[E(\varphi U \psi), u]]_k^{*,b} & = \bigvee_{i=1}^{b}(u = u_{i,0} \wedge \bigvee_{j=0}^{k}([[\psi, u_{i,j}]]_k^{*,b} \wedge \bigwedge_{t=0}^{j-1}[[\varphi, u_{i,t}]]_k^{*,b})) \\
\hline
\end{array}
$$

where $[[\varphi, u_{i,j}]]_0^{*,b}$ denotes $false$ for $j > 0$. Define

$$[[M, \varphi]]_k^{*,b} := I(u) \wedge [[M]]_k^b \wedge [[\varphi, u]]_k^{*,b}.$$

Let $f_k(\varphi)$ be the sufficient number[4] of paths for a witness (if there is any) of the ECTL formula φ. According to this encoding, we have the following theorem [18].

Theorem 7. *Let φ be an ACTL formula. $M_k \not\models_k \varphi$ iff there is some $k < k_M$ such that $[[M, \neg\varphi]]_k^{*,f_k(\neg\varphi)}$ is satisfiable.*

[4] The computation of $f_k(\varphi)$ is referred to the paper [18].

This theorem can be used as a basis for efficient error detection with SAT-based model checking. The procedure for verification of a given model M against an ACTL formula φ could be as follows:

- Start with $k = 0$;
- If $[[M, \neg\varphi]]_k^{*,f_k(\neg\varphi)}$ is satisfiable, report "the property does not hold";
- Increase k, if $k < k_M$, go to the first "if"-test;
- Report that the property holds.

This can also be used for verification of valid properties. However, it is not efficient for this purpose, since one has to reach the condition with $k = k_M$. Theorem 7 and Theorem 6 can be combined to avoid the use of the completeness threshold k_M.

Corollary 2. *Let φ be an ACTL formula. $M_k \models_k \varphi$ if there is some $k \leq k_M$ such that $[[M, \varphi]]_k^{n_k(\varphi)}$ is valid or for all $k \geq 0$, $[[M, \neg\varphi]]_k^{*,f_k(\neg\varphi)}$ is unsatisfiable. $M_k \not\models_k \varphi$ if $[[M, \varphi]]_k^{n_k(\varphi)}$ is not valid for each $k \geq 0$ or there is some $k < k_M$ such that $[[M, \neg\varphi]]_k^{*,f_k(\neg\varphi)}$ is satisfiable.*

The procedure for the verification of a given model M against an ACTL formula φ could then be as follows:

- Start with $k = 0$;
- If $[[M, \varphi]]_k^{n_k(\varphi)}$ is valid, report "the property holds";
- If $[[M, \neg\varphi]]_k^{*,f_k(\neg\varphi)}$ is satisfiable, report "the property does not hold";
- Increase k, go to the first "if"-test;

The procedure based on Corollary 2 is guaranteed to terminate with a report on whether the property holds. In theory, there is still a completeness threshold that may be reach in some cases of the satisfiability checking. Even in such cases, the advantage is that we do not need to know the completeness threshold for which the cost for the calculation is high [1, 13] and an over-approximation can be quite large.

The complexity of the procedure depends on the number of variables involved in the encoding. For a given k and an ACTL formula φ, the number of variables needed is $k^{O(|\varphi|)}$. The efficiency depends on whether there is a small k which is sufficient to certify or falsify the property. The part $O(|\varphi|)$ is small when there are few levels of nesting temporal operators (which is often the case in the practical property specification and verification).

Related Works There has not been lack of motivation and work for proving properties based on SAT. Related works include, for instance, SAT-based analysis of partial correctness assertions [11, 12], SAT-based proof of safety properties by using induction [21], conservative abstraction with counter example guided refinement [9], and interpolation based transition relation approximation for generating facts relevant with respect to given properties [14]. Proving simple liveness properties based on SAT was also considered in [1]. Recently, SAT-based verification of valid general LTL and ACTL properties has been considered in [23, 24]. The idea is to verify a model of a particular length (as short as possible) and to generate a propositional formula that is unsatisfiable if the model is unsatisfiable with respect to the given property. However the condition in these approaches is only a sufficient condition, not a sufficient and necessary condition, such that there are valid LTL and ACTL properties that cannot be verified by using these approaches alone.

8 Concluding Remarks

Model checking has been considered as one of the most practical applications of the theoretical computer science in the verification of concurrent systems. The practical applicability of explicit state model checking, introduced in [6, 7], is limited by the state explosion problem which could be caused by for instance, the representation of currency of operations by their interleaving. Therefore much effort has been put into the research aiming at minimizing models. Binary Decision Diagram (BDD) based on symbolic techniques has significantly improved the practical applicability of model checking by compactly representing transition relations and system states [4, 3, 5]. Although this is a great success, it has not solved the state explosion problem. For many problems, there is no polynomial size representation with BDD.

For combating this problem, bounded model checking based on SAT (satisfiability checking) has been introduced as a complementary technique to BDD-based symbolic model checking of LTL and ACTL properties in respectively 1999 and 2002 [2, 18]. The basic idea is to search for a counter example of a particular length (as short as possible) and to generate a propositional formula that is satisfied iff such a counter example exists. This idea is similar to that for searching finite models [22] for which we search for counter models of given sizes until we find one.

Prasad, Biere and Gupta pointed out in a survey paper [19] in 2005 that, currently, the strength of SAT-based verification techniques lies primarily in falsification. This is a remark on verification related to general temporal properties. For simple properties, there has been a lot of work and report of success, for instance, for proving simple safety and liveness properties [21, 9, 14, 1]. Recently, SAT-based verification of valid general LTL and ACTL properties has been considered in [23, 24]. However these approaches are based on semantics with existential interpretation and the condition in these approaches is not a sufficient and necessary condition, such that there are valid LTL and ACTL properties that cannot be verified by using these approaches.

This work has provided a bounded semantics for ACTL, and based on this semantics, a refinement has been developed. Then a characterization of ACTL properties by propositional formulas and an approach is presented for the verification of ACTL formulas such that a sufficient and necessary condition is provided. This means that all ACTL properties can either be verified or falsified by using this approach, with the emphasis on verification. For practical application, falsification using this approach depends on a completeness threshold which is not very efficient, and therefore a proposal for combining this approach with the approach based on the bounded semantics for ECTL is suggested for avoiding the use of such a completeness threshold.

ACKNOWLEDGMENTS: The author thanks anonymous referees for their constructive critics that helped improving this paper.

References

1. A. Biere, A. Cimmatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded Model Checking. Advances in Computers 58, Academic Press, 2003.
2. A. Biere, A. Cimmatti, E. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. LNCS 1579:193-207. TACAS 99.
3. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. LICS 1990: 428-439.
4. R. Bryant. Graph based algorithms for boolean function manipulation. IEEE Transaction on Computers 35(8):677-691. 1986.
5. R. Bryant. Binary decision diagrams and beyond: enabling technologies for formal verification. CAD'95:236-243. 1995.
6. E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. Lecture Notes in Computer Science, volume 131. Springer-Verlag, 1981
7. E. M. Clarke, E. A. Emerson and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems, 8(2):244–263, 1986.
8. Edmund M. Clarke, Somesh Jha, Yuan Lu, Helmut Veith: Tree-Like Counterexamples in Model Checking. LICS 2002: 19-29.
9. Satyaki Das and David L. Dill. Successive Approximation of Abstract Transition Relations. LICS 2001: 51-60.
10. E. Allen Emerson and E. M. Clarke. Using Branching-time Temporal Logics to Synthesize Synchronization Skeletons. Science of Computer Programming 2(3):241-266. 1982.
11. Marcelo F. Frias, Juan P. Galeotti, Carlos Lopez Pombo and Nazareno Aguirre. DynAlloy: upgrading alloy with actions. ICSE 2005: 442-451.
12. Marcelo F. Frias, Carlos Lopez Pombo, Gabriel A. Baum, Nazareno Aguirre and T. S. E. Maibaum: Reasoning about static and dynamic properties in alloy: A purely relational approach. ACM Trans. Softw. Eng. Methodol. 14(4): 478-526 (2005).
13. D. Kroening, O. Strichman. Efficient Computation of Recurrence Diameters. VMCAI 2003: 298-309.
14. Ranjit Jhala and Kenneth L. McMillan. Interpolation and SAT-Based Model Checking. CAV 2003: 1-13.
15. McMillan K L. Symbolic Model Checking. Kluwer Academic Publisher,1993.
16. M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. DAC 2001.
17. Doron A. Peled. Software Reliability Methods. Springer-Verlag. 2001.
18. W. Penczek, B. Wozna, and A. Zbrzezny. Bounded Model Checking for the Universal Fragment of CTL. Fundamenta Informaticae 51:135-156. 2002.
19. Mukul R. Prasad, Armin Biere, Aarti Gupta. A survey of recent advances in SAT-based formal verification. STTT 7(2): 156-173 (2005).
20. Sharon Shoham, Orna Grumberg: A Game-Based Framework for CTL Counterexamples and 3-Valued Abstraction-Refinement. CAV 2003: 275-287.
21. Mary Sheeran, Satnam Singh and Gunnar Stålmarck. Checking Safety Properties Using Induction and a SAT-Solver. FMCAD 2000: 108-125.
22. J. Zhang. Problems on the generation of finite models. LNAI 814 (CADE 1994):753-757.
23. W. Zhang. SAT-based verification of LTL formulas. Lecture Notes in Computer Science (FMICS 2006) 4346:277-292, Springer-Verlag, 2007.
24. W. Zhang. Verification of ACTL properties by bounded model checking. EUROCAST 2007.
25. W. Zhang, Z. Huang, and J. Zhang. Parallel Execution of Stochastic Search Procedures on Reduced SAT Instances. Lecture Notes in Computer Science 2417 (PRICAI 2002):108-117. 2002.