QBF Encoding of Temporal Properties and QBF-Based Verification*

Wenhui Zhang

State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
P.O. Box 8718, Beijing 100190, China
zwh@ios.ac.cn

Abstract. SAT and QBF solving techniques have applications in various areas. One area of the applications of SAT-solving is formal verification of temporal properties of transition system models. Because of the restriction on the structure of formulas, complicated verification problems cannot be naturally represented with SAT-formulas succinctly. This paper investigates QBF-applications in this area, aiming at the verification of branching-time temporal logic properties of transition system models. The focus of this paper is on temporal logic properties specified by the extended computation tree logic that allows some sort of fairness, and the main contribution of this paper is a bounded semantics for the extended computation tree logic. A QBF encoding of the temporal logic is then developed from the definition of the bounded semantics, and an implementation of QBF-based verification follows from the QBF encoding. Experimental evaluation of the feasibility and the computational properties of such a QBF-based verification algorithm is reported.

1 Introduction

SAT and QBF solving techniques have applications in various areas [10,13,9]. One area of the applications of SAT-solving is formal verification of temporal properties of transition system models [1,14,2,15,19,12,6]. In various situations, it can be used to quickly determine whether a property is violated and is considered as a complementary approach to the standard BDD-based verification approaches [3,16,5]. Therefore a large number of research works has been devoted into this direction. However, because of the restriction on the structure of formulas, complicated verification problems cannot be naturally represented with SAT-formulas succinctly. This paper investigates QBF-applications in this area, aiming at the verification of branching-time temporal logic properties of transition system models. Branching-time temporal logic properties involve operators that may require the existence of certain kinds of paths starting at different states of a system model. This requires the use of quantifiers in the encoding of such properties.

* Supported by the National Natural Science Foundation of China under Grant No. 61272135 and the 973 Program of China under Grant No. 2014CB340701.

S. Demri, D. Kapur, and C. Weidenbach (Eds.): IJCAR 2014, LNAI 8562, pp. 224–239, 2014.

This paper focuses on QBF encoding and QBF-based verification of temporal properties specified by the extended computation tree logic that allows some sort of fairness [8]. The main contribution of this paper is a bounded semantics for the extended computation tree logic. A QBF encoding of the temporal logic is then developed from the definition of the bounded semantics, and an implementation of QBF-based verification follows from the QBF encoding. One of the particular aspects of this implementation is that it can handle properties (branching properties combined with fairness) that are not handled by well known model checking tools such as Spin [11] and NuSMV [4]. Finally experimental evaluation of the feasibility of the QBF-based verification relative to BDD-based verification is reported.

2 Preliminaries

We recall the definition of transition system models and that of the extended computation tree logic.

2.1 Transition System Models

Let AP be a set of propositional symbols. A finite state system may be represented by a Kripke structure which is a quadruple $M = \langle S, T, I, L\rangle$ 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 $L : S \to 2^{AP}$ is a labeling function that maps each state to a subset of propositions of AP. A Kripke structure is also called a model.

Transitions A transition from a state s to another state s' is denoted $s \to s'$. $s \to s'$ iff $(s, s') \in T$.

Paths. An infinite path is an infinite sequence of states $\pi = \pi_0\pi_1 \cdots$ such that $\pi_i \to \pi_{i+1}$ for all $i \geq 0$.

Computations. A computation of M is an infinite path such that the initial state of the path is in I.

Notations. Let $\pi = \pi_0\pi_1 \cdots$ be a path. We use $\pi(s)$ to denote a path π with $\pi_0 = s$. Then $\exists\pi(s).\varphi$ means that there is a path π with $\pi_0 = s$ such that φ holds, and $\forall\pi(s).\varphi$ means that for every path π with $\pi_0 = s$, φ holds.

2.2 Extended Computation Tree Logic (eCTL)

Properties of a transition system model may be specified by temporal logic formulas. Extended computation tree logic [8] is a propositional branching-time temporal logic that extends the computation tree logic (CTL) introduced by Emerson and Clarke [7] with possibility to express simple fairness constraints. For brevity, the extended computation tree logic is hereafter denoted eCTL .

Syntax Let p range over AP. The set of eCTL formulas Φ over AP is defined as follows:

$$\Phi ::= p \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid A\Psi \mid E\Psi$$
$$\Psi ::= X\,\Phi \mid F\,\Phi \mid G\,\Phi \mid \overset{\infty}{F}\,\Phi \mid \overset{\infty}{G}\,\Phi \mid (\Phi\, U\, \Phi) \mid (\Phi\, R\, \Phi)$$

The formulas of Φ are eCTL formulas, and the formulas of Ψ are auxiliary path formulas. The property of a finite state system may be specified by an eCTL formula, and conversely, the truth of such a formula may be evaluated in a finite state system.

Definition 1. *(Semantics of eCTL) Let p denote a propositional symbol, and $\varphi, \varphi_0, \varphi_1$ denote eCTL formulas, ψ, ψ_0 denote path formulas. Let s be a state and π be a path of M. Let $M, s \models \varphi$ denote the relation that φ holds on s of M, and $M, \pi \models \psi$ denote that ψ holds on π of M. The relation $M, s \models \varphi$ and $M, \pi \models \psi$ are defined as follows.*

$M, s \models p$ *iff* $p \in L(s)$.
$M, s \models \neg\varphi_0$ *iff* $M, s \not\models \varphi_0$
$M, s \models \varphi_0 \wedge \varphi_1$ *iff* $M, s \models \varphi_0$ *and* $M, s \models \varphi_1$
$M, s \models \varphi_0 \vee \varphi_1$ *iff* $M, s \models \varphi_0$ *or* $M, s \models \varphi_1$
$M, s \models A\psi_0$ *iff* $\forall\pi(s).(M, \pi \models \psi_0)$
$M, s \models E\psi_0$ *iff* $\exists\pi(s).(M, \pi \models \psi_0)$
$M, \pi \models X\varphi_0$ *iff* $M, \pi_1 \models \varphi_0$
$M, \pi \models F\varphi_0$ *iff* $\exists k \geq 0. M, \pi_k \models \varphi_0$
$M, \pi \models G\varphi_0$ *iff* $\forall k \geq 0. M, \pi_k \models \varphi_0$
$M, \pi \models \overset{\infty}{F}\varphi_0$ *iff* $\forall i \geq 0. \exists k \geq i. M, \pi_k \models \varphi_0$
$M, \pi \models \overset{\infty}{G}\varphi_0$ *iff* $\exists i \geq 0. \forall k \geq i. M, \pi_k \models \varphi_0$
$M, \pi \models \varphi_0 U \varphi_1$ *iff* $\exists k \geq 0. (M, \pi_k \models \varphi_1 \wedge \forall 0 \leq j < k. (M, \pi_j \models \varphi_0))$
$M, \pi \models \varphi_0 R \varphi_1$ *iff* $\forall k \geq 0. (M, \pi_k \models \varphi_1 \vee \exists 0 \leq j < k. (M, \pi_j \models \varphi_0))$

Definition 2. $M \models \varphi$ *iff* $M, s \models \varphi$ *for all* $s \in I$.

Negation Normal Form. An eCTL formula is in the negation normal form (NNF), if the negation $\neg$ is applied only to propositional symbols. Every eCTL formula can be transformed into an equivalent formula in NNF by using the following equivalences.

$\neg(\varphi \wedge \psi)$	$\equiv \neg\varphi \vee \neg\psi$	$\neg(\varphi \vee \psi)$	$\equiv \neg\varphi \wedge \neg\psi$
$\neg AX\,\varphi$	$\equiv EX\neg\varphi$	$\neg EX\,\varphi$	$\equiv AX\neg\varphi$
$\neg AF\,\varphi$	$\equiv EG\neg\varphi$	$\neg EF\,\varphi$	$\equiv AG\neg\varphi$
$\neg AG\,\varphi$	$\equiv EF\neg\varphi$	$\neg EG\,\varphi$	$\equiv AF\neg\varphi$
$\neg A\overset{\infty}{F}\,\varphi$	$\equiv E\overset{\infty}{G}\neg\varphi$	$\neg E\overset{\infty}{F}\,\varphi$	$\equiv A\overset{\infty}{G}\neg\varphi$
$\neg A\overset{\infty}{G}\,\varphi$	$\equiv E\overset{\infty}{F}\neg\varphi$	$\neg E\overset{\infty}{G}\,\varphi$	$\equiv A\overset{\infty}{F}\neg\varphi$
$\neg A(\varphi\, U\, \psi)$	$\equiv E(\neg\varphi\, R\, \neg\psi)$	$\neg E(\varphi\, U\, \psi)$	$\equiv A(\neg\varphi\, R\, \neg\psi)$
$\neg A(\varphi\, R\, \psi)$	$\equiv E(\neg\varphi\, U\, \neg\psi)$	$\neg E(\varphi\, R\, \psi)$	$\equiv A(\neg\varphi\, U\, \neg\psi)$

Without loss of generality, we only consider formulas in NNF. Formulas not in NNF are considered as an abbreviation of the equivalent ones in NNF.

3 Bounded Semantics

Before presenting the QBF encoding of temporal properties, we develop a bounded semantics for eCTL. This bounded semantics extends that of the previous works for that of the existential fragment of CTL [14] and that of CTL [19]. For convenience, we fix the model under consideration to be $M = \langle S, T, I, L\rangle$ in the rest of this paper.

Finite Paths and k-Paths. A finite path is a finite prefix of an infinite path. Let $k \geq 0$. A k-path of M is a finite path of M with length $k+1$. π is a k-path, if $\pi = \pi_0 \cdots \pi_k$ such that $\pi_i \in S$ for $i = 0, ..., k$ and $(\pi_i, \pi_{i+1}) \in T$ for $i = 0, ..., k-1$.

Bounded Models. The k-model of M is a quadruple $M_k = \langle S, Ph_k, I, L\rangle$ where Ph_k is the set of all k-paths of M. M_k can be considered as an approximation of M.

Paths with Repeating States (rs-paths). An rs-path is a finite path that contains repeating states (i.e., at least two states are the same). Let $rs(\pi)$ denote that π is an rs-path. An important property of such a path is that if π is a prefix of π', then $rs(\pi) \rightarrow rs(\pi')$. For the ideas of k-paths, k-models, and rs-paths, the reader is referred to [1,14,19].

Definition 3. *(Bounded Semantics of eCTL) Let p denote a propositional symbol, and $\varphi, \varphi_0, \varphi_1$ denote eCTL formulas, ψ, ψ_0 denote path formulas. Let s be a state of M and π be a k-path of Ph_k. Let $M_k, s \models \varphi$ denote the relation that φ holds on s of M_k, and $M_k, \pi \models \psi$ denote that ψ holds on π of M_k. The relation $M_k, s \models \varphi$ and $M_k, \pi \models \psi$ are defined as follows.*

$M_k, s \models p$ *iff* $p \in L(s)$.
$M_k, s \models \neg p$ *iff* $p \notin L(s)$
$M_k, s \models \varphi_0 \wedge \varphi_1$ *iff* $(M_k, s \models \varphi_0)$ *and* $(M_k, s \models \varphi_1)$
$M_k, s \models \varphi_0 \vee \varphi_1$ *iff* $(M_k, s \models \varphi_0)$ *or* $(M_k, s \models \varphi_1)$
$M_k, s \models A\psi$ *iff* $\forall \pi(s).(M_k, \pi \models \psi)$
$M_k, s \models E\psi$ *iff* $\exists \pi(s).(M_k, \pi \models \psi)$
$M_k, \pi \models X\varphi_0$ *iff* $k \geq 1 \wedge (M_k, \pi_1 \models \varphi_0)$
$M_k, \pi \models F\varphi_0$ *iff* $\exists i \leq k.(M_k, \pi_i \models \varphi_0)$
$M_k, \pi \models G\varphi_0$ *iff* $rs(\pi) \wedge (\forall i \leq k.(M_k, \pi_i \models \varphi_0))$
$M_k, \pi \models \overset{\infty}{F}\varphi_0$ *iff* $rs(\pi) \wedge \forall i < l \leq k.(\pi_i = \pi_l \rightarrow \exists i < j \leq l.(M_k, \pi_j \models \varphi_0))$
$M_k, \pi \models \overset{\infty}{G}\varphi_0$ *iff* $rs(\pi) \wedge \forall i < l \leq k.(\pi_i = \pi_l \rightarrow \forall i < j \leq l.(M_k, \pi_j \models \varphi_0))$
$M_k, \pi \models \varphi_0 U \varphi_1$ *iff* $\exists i \leq k.(M_k, \pi_i \models \varphi_1 \wedge \forall j < i.(M_k, \pi_j \models \varphi_0))$
$M_k, \pi \models \varphi_0 R \varphi_1$ *iff* $\forall i \leq k.(M_k, \pi_i \models \varphi_1 \vee \exists j < i.(M_k, \pi_j \models \varphi_0)) \wedge (\exists j \leq k.(M_k, \pi_j \models \varphi_0) \vee rs(\pi))$

Definition 4. *$M_k \models \varphi$ iff $M_k, s \models \varphi$ for all $s \in I$.*

Let $|M|$ denote the number of states of M.

Lemma 1. *If $M, s \models \varphi$, then there is a $k \geq 0$ such that $M_k, s \models \varphi$.*

Proof: The proof is done by structural induction. For brevity (due to the page limit), we prove the two cases where φ is respectively $A\overset{\infty}{G}\varphi_0$ and $A\overset{\infty}{F}\varphi_0$, and omit the rest of the cases. Let $k = |M|$.

- Suppose that $M, s \models A\overset{\infty}{G}\varphi_0$ holds.
  Since $k = |M|$ and the transition relation is total, the length of every k-path is greater than $|M|$. Then for every k-path π, $rs(\pi)$ holds. We only need to show for every k-path π starting at s the following holds.

$$\forall i < l \leq k.(\pi_i = \pi_l \rightarrow \forall i < j \leq l.(M_k, \pi_j \models \varphi_0)).$$

  Assume $\pi_i = \pi_l$ for a k-path π. Then $\pi' = \pi_0 \cdots \pi_i(\pi_{i+1} \cdots \pi_l)^\omega$ is an infinite path starting at $s = \pi_0$. Since π' satisfies $\overset{\infty}{G}\varphi_0$, we have $\forall i < j \leq l.(M, \pi_j \models \varphi_0)$. Then according to the induction hypothesis, $\forall i < j \leq l.(M_k, \pi_j \models \varphi_0)$.
- Suppose that $M, s \models A\overset{\infty}{F}\varphi_0$ holds.
  Since $k = |M|$, for every k-path π, $rs(\pi)$ holds. We only need to show for every k-path π starting at s the following holds.

$$\forall i < l \leq k.(\pi_i = \pi_l \rightarrow \exists i < j \leq l.(M_k, \pi_j \models \varphi_0)).$$

  Assume $\pi_i = \pi_l$ for a k-path π. Then $\pi' = \pi_0 \cdots \pi_i(\pi_{i+1} \cdots \pi_l)^\omega$ is an infinite path starting at $s = \pi_0$. Since π' satisfies $\overset{\infty}{F}\varphi_0$, we have $\exists i < j \leq l.(M, \pi_j \models \varphi_0)$. Then according to the induction hypothesis, $\exists i < j \leq l.(M_k, \pi_j \models \varphi_0)$.

Lemma 2. *If $M_k, s \models \varphi$ for $k \geq |M|$, then $M, s \models \varphi$.*

Proof: The proof is done by structural induction. For brevity, we prove the two cases where φ is respectively $A\overset{\infty}{G}\varphi_0$ and $A\overset{\infty}{F}\varphi_0$, and omit the rest of the cases.

- Suppose that $M_k, s \models A\overset{\infty}{G}\varphi_0$ holds for k.
  Then for every k-path π' starting at s, we have $M_k, \pi' \models \overset{\infty}{G}\varphi_0$, i.e.,

$$rs(\pi') \wedge \forall i < l \leq k.(\pi'_i = \pi'_l \rightarrow \forall i < j \leq l.(M_k, \pi'_j \models \varphi_0)).$$

  Assume that $M, s \models A\overset{\infty}{G}\varphi_0$ does not hold. We show that this is a contradiction. According to this assumption, there is an infinite path starting at s such that $M, \pi \models \overset{\infty}{G}\varphi_0$ does not hold. Then we can construct an infinite path $\pi' = \pi'_0 \cdots \pi'_i(\pi'_{i+1} \cdots \pi'_l)^\omega$ starting at s such that $l \leq |M|$, $\pi'_x \neq \pi'_y$ for all $x < y < l$, $\pi'_i = \pi'_l$, and $M, \pi'_j \not\models \varphi_0$ for some $i < j \leq l$. Let π'' be a k-path

with $\pi'_0 \cdots \pi'_i \pi'_{i+1} \cdots \pi'_l$ as its prefix (this is possible, since $l \leq |M| \leq k$). Then according to the premise of the lemma, $M_k, \pi'' \models \overset{\infty}{G}\varphi_0$ holds. Then we have $M_k, \pi'_j \models \varphi_0$, and by the induction hypothesis, $M, \pi'_j \models \varphi_0$. This is a contradiction, which proves the lemma.

– Suppose that $M_k, s \models \overset{\infty}{AF}\varphi_0$ holds for k.

Then for every k-path π' starting at s, we have $M_k, \pi' \models \overset{\infty}{F}\varphi_0$, i.e.,

$$rs(\pi') \wedge \forall i < l \leq k.(\pi'_i = \pi'_l \rightarrow \exists i < j \leq l.(M_k, \pi'_j \models \varphi_0)).$$

Assume that $M, s \models \overset{\infty}{AF}\varphi_0$ does not hold. We show that this is a contradiction. According to this assumption, there is an infinite path starting at s such that $M, \pi \models \overset{\infty}{F}\varphi_0$ does not hold. Then we can construct an infinite path $\pi' = \pi'_0 \cdots \pi'_i (\pi'_{i+1} \cdots \pi'_l)^\omega$ starting at s such that $l \leq |M|$, $\pi'_x \neq \pi'_y$ for all $x < y < l$, $\pi'_i = \pi'_l$, and $M, \pi'_j \not\models \varphi_0$ for all $i < j \leq l$. Let π'' be a k-path with $\pi'_0 \cdots \pi'_i \pi'_{i+1} \cdots \pi'_l$ as its prefix (this is possible, since $l \leq |M| \leq k$). Then according to the premise of the lemma, $M_k, \pi'' \models \overset{\infty}{F}\varphi_0$ holds. Then we have $\exists i < j \leq l.(M_k, \pi'_j \models \varphi_0)$, and by the induction hypothesis, $\exists i < j \leq l.(M, \pi'_j \models \varphi_0)$. This is a contradiction, which proves the lemma.

Lemma 3. *If $M_k, s \models \varphi$, then $M_{k+1}, s \models \varphi$.*

Proof: The proof is done by structural induction. For brevity, we prove the case where φ is $\overset{\infty}{AG}\varphi_0$, and omit the rest of the cases. Suppose that $M_k, s \models \overset{\infty}{AG}\varphi_0$ holds for k.

Then for every k-path ζ starting at s, we have $M_k, \zeta \models \overset{\infty}{G}\varphi_0$, i.e.,

$$rs(\zeta) \wedge \forall i < l \leq k.(\zeta_i = \zeta_l \rightarrow \forall i < j \leq l.(M_k, \zeta_j \models \varphi_0)).$$

Then according to the induction hypothesis, we have

$$rs(\zeta) \wedge \forall i < l \leq k.(\zeta_i = \zeta_l \rightarrow \forall i < j \leq l.(M_{k+1}, \zeta_j \models \varphi_0)).$$

The goal is to prove that $M_{k+1}, s \models \overset{\infty}{AG}\varphi_0$ holds, i.e., for every $(k+1)$-path π of M_{k+1} starting at s, the following two properties hold.

(1) $rs(\pi)$
(2) $\forall i < l \leq k+1.(\pi_i = \pi_l \rightarrow \forall i < j \leq l.(M_{k+1}, \pi_j \models \varphi_0))$

Let $\pi = \pi_0 \cdots \pi_k \pi_{k+1}$ be a $(k+1)$-path starting at s. Since $\pi' = \pi_0 \cdots \pi_k$ is a k-path, we have the following fact.

$$rs(\pi') \wedge \forall i < l \leq k.(\pi_i = \pi_l \rightarrow \forall i < j \leq l.(M_{k+1}, \pi_j \models \varphi_0))$$

Since $rs(\pi')$ implies property (1), and the cases of property (2) where $l \leq k$ are covered by the fact, we only need to show

$$\forall i < k+1.(\pi_i = \pi_{k+1} \rightarrow \forall i < j \leq k+1.(M_{k+1}, \pi_j \models \varphi_0)).$$

Since $rs(\pi')$ holds, there is $x < y \leq k$ such that $\pi_x = \pi_y$.
Assume $\pi_i = \pi_{k+1}$. We divide the rest of the proof into two cases:

- $x < i < y$:
  Let π'' be the concatenation of $\pi_0 \cdots \pi_x$, $\pi_{y+1} \cdots \pi_{k+1}$, $\pi_{i+1} \cdots \pi_y$.
  Then π'' is a prefix of a k-path starting at s and, since $\pi_x = \pi_y$, every state s' between π_x and π_y satisfies φ_0 ($M_k, s' \models \varphi_0$ according to the premise and $M_{k+1}, s' \models \varphi_0$ according to the induction hypothesis).
  Therefore $\forall i < j \leq k+1.(M_{k+1}, \pi_j \models \varphi_0)$.
- $i \leq x$ or $y \leq i$:
  Let π'' be the path obtained by removing $\pi_{x+1} \cdots \pi_y$ from π.
  Then π'' is a prefix of a k-path starting at s and, therefore every state between π_i and π_{k+1} (of π'') satisfies φ_0, according to the premise and the induction hypothesis.
  In addition, every state in the partial-path $\pi_{x+1} \cdots \pi_y$ also satisfies φ_0 (this is needed in case $i \leq x$). Therefore we have $\forall i < j \leq k+1.(M_{k+1}, \pi_j \models \varphi_0)$.

Theorem 1 (Soundness and Completeness). *$M, s \models \varphi$ iff $M_k, s \models \varphi$ for some $k \geq 0$.*

The soundness and completeness of the bounded semantics follows from Lemma 1, Lemma 2 and Lemma 3.

Corollary 1. *$M \models \varphi$ iff $M_k \models \varphi$ for some $k \geq 0$.*

4 QBF Encoding and QBF-Based Verification

From the bounded semantics, a QBF-based characterization of eCTL formulas, extending that of CTL formulas [20], can be developed as follow. Let $k \geq 0$. Let $u_0, ..., u_k$ be a finite sequence of state variables. The sequence $u_0, ..., u_k$ (denoted by $\vec{u}$) is intended to be used as a representation of a path of M_k. This is captured by the following definition of $P_k(\vec{u})$.

Definition 5

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

Every assignment to the set of state variables $\{u_0, ..., u_k\}$ satisfying $P_k(\vec{u})$ represents a valid k-path of M. Let $rs_k(\vec{u})$ denote that the k-path represented by $\vec{u}$ is an rs-path. Formally, we have the following definition of $rs_k(\vec{u})$.

Definition 6

$$rs_k(\vec{u}) := \bigvee_{x=0}^{k-1} \bigvee_{y=x+1}^{k} u_x = u_y.$$

Let $p \in AP$ be a proposition symbol and $p(v)$ be the propositional formula such that $p(v)$ is true whenever v is assigned the truth value representing a state s in which p holds.

Definition 7 (Transformation of eCTL Formulas). *Let $k \geq 0$. Let v be a state variable and φ be an eCTL formula. The encoding $[[\varphi, v]]_k$ is defined as follows.*

$[[p, v]]_k$	$= p(v)$
$[[\neg p, v]]_k$	$= \neg p(v)$
$[[\varphi \vee \psi, v]]_k$	$= [[\varphi, v]]_k \vee [[\psi, v]]_k$
$[[\varphi \wedge \psi, v]]_k$	$= [[\varphi, v]]_k \wedge [[\psi, v]]_k$
$[[A\varphi, v]]_k$	$= \forall \vec{u}.(P(\vec{u}) \wedge v = u_0 \rightarrow [[\varphi, \vec{u}]]_k)$
$[[E\varphi, v]]_k$	$= \exists \vec{u}.(P(\vec{u}) \wedge v = u_0 \wedge [[\varphi, \vec{u}]]_k)$
$[[X\varphi, \vec{u}]]_k$	$= k \geq 1 \wedge [[\varphi, u_1]]_k$
$[[F\psi, \vec{u}]]_k$	$= \bigvee_{j=0}^{k} [[\psi, u_j]]_k$
$[[G\psi, \vec{u}]]_k$	$= \bigwedge_{j=0}^{k} [[\psi, u_j]]_k \wedge rs_k(\vec{u}))$
$[[\overset{\infty}{F}\psi, \vec{u}]]_k$	$= rs_k(\vec{u}) \wedge \bigwedge_{i=0}^{k} (\bigwedge_{l=i+1}^{k} (u_i = u_l \rightarrow \bigvee_{j=i+1}^{l} [[\psi, u_j]]_k))$
$[[\overset{\infty}{G}\psi, \vec{u}]]_k$	$= rs_k(\vec{u}) \wedge \bigwedge_{i=0}^{k} (\bigwedge_{l=i+1}^{k} (u_i = u_l \rightarrow \bigwedge_{j=i+1}^{l} [[\psi, u_j]]_k))$
$[[\varphi U \psi, \vec{u}]]_k$	$= \bigvee_{j=0}^{k} ([[\psi, u_j]]_k \wedge \bigwedge_{t=0}^{j-1} [[\varphi, u_t]]_k)$
$[[\varphi R \psi, \vec{u}]]_k$	$= \bigwedge_{j=0}^{k} ([[\psi, u_j]]_k \vee \bigvee_{t=0}^{j-1} [[\varphi, u_t]]_k) \wedge (\bigvee_{t=0}^{k} [[\varphi, u_t]]_k \vee rs_k(\vec{u}))$

Note that the transition relation of M is total, and therefore every finite path either can be extended to a k-path or has a k-path as its prefix. Let $v(s)$ denote that the state variable v has been assigned a value corresponding to the state s. The following theorem follows from the transformation scheme.

Theorem 2. *Let φ be an eCTL formula. $M_k, s \models \varphi$ iff $[[\varphi, v(s)]]_k$ holds.*

Let $I(v)$ denote the propositional formula that restricts potential values of v to the initial states of M.

Corollary 2. *Let φ be an eCTL formula. $M \models \varphi$ iff there is a $k \geq 0$ such that $\forall v.(I(v) \rightarrow [[\varphi, v]]_k)$, and $M \not\models \varphi$ iff there is a $k \geq 0$ such that $\exists v.(I(v) \wedge [[\neg\varphi, v]]_k)$.*

Following from Theorem 1, we have $M \models \varphi$ iff there is a $k \geq 0$ such that $M_k \models \varphi$. According to Theorem 2, we have $M \models \varphi$ iff there is a $k \geq 0$ such that $\forall v.(I(v) \rightarrow [[\varphi, v]]_k)$. The second part of the corollary is shown as follows.

- Suppose that $M \not\models \varphi$.
  Then $(\exists s \in I, M, s \not\models \varphi)$, and therefore $(\exists s \in I, M, s \models \neg\varphi)$.
  According to Theorem 1, $(\exists s \in I, \exists k \geq 0, M_k, s \models \neg\varphi)$.
  Therefore there is a $k \geq 0$ such that $\exists s \in I$, $M_k, s \models \neg\varphi$ holds, and then there is a $k \geq 0$ such that $\exists v.(I(v) \wedge [[\neg\varphi, v]]_k)$, according to Theorem 2.

– On the other hand, suppose that $M \models \varphi$.
  Then $\forall s \in I, M, s \models \varphi$, and therefore $\neg(\exists s \in I, M, s \models \neg\varphi)$.
  According to Theorem 1, $\neg(\exists s \in I, \exists k \geq 0, M_k, s \models \neg\varphi)$.
  Therefore $\neg(\exists k \geq 0, \exists s \in I, M_k, s \models \neg\varphi)$.
  Therefore $\neg(\exists k \geq 0, \exists v.(I(v) \wedge [[\neg\varphi, v]]_k))$, according to Theorem 2.

Bounded Correctness Checking Let φ be an eCTL formula. Following from Corollary 2, we can formulate a bounded correctness checking algorithm for $M \models \varphi$, as follows.

Init $k = 0$; .
If $\forall v.(I(v) \rightarrow [[\varphi, v]]_k)$ holds, report that φ holds;
If $\exists v.(I(v) \wedge [[\neg\varphi, v]]_k)$ holds, report that φ does not hold;
Increase k, go to the first "if"-test;

The correctness and the termination are guaranteed by Corollary 2. The algorithm is a combination of checking whether φ holds directly by the bounded semantics, and on the other hand checking whether φ does not hold also by the bounded semantics. The latter part is in accordance with the traditional bounded model checking approach [1].

5 Implementation and Experimental Evaluation

The implementation of the bounded correctness checking algorithm involves the following functionalities:

– For the finite state program, convert the program into a Boolean program;
– Produce a Boolean formula for the initial states (i.e., $I(v)$);
– Produce a Boolean formula for the transition relation (i.e., $T(v, v')$);
– For the property specified by an eCTL formula with a given k, produce a QBF-formula according to the transformation scheme;
– Combine the QBF-formula with the Boolean formula representing the initial states;
– Apply a QBF-solving algorithm to check the truth of the combined formula.

The proposed QBF-based approach has been implemented in a verification tool, denoted VERDS[1], and an experimental evaluation has been carried out. We first present an example to show the application of the approach, and then the experimental evaluation is reported.

5.1 An Illustrative Example

The example is a concurrent program representing a formulation of Peterson's mutual exclusion algorithm [18] as a first order transition system [17]. 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:

[1] `http://lcs.ios.ac.cn/~zwh/verds/`

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

Let the formula specifying the set of the initial states be $a = s_0 \wedge b = t_0 \wedge x = y = 0$. The value of t is arbitrary at the initial state. The following explains the meaning of some of the constants.

$a = s_i$:	process A is waiting for entering the critical region when $i = 1$, is in the critical region when $i = 2$, has left the critical region when $i = 3$.
$b = t_i$:	process B is waiting for entering the critical region when $i = 1$, is in the critical region when $i = 2$, has left the critical region when $i = 3$.

Let the following be the properties of the program we want to verify.

p_1:	$AF(a = s_2 \vee b = t_2)$
p_2:	$AG(\neg(a = s_2 \wedge b = t_2))$
p_3:	$\overset{\infty}{AG}(((a = s_1) \rightarrow AF(a = s_2)) \wedge ((b = t_1) \rightarrow AF(b = t_2)))$
p_4:	$\overset{\infty}{AG}(((a = s_1) \rightarrow EF(a = s_2)) \wedge ((b = t_1) \rightarrow EF(b = t_2)))$

The first 2 properties are simple CTL properties for mutual exclusion algorithms, and the last 2 properties are particular eCTL properties.

Verification The input to the verification tool VERDS must be written in the language specified in [21]. Let the input be as follows, in which the temporal operator $\overset{\infty}{AG}$ is written as AFG.

```
VVM
VAR   x:0..1; y:0..1; t:0..1; a:{s0,s1,s2,s3}; b:{t0,t1,t2,t3};
INIT  x=0; y=0; a=s0; b=t0;
TRANS a=s0:            (y,t,a):=(1,1,s1);
      a=s1&(x=0|t=0):  (a):=(s2);
      a=s2:            (y,a):=(0,s3);
      a=s2:            (a):=(s2);
      a=s3:            (y,t,a):=(1,1,s1);
      b=t0:            (x,t,b):=(1,0,t1);
      b=t1&(y=0|t=1):  (b):=(t2);
      b=t2:            (x,b):=(0,t3);
      b=t2:            (b):=(t2);
      b=t3:            (x,t,b):=(1,0,t1);
SPEC  AF((a=s2|b=t2));
      AG(!(a=s2&b=t2));
      AFG((!a=s1|AF(a=s2))&(!b=t1|AF(b=t2)));
      AFG((!a=s1|EF(a=s2))&(!b=t1|EF(b=t2)));
```

Suppose that the input is contained in the file "tn1mutex.vvm". For checking the i-th property, we use the following command, where i is to be replaced by a given number.

```
verds -QBF -ck i tn1mutex.vvm
```

The verification result for the third property (with $i = 3$) is shown as follows.

VERSION:	verds 1.45 - JAN 2014
FILE:	tn1mutex.vvm
PROPERTY:	$AFG((!(a = 1)\|AF(a = 2)\&(!(b = 1)\|AF(b = 2)))$
INFO:	applying an internal QBF-solver
bound =	0
.	
.	
bound =	4
CONCLUSION:	FALSE

The verification process for the other properties are similar. A summary of the verification results is as follows, where the first row specifies the properties, and 2nd and 3rd row show respectively the satisfiability of the formula and the least k for certifying the satisfiability.

	p_1	p_2	p_3	p_4
T/F	T	T	F	T
k	3	10	4	10

The above example shows that, for some problem instances, the satisfiability or unsatisfiability may be determined when k is relatively small. In such cases, the QBF-based verification may have advantage over the traditional symbolic model checking approach. In the following, we present a comparison of such an approach with the traditional model checking approach.

5.2 Experimental Evaluation

This subsection contains a summary of an experimental evaluation of QBF-based verification implemented in VERDS (to be referred to as VERDS-QBF in the rest of the paper). The experimental evaluation compares this QBF-based verification with BDD-based verification implemented in NuSMV [4] version 2.5.0[2]. The comparison is based on the use of two types of random Boolean programs and 24 properties. A description of the programs and the properties is as follows.

Remarks. The comparison is not meant to draw a conclusion on which verification approach is better. Rather, it will show that there is a large number of cases on which one approach is better than the other, and vice versa, and in this sense the two approaches may be considered complementary.

[2] `http://nusmv.irst.itc.it/`

Programs with Concurrent Processes. The parameters of the first set of random Boolean programs are as follows:

a: number of processes
b: number of all variables
c: number of share variables
d: number of local variables in a process

The shared variables are initially set to a random value in $\{0, 1\}$, and the local variables are initially set to 0. For each process, the shared variables and the local variables are assigned the negation of a variable randomly chosen from these variables.

Programs with Concurrent Sequential Processes. The parameters of the second set of random Boolean programs are as follows, in addition to a, b, c, d specified above.

t: number of transitions in a process
p: number of parallel assignments in each transition

For each concurrent sequential process, besides the b Boolean variables, there is a local variable representing program locations, with e possible values. The shared variables are initially set to a random value in $\{0, 1\}$, and the local variables are initially set to 0. For each transition of a process, p pairs of shared variables and local variables are randomly chosen among the shared variables and the local variables, such that the first element of such a pair is assigned the negation of the second element of the pair. Transitions are numbered from 0 to $t - 1$, and are executed consecutively, and when the end of the sequence of the transitions is reached, it loops back to the execution of the transition numbered 0.

Types of Properties. The properties are specified by a subset of 24 eCTL formulas (which are actually all CTL properties, since BDD-based CTL model checking in NuSMV is used as the reference in the evaluation). These properties involve AG, AF properties, and more complicated ones specified with different combinations of operators with one or two levels of nesting (with two levels of nesting when AX or EX is involved). Properties p_{01} to p_{12} are shown below, where v_i are global variables.

p_{01} :	$AG(\bigvee_{i=1}^{c} v_i)$	p_{07} :	$A(v_1\ U\ A(v_2\ U\ \bigvee_{i=3}^{c} v_i)$
p_{02} :	$AF(\bigvee_{i=1}^{c} v_i)$	p_{08} :	$A(v_1\ U\ E(v_2\ U\ \bigvee_{i=3}^{c} v_i)$
p_{03} :	$AG(v_1 \rightarrow AF(v_2 \wedge \bigvee_{i=3}^{c} v_i))$	p_{09} :	$A(v_1\ U\ A(v_2\ R\ \bigvee_{i=3}^{c} v_i)$
p_{04} :	$AG(v_1 \rightarrow EF(v_2 \wedge \bigvee_{i=3}^{c} v_i))$	p_{10} :	$A(v_1\ U\ E(v_2\ R\ \bigvee_{i=3}^{c} v_i)$
p_{05} :	$EG(v_1 \rightarrow AF(v_2 \wedge \bigvee_{i=3}^{c} v_i))$	p_{11} :	$A(AXv_1\ R\ AX\ A(v_2\ U\ \bigvee_{i=3}^{c} v_i)$
p_{06} :	$EG(v_1 \rightarrow EF(v_2 \wedge \bigvee_{i=3}^{c} v_i))$	p_{12} :	$A(EXv_1\ R\ EX\ E(v_2\ U\ \bigvee_{i=3}^{c} v_i)$

Properties p_{13} to p_{24} are similar to p_{01} to p_{12} where the difference is that $\wedge$ and $\bigvee$ are replaced by respectively $\vee$ and $\bigwedge$.

Experimental Setup. The comparison of advantage and disadvantage is based on the time used for the verification problem instances. The experimental data were obtained by running the tools on a Linux platform. For QBF-based verification, the following command is used for running VERDS-QBF.

```
verds -QBF filename
```

For BDD-based verification, we run NuSMV (without counter-example generation) by the following command.

```
NuSMV -dcx filename
```

The option -dcx is for avoiding the generation of counter-examples. This option is used, since the corresponding use of VERDS does not generate counter-examples.

Experimental Data for Programs with Concurrent Processes. For this type of programs, we test different sizes of the programs with 3 processes ($a = 3$), and let b vary over the set of values $\{12, 24, 36\}$, then set $c = b/2$, $d = c/a$. Each of the 24 properties is tested on 20 test cases for each value of b. For brevity, for each type of properties, a summary of the experimental data is presented in the left part of Fig. 1, where N is the number of test cases, T is the number of test cases in which the property is true, F is the number of test cases in which the property is false, *adv* is the number of cases in which VERDS-QBF has an advantage with respect to the usage of time. In this part of the evaluation, VERDS-QBF has advantage in 1190 of 1440 test cases. On the relative advantage of verification and falsification, VERDS-QBF has better advantage in the case of falsification.

Experimental Data for Programs with Concurrent Sequential Processes. For this type of programs, we test different sizes of the programs with 2 processes ($a = 2$), and let b vary over the set of values $\{12, 16, 20\}$, and then set $c = b/2$, $d = c/a$, $t = c$, and $p = 4$. Similarly, each property is tested on 20 test cases for each value of b, and a summary of the experimental data is presented in the right part of Fig. 1. In this part of the evaluation, VERDS-QBF has advantage in 739 of 1440 test cases. On the relative advantage of verification and falsification, VERDS-QBF has also better advantage in falsification in this part of the evaluation.

Summary. Based on the total of 2880 test cases[3], the experimental evaluation[4] shows that the QBF-based verification does not have advantage in verifying any of the properties that start with *AG*. On the other hand, the QBF-based verification may have advantages in parts (ranging from a few percent to a large percent) of the test cases of other types of verification and falsification problems (including falsification of AG properties). On the relative advantage of verification and falsification, VERDS-QBF has better advantage in falsification in both

[3] Available at `http://lcs.ios.ac.cn/~zwh/tr/verds130ee.rar`
[4] Details available at `http://lcs.ios.ac.cn/~zwh/tr/verds130eeq.pdf`

Data for Concurrent Processes:

property	adv/T	adv/F	adv/N
p_{01}	-	60/60	60/60
p_{02}	60/60	-	60/60
p_{03}	0/3	43/57	43/60
p_{04}	0/60	-	0/60
p_{05}	46/53	1/7	47/60
p_{06}	53/60	-	53/60
p_{07}	60/60	-	60/60
p_{08}	60/60	-	60/60
p_{09}	50/52	5/8	55/60
p_{10}	60/60	-	60/60
p_{11}	13/13	45/47	58/60
p_{12}	60/60	-	60/60
p_{13}	-	60/60	60/60
p_{14}	3/3	55/57	58/60
p_{15}	0/8	38/52	38/60
p_{16}	0/60	-	0/60
p_{17}	46/56	1/4	47/60
p_{18}	53/60	-	53/60
p_{19}	5/5	54/55	59/60
p_{20}	16/21	30/39	46/60
p_{21}	3/3	56/57	59/60
p_{22}	3/3	56/57	59/60
p_{23}	-	60/60	60/60
p_{24}	24/31	11/29	35/60
sum	615/791	575/649	1190/1440

Data for Concurrent Seq. Processes:

property	adv/T	adv/F	adv/N
p_{01}	0/53	2/7	2/60
p_{02}	60/60	-	60/60
p_{03}	0/10	1/50	1/60
p_{04}	0/60	-	0/60
p_{05}	0/46	0/14	0/60
p_{06}	0/60	-	0/60
p_{07}	60/60	-	60/60
p_{08}	60/60	-	60/60
p_{09}	36/54	0/6	36/60
p_{10}	53/60	-	53/60
p_{11}	33/47	0/13	33/60
p_{12}	52/60	-	52/60
p_{13}	-	60/60	60/60
p_{14}	4/4	4/56	8/60
p_{15}	0/10	1/50	1/60
p_{16}	0/60	-	0/60
p_{17}	0/48	0/12	0/60
p_{18}	0/60	-	0/60
p_{19}	8/8	52/52	60/60
p_{20}	13/18	27/42	40/60
p_{21}	4/4	54/56	58/60
p_{22}	4/4	55/56	59/60
p_{23}	-	59/60	59/60
p_{24}	8/16	29/44	37/60
sum	395/862	344/578	739/1440

Fig. 1. Experimental Data for the two Types of Programs

of the types of programs. In summary, QBF-based verification has advantage in more than 50 percent of the test cases, which are well distributed among verification and falsification of universal properties.

6 Concluding Remarks

Bounded semantics of eCTL and QBF-based characterization of eCTL based on such a semantics have been presented. A verification algorithm of eCTL properties based on solving QBF-formulas has then been established.

The traditional application area of SAT-based verification has mainly been on the error detection of various universal properties such as LTL and the universal fragments of CTL* [1,14,15]. QBF-based verification presented in this paper applies to the set of eCTL properties (that may be specified with both universal and existential path quantifiers), and can handle verification and falsification problems with bounded models. Furthermore, one of the particular aspects of this implementation is that it can handle properties, for instance, of the form

$A^{\infty}G(p \rightarrow EFq)$, that are not handled by well known model checking tools such as Spin [11] and NuSMV [4].

Experimental evaluation of such an approach has been presented. The test cases have shown that QBF-based verification and BDD-based verification have their own advantages and may be considered complementary in the verification of different problem instances.

The efficiency of QBF-based verification depends very much on the QBF-solving techniques. External QBF-solvers may be used to increase the efficiency of the verification. Improving the efficiency by optimizing the QBF-based encoding and by enhancing QBF-solving techniques remains as future works.

References

1. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic Model Checking without BDDs. In: Cleaveland, W.R. (ed.) TACAS/ETAPS 1999. LNCS, vol. 1579, pp. 193–207. Springer, Heidelberg (1999)
2. Biere, A., Cimmatti, A., Clarke, E., Strichman, O., Zhu, Y.: Bounded Model Checking. Advances in Computers, vol. 58. Academic Press (2003)
3. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, J.: Symbolic model checking: 10^{20} states and beyond. LICS, pp. 428–439 (1990)
4. Cimatti, A., Clarke, E.M., Giunchiglia, F., Roveri, M.: NUSMV: A New Symbolic Model Verifier. In: Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 495–499. Springer, Heidelberg (1999)
5. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. The MIT Press (1999)
6. Duan, Z., Tian, C., Yang, M., He, J.: Bounded Model Checking for Propositional Projection Temporal Logic. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 591–602. Springer, Heidelberg (2013)
7. Emerson, E.A., Clarke, E.M.: Using Branching-time Temporal Logics to Synthesize Synchronization Skeletons. Sci. of Comp. Prog. 2(3), 241–266 (1982)
8. Emerson, E.A., Halpern, J.Y.: "Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic. J. ACM 33(1), 151–178 (1986)
9. Goultiaeva, A., Van Gelder, A., Bacchus, F.: A Uniform Approach for Generating Proofs and Strategies for Both True and False QBF Formulas. In: IJCAI 2011, pp. 546–553 (2011)
10. Hoffmann, J., Gomes, C.P., Selman, B., Kautz, H.A.: SAT Encodings of State-Space Reachability Problems in Numeric Domains. In: IJCAI 2007, pp. 1918–1923 (2007)
11. Holzmann, G.J.: The model checker Spin. IEEE Transactions on Software Engineering 23(5), 279–295 (1997)
12. Kemper, S.: SAT-based verification for timed component connectors. Sci. Comput. Program. 77(7-8), 779–798 (2012)
13. Kontchakov, R., Pulina, L., Sattler, U., Schneider, T., Selmer, P., Wolter, F., Zakharyaschev, M.: Minimal Module Extraction from DL-Lite Ontologies Using QBF Solvers. In: IJCAI 2009, pp. 836–841 (2009)
14. Penczek, W., Wozna, B., Zbrzezny, A.: Bounded Model Checking for the Universal Fragment of CTL. Fundamenta Informaticae 51, 135–156 (2002)
15. Wozna, B.: ATCL* properties and Bounded Model Checking. Fundam. Inform. 63(1), 65–87 (2004)

16. McMillan, K.L.: Symbolic Model Checking. Kluwer Academic Publisher (1993)
17. Peled, D.A.: Software Reliability Methods. Springer (2001)
18. Peterson, G.L.: Myths About the Mutual Exclusion Problem. Information Processing Letters 12(3), 115–116 (1981)
19. Zhang, W.: Bounded Semantics of CTL and SAT-based Verification. In: Breitman, K., Cavalcanti, A. (eds.) ICFEM 2009. LNCS, vol. 5885, pp. 286–305. Springer, Heidelberg (2009)
20. Zhang, W.: Bounded Semantics of CTL. Institute of Software, Chinese Academy of Sciences. Technical Report ISCAS-LCS-10-16 (2010)
21. Zhang, W.: VERDS modeling language, `http://lcs.ios.ac.cn/~zwh/verds/`