SAT-based Verification of LTL Formulas

Wenhui Zhang *

Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
zwh@ios.ac.cn

Abstract. Bounded model checking (BMC) based on satisfiability testing (SAT) has been introduced as a complementary technique to BDD-based symbolic model checking of LTL properties in recent years and a lot of successful work has been done with this approach. The basic idea is to search for a counter example of a particular length and to generate a propositional formula that is satisfied iff such a counter example exists. An over approximation of the length that need to be checked in order to certify that the system is error free is usually too big, such that it is not practical to use this approach for checking systems that are error free with respect to given properties. Even if we know the exact threshold, for a reasonably large system, this threshold would possibly also be large enough to make the verification become intractable due to the complexity of solving the corresponding SAT instance. This study is on a different direction and the aim of this study is verification of valid properties. We propose an approach to (partly) avoid the use of the completeness threshold as the verification criteria when checking systems that are error free with respect to LTL properties. The benefit of the use of this approach may be very large compared to the use of the completeness threshold. Though, Prasad, Biere and Gupta pointed out in a survey paper [18] that, currently, the strength of SAT-based verification techniques lies primarily in falsification, this study explores the strength of SAT-based techniques for verification and the case study shows that this is a promising approach.

1 Introduction

Model checking has been successfully used in the last decade for the formal verification of finite state systems. It is considered as one of the most practical applications of theoretical computer science in the verification of concurrent systems. However the practical applicability of model checking 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. The methods include application of cone of influence reduction [1], semantic minimization [19], state

* Supported by the National Natural Science Foundation of China under Grant No. 60373050, 60421001 and 60573012, and the National Grand Fundamental Research 973 Program of China under Grant No. 2002cb312200.

information compression [10], abstraction techniques [6, 13], partial order reductions [21, 22], symmetry reductions [9], compositional techniques for splitting verification tasks [8, 1], case-based partition techniques [14, 23], and BDD based symbolic techniques for compactly representing transition relations and system states [5, 4].

Bounded model checking (BMC) based on satisfiability testing (SAT) has been introduced as a complementary technique to BDD-based symbolic model checking of LTL properties [3]. A lot of successful work has been done with this approach [2, 18]. The basic idea is to search for a counter example of a particular length and to generate a propositional formula that is satisfied iff such a counter example exists. The efficiency of this method is based on the observation that if a system is faulty then only a fragment of its state space is sufficient for finding an error. Given a finite transition system M, an LTL formula φ and a natural number k, a BMC procedure decides whether there exists a computation in M of length k or less that violates φ. SAT based BMC is performed by generating a propositional formula which is satisfiable if and only if such a path exists. BMC is conducted in an iterative process where k is incremented until either (i) an error is found, (ii) the problem becomes intractable due to the complexity of solving the corresponding SAT instance, or (iii) k reaches some pre-computed completeness threshold which indicates that M satisfies φ. If we have given M and φ such that M satisfies φ, then the practical value of this approach depends on the existence of a relatively small value of the completeness threshold. Computing an exact value of the completeness threshold for a given model and formula is difficult. A general over approximation of the completeness threshold is $|M| \cdot 2^{|\varphi|}$ where $|M|$ is the size of the model and $|\varphi|$ is the size of the formula. This approximation is obviously impractical for checking systems that are error free with respect to given properties. For reducing this approximation, completeness threshold has been studied for several types of LTL formulas [12, 7].

As stated in [12], knowing the completeness threshold is essential for making BMC complete. Without it, there is no way of knowing whether the property holds or rather the bound is not sufficiently high. Even if we know the completeness threshold, for a reasonably large system, this threshold would possibly be large enough to make the verification become intractable due to the complexity of solving the corresponding SAT instance. This study is on a different direction that proposes an approach that (partly) avoids this problem and may prove whether the property holds without knowing a completeness threshold. This kind of research has also been considered in [2] for simple liveness properties of the form Fp. There is also a lot of work on proving safety properties based on SAT, the related works are for instance, proving safety properties by using induction [20, 15], conservative abstraction with counter example guided refinement [**?**], and interpolation based transition relation approximation for generating facts relevant with respect to given properties [11]. In this work, we study LTL properties in general.

2 Propositional Linear Temporal Logic

Propositional linear temporal logic (LTL) is a logic introduced by Pnueli as a specification language for concurrent programs [17]. Let AP be a set of proposition symbols. The set of LTL formulas is defined as follows:

- Every member of AP is an LTL formula.
- Logical connectives of LTL include: $\neg$, $\wedge$, $\vee$, and $\rightarrow$.
  If φ and ψ are LTL formulas, then so are $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, and $\varphi \rightarrow \psi$.
- Temporal operators include: X, F, G, U, and R.
  If φ and ψ are LTL formulas, then so are: $X\,\varphi$, $F\,\varphi$, $G\,\varphi$, $\varphi\,U\,\psi$, and $\varphi\,R\,\psi$.

2.1 Semantics of LTL

The formal semantics of LTL is defined with respect to paths of a Kripke structure. Let $M = \langle S, T, I, L\rangle$ be a Kripke structure 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. Let φ be a temporal formula. Let $\pi = \pi_0\pi_1 \cdots$ be a path of M and π^i be the subpath of π starting at π_i. We define the relation φ holds on π, denoted $\pi \models \varphi$, as follows.

$$
\begin{array}{ll}
\pi \models p & \text{iff } p \in L(\pi_0) \\
\pi \models \neg\varphi & \text{iff } \pi \not\models \varphi \\
\pi \models \varphi \wedge \psi & \text{iff } \pi \models \varphi \text{ and } \pi \models \psi \\
\pi \models \varphi \vee \psi & \text{iff } \pi \models \varphi \text{ or } \pi \models \psi \\
\pi \models \varphi \rightarrow \psi & \text{iff } \pi \models \varphi \text{ implies } \pi \models \psi \\
\pi \models X\varphi & \text{iff } \pi^1 \models \varphi \\
\pi \models F\varphi & \text{iff } \exists k \geq 0.\pi^k \models \varphi \\
\pi \models G\varphi & \text{iff } \forall k \geq 0.\pi^k \models \varphi \\
\pi \models \varphi U \psi & \text{iff } \exists k \geq 0.\forall j < k.(\pi^k \models \psi \wedge \pi^j \models \varphi) \\
\pi \models \varphi R \psi & \text{iff } \forall j \geq 0.(\pi^j \models \psi) \vee \exists k \geq 0.((\pi^k \models \varphi) \wedge (\forall j \leq k.(\pi^j \models \psi))
\end{array}
$$

For simplicity, we call a Kripke structure a model. An LTL formula φ is *true* in the model M, denoted $M \models \varphi$, iff φ is *true* on all paths starting from an arbitrary initial state of M.

2.2 Bounded Semantics of LTL Formulas in NNF

An LTL formula is in negation normal form (NNF), if the symbol $\rightarrow$ does not appear in the formula and $\neg$ is applied only to proposition symbols. Every formula can be transformed into a formula in NNF by using the following rules:

$$
\begin{array}{llll}
\varphi \rightarrow \psi & = \neg\varphi \vee \psi & \neg(\varphi \wedge \psi) & = (\neg\varphi \vee \neg\psi) \\
\neg(\varphi \vee \psi) & = (\neg\varphi \wedge \neg\psi) & \neg X\varphi & = X\neg\varphi \\
\neg F\varphi & = G\neg\varphi & \neg G\varphi & = F\neg\varphi \\
\neg(\varphi U \psi) & = \neg\varphi R \neg\psi & \neg(\varphi R \psi) & = \neg\varphi U \neg\psi
\end{array}
$$

In the following, we only consider LTL formulas in NNF. Let $M = \langle S, T, I, L\rangle$ be a model and $k \in \mathbf{N}$. Let $\pi = \pi_0\pi_1 \cdots$ be an infinite path of M. If $u = \pi_0 \cdots \pi_k$ and $v = \pi_l \cdots \pi_k$ for some $0 \leq l \leq k$, we call $\pi = u \cdot v^\omega$ a (k, l)-loop. If π is a (k, l)-loop for some $0 \leq l \leq k$, we call π a k-loop.

Definition 1 (Bounded Semantics for a Loop). *Let $k \geq 0$ and π be a k-loop. Then an LTL formula φ is valid on π with bound k, written $\pi \models_k \varphi$, iff $\pi \models \varphi$.*

Definition 2 (Bounded Semantics without a Loop). *Let $k \geq 0$ and π be a path which is not a k-loop. Then an LTL formula φ is valid on π with bound k, written $\pi \models_k \varphi$, iff $\pi \models_k^0 \varphi$ where:*

$$
\begin{array}{ll}
\pi \models_k^i p & \text{iff } p \in L(\pi_i) \\
\pi \models_k^i \neg p & \text{iff } \pi \not\models_k^i p \\
\pi \models_k^i \varphi \wedge \psi & \text{iff } \pi \models_k^i \varphi \text{ and } \pi \models_k^i \psi \\
\pi \models_k^i \varphi \vee \psi & \text{iff } \pi \models_k^i \varphi \text{ or } \pi \models_k^i \psi \\
\pi \models_k^i X\varphi & \text{iff } i < k \text{ and } \pi \models_k^{i+1} \varphi \\
\pi \models_k^i F\varphi & \text{iff } \exists j \in \{i, ..., k\}.\pi \models_k^j \varphi \\
\pi \models_k^i G\varphi & \text{iff } \mathit{false}. \\
\pi \models_k^i \varphi U \psi & \text{iff } \exists j \in \{i, ..., k\}.\forall n \in \{i, ..., j-1\}.(\pi \models_k^j \psi \wedge \pi \models_k^n \varphi) \\
\pi \models_k^i \varphi R \psi & \text{iff } \exists j \in \{i, ..., k\}.((\pi \models_k^j \varphi) \wedge \forall n \in \{i, ..., j\}.(\pi \models_k^n \psi))
\end{array}
$$

Note that $\pi \models_k^i G\varphi$ is *false* by definition. This is explained by that a global property can only be witnessed by an infinite path (or a path with a loop).

Theorem 1. *Let M be a model, φ an LTL formula. Then $M \not\models \varphi$ iff there is a path π and a $k \geq 0$ such that $\pi \models_k \neg\varphi$.*

3 Encoding the Model in SAT-Formulas

Since we have $F\varphi = \mathit{true}\, U\varphi$ and $\varphi R\psi = (\psi U(\varphi \wedge \psi)) \vee G\psi$, we only consider formulas of the form $\varphi \vee \psi, \varphi \wedge \psi, X\varphi, G\varphi, \varphi U\psi$ constructed from propositions and the negation of propositions.

Given a model M, an LTL formula φ and a bound k, we will construct encodings for the pair (M, φ). Let $u_0, ..., u_k$ be a finite sequence of states on a path π. We first define $[[M]]_k$ to be a formula representing that $u_0 \cdots u_k$ is a finite prefix of a valid path of M.

Definition 3 (Transition Relation). *Let $M = \langle S, T, I, L\rangle$ be a model and $k \geq 0$.*

$$[[M]]_k := I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1})$$

This translation of transition relation corresponds to that in [3]. Let $M = \langle S, T, I, L\rangle$ be a model. Let u, w (possibly with subscripts) represent individual states. 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. For a state and a formula, we first present the encoding for (formula,state) pair as done in [3] (however with a slightly different version). Then we propose an encoding for (formula,state) pair for the purpose of verification.

3.1 Encoding of LTL Formulas

Let min() be the minimum operation and $s(i,k,l)$ denote

$$\text{if } (k = i) \text{ then } l \text{ else } i + 1.$$

Definition 4 (Translation of LTL formulas). *Given a state $u \in \{u_0, ..., u_k\}$ and a formula φ, the encoding is denoted by $[[\varphi, u]]_{k,l}$.*

$$\begin{array}{ll}
[[p,u]]_{k,l} & = p(u) \\
[[\neg p,u]]_{k,l} & = \neg p(u) \\
[[\varphi \vee \psi,u]]_{k,l} & = [[\varphi,u]]_{k,l} \vee [[\psi,u]]_{k,l} \\
[[\varphi \wedge \psi,u]]_{k,l} & = [[\varphi,u]]_{k,l} \wedge [[\psi,u]]_{k,l} \\
\hline
[[X\varphi,u_i]]_{k,l} & = [[\varphi,u_{s(i,k,l)}]]_{k,l} \\
[[G\varphi,u_i]]_{k,l} & = \bigwedge_{j=\min(i,l)}^{k} [[\varphi,u_j]]_{k,l} \\
[[\varphi U \psi,u_i]]_{k,l} & = \bigvee_{j=i}^{k}([[\psi,u_j]]_{k,l} \wedge \bigwedge_{t=i}^{j-1}[[\varphi,u_t]]_{k,l}) \vee \\
 & \bigwedge_{t=i}^{k}[[\varphi,u_t]]_{k,l} \wedge \bigvee_{j=l}^{i-1}([[\psi,u_j]]_{k,l} \wedge \bigwedge_{t=l}^{j-1}[[\varphi,u_t]]_{k,l})
\end{array}$$

where $[[\varphi, u_{-1}]]_{k,l} = \mathit{false}$.

In the above definition, u_{-1} is a special symbol used only for the purpose of uniform formula representation (avoiding specification of different cases explicitly). In the real transformation, formulas containing this symbol are to be replaced by *true* or *false* according to their meaning, for instance, $[[p \vee q, u_{-1}]]_{k,l}$ must be replaced by *false* and not by $[[p, u_{-1}]]_{k,l} \vee [[q, u_{-1}]]_{k,l}$. In addition, we define $T(u_k, u_{-1}) = \mathit{true}$. The subscript (k,l) in the definition indicates that the path is a (k,l)-loop for $l \geq 0$, otherwise the path is considered loop free.

Definition 5. $[[M,\varphi]]_k := [[M]]_k \wedge \bigvee_{l=-1}^{k}(T(u_k,u_l) \wedge [[\varphi,u_0]]_{k,l})$

The encoding of $[[M,\varphi]]_k$ corresponds to that in [3] with some modification, i.e. a condition $\bigwedge_{l=0}^{k} \neg T(u_k,u_l)$ representing loop-free-ness is removed (or more precisely, replaced by *true*)[1]. This change does not affect the satisfiability of the formula. This fact is to be established and presented as Theorem 2.

Lemma 1. $[[\varphi,u_0]]_{k,-1} \rightarrow [[\varphi,u_0]]_{k,l}$ for $l \in \{0,...,k\}$.

[1] This is not only a matter of representational simplicity. With this clause in the formulation, we would not be able to prove Lemma 3 and then the proof of Theorem 5 would be different and more complicated.

Proof: We prove a more general property

$$[[\varphi, u_i]]_{k,-1} \rightarrow [[\varphi, u_i]]_{k,l} \text{ for } i \in \{0, ..., k\} \text{ and } l \in \{0, ..., k\}$$

by structural induction. The case is trivial for φ being a proposition or negation of a proposition. Assume the induction hypothesis.

- The case is trivial for φ being a conjunctive or disjunctive formula.
- If $\varphi = X\varphi_0$, then
  $[[\varphi, u_i]]_{k,-1}$ is either *false* $(i = k)$ or the same as $[[\varphi_0, u_{i+1}]]_{k,-1}$ $(i < k)$.
  In the latter case, $[[\varphi, u_i]]_{k,l} = [[\varphi_0, u_{i+1}]]_{k,l}$.
  Therefore, according to the induction hypothesis, $[[\varphi, u_i]]_{k,-1} \rightarrow [[\varphi, u_i]]_{k,l}$.
- If $\varphi = G\varphi_0$, then $[[\varphi, u_i]]_{k,-1}$ is *false*. Therefore $[[\varphi, u_i]]_{k,-1} \rightarrow [[\varphi, u_i]]_{k,l}$.
- If $\varphi = \varphi_0 U \varphi_1$, then $\bigvee_{j=-1}^{i-1}([[\varphi_1, u_j]]_{k,-1} \wedge \bigwedge_{t=-1}^{j-1}[[\varphi_0, u_t]]_{k,-1}) = \mathit{false}$.
  Therefore $[[\varphi, u_i]]_{k,-1} = \bigvee_{j=i}^{k}([[\varphi_1, u_j]]_{k,-1} \wedge \bigwedge_{t=i}^{j-1}[[\varphi_0, u_t]]_{k,-1})$.
  Then, according to the induction hypothesis,
  $[[\varphi, u_i]]_{k,-1} \rightarrow \bigvee_{j=i}^{k}([[\varphi_1, u_j]]_{k,l} \wedge \bigwedge_{t=i}^{j-1}[[\varphi_0, u_t]]_{k,l})$.
  Since the right side of the implication is a disjunctive part of $[[\varphi, u_i]]_{k,l}$, we obtain $[[\varphi, u_i]]_{k,-1} \rightarrow [[\varphi, u_i]]_{k,l}$. □

Theorem 2. *Let M be a model, φ be an LTL formula. Let $k \geq 0$. There is a path π of M such that $\pi \models_k \varphi$ iff $[[M, \varphi]]_k$ is satisfiable.*

Theorem 2 corresponds to the normal soundness theorem of bounded LTL model checking [3]. As explained, the only different in the encoding $[[M, \varphi]]_k$ and that in [3] is that a condition representing loop-free-ness is removed. The fact that this change does not affect the satisfiability of the formula can be proved easily based on Lemma 1.

3.2 Encoding of LTL formulas for Verification

Definition 6 (Translation of LTL formulas for Verification). *Given a state $u \in \{u_0, ..., u_k\}$ and a formula φ, the encoding is denoted by $[[\varphi, u]]_k^v$.*

$[[p, u]]_k^v$	$= p(u)$
$[[\neg p, u]]_k^v$	$= \neg p(u)$
$[[\varphi \vee \psi, u]]_k^v$	$= [[\varphi, u]]_k^v \vee [[\psi, u]]_k^v$
$[[\varphi \wedge \psi, u]]_k^v$	$= [[\varphi, u]]_k^v \wedge [[\psi, u]]_k^v$
$[[X\varphi, u_i]]_k^v$	$= [[\varphi, u_{i+1}]]_k^v$
$[[G\varphi, u_i]]_k^v$	$= \bigwedge_{j=i}^{k}[[\varphi, u_j]]_k^v$
$[[\varphi U \psi, u_i]]_k^v$	$= \bigvee_{j=i}^{k}([[\psi, u_j]]_k^v \wedge \bigwedge_{t=i}^{j-1}[[\varphi, u_t]]_k^v) \vee \bigwedge_{t=i}^{k}[[\varphi, u_t]]_k^v$

where $[[\varphi, u_{k+1}]]_k^v = \mathit{true}$.

Definition 7. $[[M, \varphi]]_k^v := [[M]]_k \wedge [[\varphi, u_0]]_k^v$

In the following, we shall establish that there is no path π and $k \geq 0$ such that $\pi \models_k \varphi$ if there is some i such that $[[M, \varphi]]_i^v$ is unsatisfiable.

Definition 8. $[[M, \varphi, u_i]]_k^v := [[M]]_k \wedge [[\varphi, u_i]]_k^v$.

Proposition 1. *For all $i \in \{0, ..., k\}$, the following equivalences holds:*

(1) $[[M, \varphi_0 \vee \varphi_1, u_i]]_k^v = [[M, \varphi_0, u_i]]_k^v \vee [[M, \varphi_1, u_i]]_k^v$
(2) $[[M, \varphi_0 \wedge \varphi_1, u_i]]_k^v = [[M, \varphi_0, u_i]]_k^v \wedge [[M, \varphi_1, u_i]]_k^v$
(3) $[[M, X\varphi, u_i]]_k^v = [[M, \varphi, u_{i+1}]]_k^v$
(4) $[[M, G\varphi, u_i]]_k^v = \bigwedge_{j=i}^{k} [[M, \varphi, u_j]]_k^v$
(5) $[[M, \varphi U \psi, u_i]]_k^v = \bigvee_{j=i}^{k} ([[M, \psi, u_j]]_k^v \wedge \bigwedge_{t=i}^{j-1} [[M, \varphi, u_t]]_k^v) \vee \bigwedge_{t=i}^{k} [[M, \varphi, u_t]]_k^v$

Proof: We only prove the first equivalence. The others are similar. We have

$$\begin{aligned} & [[M, \varphi_0 \vee \varphi_1, u_i]]_k^v \\ = \; & [[M]]_k \wedge [[\varphi_0 \vee \varphi_1, u_i]]_k^v \\ = \; & [[M]]_k \wedge ([[\varphi_0, u_i]]_k^v \vee [[\varphi_1, u_i]]_k^v) \\ = \; & [[M, \varphi_0, u_i]]_k^v \vee [[M, \varphi_1, u_i]]_k^v \end{aligned}$$

This was what needed to be proved.

Lemma 2. $[[M, \varphi, u_i]]_{k+1}^v \rightarrow [[M, \varphi, u_i]]_k^v$ *for* $i \in \{0, ..., k\}$.

Proof: This can be proved based on structural induction on φ. The proof is straightforward and is omitted.

Theorem 3. $[[M, \varphi]]_{k+1}^v \rightarrow [[M, \varphi]]_k^v$.

Proof: This follows directly from Lemma 2.

Theorem 4. $[[M, \varphi]]_k \rightarrow [[M, \varphi]]_k^v$.

Proof: Since $[[M, \varphi]]_k = [[M]]_k \wedge \bigvee_{l=-1}^{k} (T(u_k, u_l) \wedge [[\varphi, u_0]]_{k,l})$ and $[[M, \varphi]]_k^v = [[M]]_k \wedge [[\varphi, u_0]]_k^v$, it is sufficient to prove that $[[\varphi, u_0]]_{k,l} \rightarrow [[\varphi, u_0]]_k^v$. We prove

$$[[\varphi, u_i]]_{k,l} \rightarrow [[\varphi, u_i]]_k^v \text{ for } i \in \{0, ..., k\} \text{ and } l \in \{0, ..., k\}$$

by structural induction. The case is trivial for φ being a proposition or negation of a proposition. Assume the induction hypothesis.

- The case is trivial for φ being a conjunctive or disjunctive formula.
- If $\varphi = X\varphi_0$, we have two cases.
  For $i = k$, we have $[[\varphi, u_i]]_k^v = \mathit{true}$. Therefore $[[\varphi, u_i]]_{k,l} \rightarrow [[\varphi, u_i]]_k^v$.
  For $i < k$, we have $[[\varphi, u_i]]_{k,l} = [[\varphi_0, u_{i+1}]]_{k,l}$ and $[[\varphi, u_i]]_k^v = [[\varphi_0, u_{i+1}]]_k^v$.
  Therefore, according to the induction hypothesis, $[[\varphi, u_i]]_{k,l} \rightarrow [[\varphi, u_i]]_k^v$.
- If $\varphi = G\varphi_0$, then
  $[[\varphi, u_i]]_{k,l} = \bigwedge_{j=i}^{k} [[\varphi_0, u_j]]_{k,l} \wedge \bigwedge_{j=\min(i,l)}^{i-1} [[\varphi_0, u_j]]_{k,l}$
  $[[\varphi, u_i]]_k^v = \bigwedge_{j=i}^{k} [[\varphi_0, u_j]]_k^v$.
  Therefore, according to the induction hypothesis, $[[\varphi, u_i]]_{k,l} \rightarrow [[\varphi, u_i]]_k^v$.

– If $\varphi = \varphi_0 U \varphi_1$, then
$[[\varphi_0 U \varphi_1, u_i]]_{k,l}$
$= \bigvee_{j=i}^{k}([[\varphi_1, u_j]]_{k,l} \wedge \bigwedge_{t=i}^{j-1}[[\varphi_0, u_t]]_{k,l}) \vee$
$\bigwedge_{t=i}^{k}[[\varphi_0, u_t]]_{k,l} \wedge \bigvee_{j=l}^{i-1}([[\varphi_1, u_j]]_{k,l}^{m} \wedge \bigwedge_{t=l}^{j-1}[[\varphi_0, u_t]]_{k,l})$
$[[\varphi_0 U \varphi_1, u_i]]_k^v = \bigvee_{j=i}^{k}([[\varphi_1, u_j]]_k^v \wedge \bigwedge_{t=i}^{j-1}[[\varphi_0, u_t]]_k^v) \vee \bigwedge_{t=i}^{k}[[\varphi_0, u_t]]_k^v$.
Therefore, according to the induction hypothesis, $[[\varphi, u_i]]_{k,l} \rightarrow [[\varphi, u_i]]_k^v$.

Lemma 3. *If* $[[M]]_k \wedge [[\varphi, u_0]]_{k,-1}$ *is satisfiable, then* $[[M]]_{k+1} \wedge [[\varphi, u_0]]_{k+1,-1}$ *is satisfiable.*

Proof: Let $u_0, ..., u_k$ be a set of states (each represented by a set of literals) that satisfy $[[M]]_k \wedge [[\varphi, u_0]]_{k,-1}$. Since the transition relation in M is total, there is a state u_{k+1} such that $T(u_k, u_{k+1})$. We prove that $u_0, ..., u_k, u_{k+1}$ is a set of states that satisfies $[[M]]_{k+1} \wedge [[\varphi, u_0]]_{k+1,-1}$. Since $[[M]]_k \wedge [[\varphi, u_0]]_{k,-1}$ and $T(u_k, u_{k+1})$, then $[[M]]_{k+1} = [[M]]_k \wedge T(u_k, u_{k+1})$ is *true*. Then it is sufficient to prove that

$$[[\varphi, u_i]]_{k,-1} \rightarrow [[\varphi, u_i]]_{k+1,-1} \text{ for all } i \in \{0, ..., k\}.$$

This can then be proved based on structural induction. The proof is omitted.

Lemma 4. *Let* l *be non-negative. If* $[[M]]_k \wedge T(u_k, u_l) \wedge [[\varphi, u_0]]_{k,l}$ *is satisfiable, then* $[[M]]_{k+1} \wedge T(u_{k+1}, u_{l+1}) \wedge [[\varphi, u_0]]_{k+1,l+1}$ *is satisfiable.*

Proof: Let $u_0, ..., u_k$ be a set of states that satisfy $[[M]]_k \wedge T(u_k, u_l) \wedge [[\varphi, u_0]]_{k,l}$. Let $u_{k+1} = u_l$. We prove that $u_0, ..., u_k, u_{k+1}$ is a set of states that satisfies $[[M]]_{k+1} \wedge T(u_{k+1}, u_{l+1}) \wedge [[\varphi, u_0]]_{k+1,l+1}$. We have

$$\begin{aligned} & [[M]]_{k+1} \wedge T(u_{k+1}, u_{l+1}) \\ = & [[M]]_k \wedge T(u_k, u_{k+1}) \wedge T(u_{k+1}, u_{l+1}) \\ = & [[M]]_k \wedge T(u_k, u_l) \wedge T(u_l, u_{l+1}) \\ = & [[M]]_k \wedge T(u_k, u_l) \end{aligned}$$

Since $[[M]]_k \wedge T(u_k, u_l) \wedge [[\varphi, u_0]]_{k,l}$, then $[[M]]_{k+1} \wedge T(u_{k+1}, u_{l+1})$ is *true*. Then it is sufficient to prove that

$$[[\varphi, u_i]]_{k,l} \rightarrow [[\varphi, u_i]]_{k+1,l+1} \text{ for all } i \in \{0, ..., k\}.$$

This can then be proved based on structural induction. The proof is omitted.

Theorem 5. *If* $[[M, \varphi]]_k$ *is satisfiable, then* $[[M, \varphi]]_{k+1}$ *is satisfiable.*

Proof: Suppose that $[[M, \varphi]]_k = [[M]]_k \wedge \bigvee_{l=-1}^{k}(T(u_k, u_l) \wedge [[\varphi, u_0]]_{k,l})$ is *true*. Then $[[M]]_k \wedge (T(u_k, u_l) \wedge [[\varphi, u_0]]_{k,l})$ is *true* for some $l \in \{-1, 0, ..., k\}$. There are two cases $l = -1$ and $l \in \{0, ..., k\}$. In the former case, Lemma 3 implies that $[[M]]_{k+1} \wedge (T(u_{k+1}, u_{-1}) \wedge [[\varphi, u_0]]_{k+1,-1})$ is satisfiable. Therefore $[[M, \varphi]]_{k+1}$ is satisfiable. In the latter case, Lemma 4 implies that $[[M]]_{k+1} \wedge (T(u_{k+1}, u_{l+1}) \wedge [[\varphi, u_0]]_{k+1,l+1})$ is satisfiable. Therefore $[[M, \varphi]]_{k+1}$ is satisfiable also in this case.

Theorem 6. *If $[[M, \varphi]]_k^v$ is unsatisfiable for some k, then $M \models \neg\varphi$.*

Proof: Suppose that $M \models \neg\varphi$ does not hold. Then there is a path π of M and a $k' \geq 0$ such that $\pi \models_{k'} \varphi$ according to Theorem 1. Then $[[M, \varphi]]_{k'}$ is satisfiable according to Theorem 2. Then $[[M, \varphi]]_n$ is satisfiable for $n \geq k'$ according to Theorem 5. Then $[[M, \varphi]]_n^v$ is satisfiable for $n \geq k'$ according to Theorem 4. Choose n' such that $n' \geq k$ and $n' \geq k'$. Then $[[M, \varphi]]_{n'}^v$ is satisfiable. Then $[[M, \varphi]]_{k''}^v$ is satisfiable for all $n' \geq k''$ according to Theorem 3. This contradicts with that $[[M, \varphi]]_k^v$ is unsatisfiable, since $n' \geq k$. This proves the theorem.

4 SAT-based Verification

Theorem 6 provides a theoretical basis for verification and Theorem 2 provides a theoretical basis for error detection. The theorems suggest the following combination of verification and error detection approach. Let M be a model and φ be a temporal formula to be verified.

- Start with $k = 0$;
- If $[[M, \neg\varphi]]_k^v$ is unsatisfiable, report that $M \models \varphi$ is valid;
- If $[[M, \neg\varphi]]_k$ is satisfiable, report that $M \models \varphi$ does not hold;
- If a completeness threshold is reached, report that $M \models \varphi$ is valid;
- Increase k and repeat the process.

Note that $\neg\varphi$ represents the formula in NNF corresponding to $\neg\varphi$. In many cases, as will be demonstrated in the case study, the procedure may terminate before reaching a completeness threshold. However, in the general case, it may be necessary to repeat the process until a completeness threshold is reached. For instance, if we have the trivial property $\varphi = G\,true$, which is *true* for all systems, then we have $[[M, \neg\varphi]]_k = false$ and $[[M, \neg\varphi]]_k^v = I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1})$. Then the first one is unsatisfiable and the second is always satisfiable. The above approach can only terminate when a completeness threshold is reached.

This is a theoretical formulation. In practice, for a reasonably large system, the threshold would possibly be large enough to make the verification become intractable due to the complexity of solving the corresponding SAT instance, and the process will be interrupted by time or memory constraints.

In the rest of this section, we discuss some types of simple properties which can then be a background for the case-study of the use of this approach for verification in the next section.

A Safety Property: A simple safety property is of the forms pRq where p and q are propositions. For verifying this property, we need to calculate $[[M, \neg pU\neg q]]_k^v$. This formula expands to

$$I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge (\bigvee_{j=0}^{k} (\neg q(u_j) \wedge \bigwedge_{t=0}^{j-1} \neg p(u_t)) \vee \bigwedge_{t=0}^{k} \neg p(u_t))$$

Therefore $M \models pRq$ if there is a k such that

$$I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge (\bigvee_{j=0}^{k} (\neg q(u_j) \wedge \bigwedge_{t=0}^{j-1} \neg p(u_t)) \vee \bigwedge_{t=0}^{k} \neg p(u_t))$$

is unsatisfiable.

A Co-Safety Property: A simple co-safety property is of the form pUq where p and q are propositions. For verifying this property, we need to calculate $[[M, \neg pR\neg q]]_k^v$. This formula is equivalent to $[[M, (\neg qU(\neg q \wedge \neg p)) \vee G\neg q]]_k^v$ which expands to

$$I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge \bigvee_{j=0}^{k} (\neg q(u_j) \wedge \neg p(u_j) \wedge \bigwedge_{t=0}^{j-1} \neg q(u_t)) \vee \bigwedge_{t=0}^{k} \neg q(u_t))$$

Therefore $M \models pUq$ if there is a k such that

$$I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge (\bigvee_{j=0}^{k} (\neg p(u_j) \wedge \bigwedge_{t=0}^{j} \neg q(u_t)) \vee \bigwedge_{t=0}^{k} \neg q(u_t))$$

is unsatisfiable. Furthermore, we have the following lemma.

Lemma 5. *if $M \models pUq$, then there is a k such that*

$$I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge (\bigvee_{j=0}^{k} (\neg p(u_j) \wedge \bigwedge_{t=0}^{j} \neg q(u_t)) \vee \bigwedge_{t=0}^{k} \neg q(u_t))$$

is unsatisfiable.

Proof: We prove that if

$$I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge (\bigvee_{j=0}^{k} (\neg p(u_j) \wedge \bigwedge_{t=0}^{j} \neg q(u_t)) \vee \bigwedge_{t=0}^{k} \neg q(u_t))$$

is satisfiable for all k, then $M \not\models pUq$, i.e. $[[M, \neg pR\neg q]]_{k'}$ is satisfiable for some k'. We have

$$\begin{aligned}
& [[M, \neg pR\neg q]]_k \\
= \; & [[M, (\neg qU(\neg q \wedge \neg p)) \vee G\neg q]]_k \\
= \; & I(u_0) \wedge \textstyle\bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge \\
& \textstyle\bigvee_{l=-1}^{k} (T(u_k, u_l) \wedge [[(\neg qU(\neg q \wedge \neg p) \vee G\neg q), u_0]]_{k,l}) \\
= \; & I(u_0) \wedge \textstyle\bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge \\
& ([[[(\neg qU(\neg q \wedge \neg p) \vee G\neg q), u_0]]_{k,-1} \vee \\
& \textstyle\bigvee_{l=0}^{k} (T(u_k, u_l) \wedge [[(\neg qU(\neg q \wedge \neg p) \vee G\neg q), u_0]]_{k,l})) \\
= \; & I(u_0) \wedge \textstyle\bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge \\
& (\textstyle\bigvee_{j=0}^{k} (\neg q(u_j) \wedge \neg p(u_j) \wedge \bigwedge_{t=0}^{j-1} \neg q(u_t)) \vee \\
& \textstyle\bigvee_{l=0}^{k} (T(u_k, u_l) \wedge (\bigvee_{j=0}^{k} (\neg q(u_j) \wedge \neg p(u_j) \wedge \bigwedge_{t=0}^{j-1} \neg q(u_t)) \vee \bigwedge_{j=0}^{k} \neg q(u_j))) \\
= \; & I(u_0) \wedge \textstyle\bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge \\
& (\textstyle\bigvee_{j=0}^{k} (\neg p(u_j) \wedge \bigwedge_{t=0}^{j} \neg q(u_t)) \vee \bigvee_{l=0}^{k} (T(u_k, u_l) \wedge \bigwedge_{j=0}^{k} \neg q(u_j)))
\end{aligned}$$

The difference between $[[M, \neg pR\neg q]]_k^v$ and $[[M, \neg pR\neg q]]_k$ is $T(u_k, u_l)$ for some $k \geq l \geq 0$. Since the transition relation is total, there is some k and l such that $T(u_k, u_l)$ is *true*. This proves the lemma. □

Corollary 1. *$M \models pUq$ iff there is a k such that*

$$I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge (\bigvee_{j=0}^{k} (\neg p(u_j) \wedge \bigwedge_{t=0}^{j} \neg q(u_t)) \vee \bigwedge_{t=0}^{k} \neg q(u_t))$$

is unsatisfiable.

Proof: This follows from the Lemma 5 and Theorem 5.

A Liveness Property: Naturally, Fp is a special case of qUp. By simplifying the previously obtained equation, we have

$$[[M, G\neg p]]_k^v = I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge \bigwedge_{i=0}^{k} \neg p(u_i)$$

Then according to Corollary 1, $M \models Fp$ iff there is a k such that

$$I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge \bigwedge_{i=0}^{k} \neg p(u_i)$$

is unsatisfiable. Note that this is consistent with the liveness property of the form Fp considered in [2] where the translation of $M \models Fp$ is as follows:

$$[[M, Fp]]_k = I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \rightarrow \bigvee_{i=0}^{k} p(u_i)$$

Then $M \models Fp$ iff there is a k such that $[[M, Fp]]_k$ is valid.

5 A Case Study

We consider verification of properties of the form pUq and pRq of a mutual exclusion algorithm. We first present our tool for verification, and then the experimental results.

5.1 Verification Tool: VERBS

We have developed a tool called VERBS (VERification Based on Sat) based on our satisfiability checking tool BOSCH (BOolean Satisfiability CHecker)[2]. The input to the tool is as follows:

- a file containing the specification of state variables;
- a file containing the specification of initial states in CNF format;
- a file containing the specification of the transition relation in CNF format;
- a file containing the specification of the property;
- a number k representing the length of transition ($k = 0, 1, 2, ...$).

Currently the subset of LTL formulas handled by the tool is of the forms $\varphi U \psi$ and $\varphi R \psi$, where φ, ψ are propositional formulas in DNF format. The tool first converts all these information to a CNF formula and then calls BOSCH for satisfiability checking.

[2] A tool based on DPLL and similar principles as the tool for parallel execution of stochastic search procedures on reduced SAT instances [24].

5.2 Presentation of the Case

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. Let the system consist of two processes: A and B with the following specification in a first order transition system [16]:

Process A:

$$\begin{array}{ll} a = s_0 & \longrightarrow (y, t, a) := (1, 1, s_1) \\ a = s_1 \wedge (x = 0 \vee t = 0) & \longrightarrow (a) := (s_2) \\ a = s_2 & \longrightarrow (y, a) := (0, s_3) \\ a = s_3 & \longrightarrow (y, t, a) := (1, 1, s_1) \end{array}$$

Process B:

$$\begin{array}{ll} b = t_0 & \longrightarrow (x, t, b) := (1, 0, t_1) \\ b = t_1 \wedge (y = 0 \vee t = 1) & \longrightarrow (b) := (t_2) \\ b = t_2 & \longrightarrow (x, b) := (0, t_3) \\ b = t_3 & \longrightarrow (x, t, b) := (1, 0, t_1) \end{array}$$

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

1. One of the processes reached the critical region ($a = s_2 \vee b = t_2$) releases the property that the system is either at the initial state ($a = s_0 \wedge b = t_0$) or some is waiting to enter the critical region ($x = 1 \vee y = 1$);
2. The value of y and t is consistent ($y = t$) unless both processes have tried to get into the critical region ($a \geq s_1 \wedge b \geq t_1$) and this continues until some process exited the critical region ($a = s_3 \vee b = t_3$).

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:

$$\begin{array}{l} I(a_0, a_1, b_0, b_1, x, y, t) \equiv \\ \quad x = 0 \wedge y = 0 \wedge t = 0 \wedge a_0 = 0 \wedge a_1 = 0 \wedge b_0 = 0 \wedge b_1 = 0 \end{array}$$

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

where $same(S)$ represents $v'_1 = v_1 \wedge \cdots \wedge v'_n = v_n$ for the set of propositions $S = \{v_1, ..., v_n\}$. Let

$p_1 \equiv (a_0 = 1 \wedge a_1 = 0 \vee b_0 = 1 \wedge b_1 = 0)$
$q_1 \equiv (a_0 = 0 \wedge a_1 = 0 \wedge b_0 = 0 \wedge b_1 = 0) \vee (x = 1 \vee y = 1)$
$p_2 \equiv (a_1 = 1 \vee a_0 = 1) \wedge (b_1 = 1 \vee b_0 = 1) \vee (y = 0 \wedge t = 0 \vee y = 1 \wedge t = 1)$
$q_2 \equiv (a_0 = 1 \wedge a_1 = 1 \vee b_0 = 1 \wedge b_1 = 1)$

We check the two properties: $M \models p_1 R q_1$ and $M \models p_2 U q_2$.

Property 1: For $M \models p_1 R q_1$, we check the satisfiability of $[[M, \neg(p_1 R q_1)]]_k$ and $[[M, \neg(p_1 R q_1)]]^v_k$ for $k = 0, 1, 2, ...$, until the first formula is satisfiable, the second formulas is unsatisfiable, or the completeness threshold is reached. Here, we only consider the use of $[[M, \neg(p_1 R q_1)]]^v_k$ to the verification of the property (since verification is the main concern of this paper). Let $V_i = \{u_{i,0}, ..., u_{i,6}\}$ and $same(S)$ represent $u_{i+1,j} = u_{i,j}$ for each $u_{i,j} \in S$. We have

$$[[M, \neg(p_1 R q_1)]]^v_k = I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge [[\neg(p_1 R q_1), u_0]]^v_k$$

where

$I(u_0) \equiv \neg u_{04} \wedge \neg u_{05} \wedge \neg u_{06} \wedge \neg u_{00} \wedge \neg u_{01} \wedge \neg u_{02} \wedge \neg u_{03}$

$T(u_i, u_{i+1}) \equiv$
$\neg u_{i,0} \wedge \neg u_{i,1} \wedge u_{i+1,5} \wedge u_{i+1,6} \wedge u_{i+1,1} \wedge same(V_i \setminus \{u_{i,5}, u_{i,6}, u_{i,1}\}) \vee$
$\neg u_{i,0} \wedge u_{i,1} \wedge (\neg u_{i,4} \vee \neg u_{i,6}) \wedge u_{i+1,0} \wedge \neg u_{i+1,1} \wedge same(V_i \setminus \{u_{i,0}, u_{i,1}\}) \vee$
$u_{i,0} \wedge \neg u_{i,1} \wedge \neg u_{i+1,5} \wedge u_{i+1,1} \wedge same(V_i \setminus \{u_{i,5}, u_{i,1}\}) \vee$
$u_{i,0} \wedge u_{i,1} \wedge u_{i+1,5} \wedge u_{i+1,6} \wedge \neg u_{i+1,0} \wedge same(V_i \setminus \{u_{i,5}, u_{i,6}, u_{i,0}\}) \vee$
$\neg u_{i,2} \wedge \neg u_{i,3} \wedge u_{i+1,4} \wedge \neg u_{i+1,6} \wedge u_{i+1,3} \wedge same(V_i \setminus \{u_{i,4}, u_{i,6}, u_{i,3}\}) \vee$
$\neg u_{i,2} \wedge u_{i,3} \wedge (\neg u_{i,5} \vee u_{i,6}) \wedge u_{i+1,2} \wedge \neg u_{i+1,3} \wedge same(V_i \setminus \{u_{i,2}, u_{i,3}\}) \vee$
$u_{i,2} \wedge \neg u_{i,3} \wedge \neg u_{i+1,4} \wedge u_{i+1,3} \wedge same(V_i \setminus \{u_{i,4}, u_{i,3}\}) \vee$
$u_{i,2} \wedge u_{i,3} \wedge u_{i+1,4} \wedge \neg u_{i+1,6} \wedge \neg u_{i+1,2} \wedge same(V_i \setminus \{u_{i,4}, u_{i,6}, u_{i,2}\})$

$[[\neg(p_1 R q_1), u_0]]^v_k \equiv$
$\bigvee_{j=0}^{k}((u_{j,0} \vee u_{j,1} \vee u_{j,2} \vee u_{j,3}) \wedge \neg u_{j,4} \wedge \neg u_{j,5} \wedge$
$\bigwedge_{t=0}^{j-1}((\neg u_{t,0} \vee u_{t,1}) \wedge (\neg u_{t,2} \vee u_{t,3}))) \vee \bigwedge_{t=0}^{k}((\neg u_{t,0} \vee u_{t,1}) \wedge (\neg u_{t,2} \vee u_{t,3}))$

Let us use $\varphi(k)$ to denote the conjunction of the above formulas. By feeding files with formulas representing the initial states, the transition relation and the property in required format into VERBS, we obtain that $\varphi(0)$, $\varphi(1)$ and $\varphi(2)$ are satisfiable, while $\varphi(3)$ is unsatisfiable and this proves $M \models p_1 R q_1$.

The result of running VERBS provides information to be interpreted as a path, if the result is satisfiable. For instance, for $k = 2$ and $\varphi(2)$ satisfiable, we can extract the following path from the information:

$a_0 = 0, a_1 = 0, b_0 = 0, b_1 = 0, x = 0, y = 0, t = 0$
$a_0 = 0, a_1 = 1, b_0 = 0, b_1 = 0, x = 0, y = 1, t = 1$
$a_0 = 0, a_1 = 1, b_0 = 0, b_1 = 1, x = 1, y = 1, t = 0$

This path information could sometimes be useful for error location, if the property is not valid. For instance, if we check $M \models Gp_2$ with $k = 4$, we obtain a path with a loop as follows.

$$\begin{array}{l} a_0 = 0, a_1 = 0, b_0 = 0, b_1 = 0, x = 0, y = 0, t = 0 \\ a_0 = 0, a_1 = 1, b_0 = 0, b_1 = 0, x = 0, y = 1, t = 1 \\ a_0 = 1, a_1 = 0, b_0 = 0, b_1 = 0, x = 0, y = 1, t = 1 \\ a_0 = 1, a_1 = 1, b_0 = 0, b_1 = 0, x = 0, y = 0, t = 1 \\ a_0 = 0, a_1 = 1, b_0 = 0, b_1 = 0, x = 0, y = 1, t = 1 \end{array}$$

This is a path with a loop that starts from the second state and has length 3. With this path information, we know that the 4-th state of this path violates p_2. Therefore Gp_2 does not hold in M. Then we may conclude that either Gp_2 is not a necessary requirement of such a model or the model should be modified in order to avoid such a path.

Property 2: For $M \models p_2 U q_2$, we check the satisfiability of $[[M, \neg(p_2 U q_2)]]_k$ and $[[M, \neg(p_2 U q_2)]]_k^v$ for $k = 0, 1, 2, ...$, until the first formula is satisfiable, the second formulas is unsatisfiable, or the completeness threshold is reached. We have

$$[[M, \neg(p_2 U q_2)]]_k^v = I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1}) \wedge [[\neg(p_2 U q_2), u_0]]_k^v$$

where $I(u_0)$ and $T(u_i, u_{i+1})$ are the same as that already specified previously, and $[[\neg(p_2 U q_2), u_0]]_k^v$ is as follows:

$\bigvee_{j=0}^{k}((\neg u_{j,0} \wedge \neg u_{j,1} \vee \neg u_{j,2} \wedge \neg u_{j,3}) \wedge (\neg u_{j,5} \wedge \neg u_{j,6}) \wedge (u_{j,5} \wedge u_{j,6}) \wedge$
$\bigwedge_{t=0}^{j}((\neg u_{t,0} \vee \neg u_{t,1}) \wedge (\neg u_{t,2} \vee \neg u_{t,3}))) \vee \bigwedge_{t=0}^{k}((\neg u_{t,0} \vee \neg u_{t,1}) \wedge (\neg u_{t,2} \vee \neg u_{t,3}))$

Let $\psi(k)$ denote the conjunction of this formula and $I(u_0) \wedge \bigwedge_{i=0}^{k-1} T(u_i, u_{i+1})$. We obtain that $\psi(0)$, $\psi(1)$, $\psi(2)$, $\psi(3)$ are satisfiable, while $\psi(4)$ is unsatisfiable and this proves $M \models p_2 U q_2$.

Table 1. Experimental Data

Property	k	Variables	Clauses	SAT
$\varphi(k)$	0	7+2	13	yes
	1	14+11	127	yes
	2	21+20	243	yes
	3	28+29	361	no

Property	k	Variables	Clauses	SAT
$\psi(k)$	0	7+2	18	yes
	1	14+11	137	yes
	2	21+20	258	yes
	3	28+29	381	yes
	4	35+38	506	no

Summary: Table 1 is a summary of the experimental data on a Sun Blade 1000 with 750 MHz and 512 MB. The number of variables is divided into two parts: 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.

6 Concluding Remarks

We have presented encodings of pairs of model and formula in SAT for the purpose of both verification of valid properties and error detection, in which the encoding with the emphasis on error detection is basically the same as that in [3], and proposed an approach to verify $M \models \varphi$ in the following way.

- Start with $k = 0$;
- If $[[M, \neg\varphi]]_k^v$ is unsatisfiable, report that $M \models \varphi$ is valid;
- If $[[M, \neg\varphi]]_k$ is satisfiable, report that $M \models \varphi$ does not hold;
- If a completeness threshold is reached, report that $M \models \varphi$ is valid;
- Increase k and repeat the process.

The case-study presented in the previous section shows that this approach is useful for checking formulas that are valid in a model (though, there are also weaknesses of the approach, cf. the discussion in Section 4), in the sense that the iteration stopped before a completeness threshold is reached. Although the system presented is simple with the completeness threshold bounded by a relatively small number, it is easy to construct systems by extending the model, such that the completeness threshold is larger than any given number, while the verification can still stop when k reaches respectively 3 and 4 for the given properties. Therefore the benefit of the use of this approach could be arbitrary large compared to the use of the completeness threshold, and this extends the practical capability of SAT based model checking to the verification of valid properties.

In a survey paper [18], Prasad, Biere and Gupta pointed out 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 [20, 15, 11, 2]. This study explores the strength of SAT-based techniques for verification of general LTL properties and the case study shows that this is a promising approach for certain types of applications as demonstrated in the case study.

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

References

1. S. Berezin and S. Campos and E. M. Clarke. Compositional Reasoning in Model Checking. Proceedings of COMPOS'97. Lecture Notes in Computer Science 1536: 81-102. 1998.
2. A. Biere, A. Cimmatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded Model Checking. Advances in Computers 58, Academic Press, 2003.
3. A. Biere, A. Cimmatti, E. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. LNCS 1579:193-207. TACAS 99.

4. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. IEEE Symposium on Logic in Computer Science 5: 428-439, 1990.
5. R. Bryant. Graph based algorithms for boolean function manipulation. IEEE Transaction on Computers 35(8):677-691. 1986.
6. E. M. Clarke, O. Grumberg and D. E. Long. Model Checking and Abstraction. ACM Transactions on Programming Languages and Systems 16(5): 1512-1542, 1994.
7. E. M. Clarke, D. Kroening, J. Ouaknine, and O. Strichman. Completeness and Complexity of Bounded Model Checking. VMCAI 2004: 85-96.
8. E. M. Clarke, D. E. Long and K. L. McMillan. Compositional Model Checking. IEEE Symposium on Logic in Computer Science 4: 353-362, 1989.
9. E. Allen Emerson and A. P. Sistla. Symmetry and model checking. Formal Methods in System Design 9:105-131. 1995.
10. J. Gregoire. Verification Model Reduction through Abstraction. Formal Design Techniques VII, 280-282, 1995.
11. Ranjit Jhala and Kenneth L. McMillan. Interpolation and SAT-based Model Checking. CAV 2003: 1-13.
12. D. Kroening, O. Strichman. Efficient Computation of Recurrence Diameters. VMCAI 2003: 298-309.
13. C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. Journal of Formal methods in System Design 6:1-35. 1995.
14. K. L. McMillan. Verification of Infinite State Systems by Compositional Model Checking. Lecture Notes in Computer Science 1703:219-234. CHARME 1999.
15. Leonardo de Moura, Harald Ruess, Maria Sorea. Bounded Model Checking and Induction: ¿From Refutation to Verification. CAV 2003: 14-26.
16. Doron A. Peled. Software Reliability Methods. Springer-Verlag. 2001.
17. A. Pnueli. A temporal logic of concurrent programs. Theoretical Computer Science 13:45-60. 1981.
18. Mukul R. Prasad, Armin Biere, Aarti Gupta. A survey of recent advances in SAT-based formal verification. STTT 7(2): 156-173 (2005).
19. V. Roy and R. de Simone. Auto/Autograph. In Computer Aided Verification. DIMACS series in Discrete Mathematics and Theoretical Computer Science 3: 235-250, June 1990.
20. Mary Sheeran, Satnam Singh and Gunnar Stlmarck. Checking Safety Properties Using Induction and a SAT-Solver. FMCAD 2000: 108-125.
21. A. Valmaru. Stubborn sets for reduced state space generation. LNCS 483(ICATPN'89):491-515. 1989.
22. P. Wolper and P. Godefroid. Partial-order methods for temporal verification. LNCS 715(CONCUR'93):233-246. 1993.
23. W. Zhang. Combining Static Analysis and Case-based Search Space Partitioning for Reducing Peak Memory in Model Checking. Journal of Computer Science and Technology 18(6):762-770, 2003.
24. W. Zhang, Z. Huang, and J. Zhang. Parallel Execution of Stochastic Search Procedures on Reduced SAT Instances. Lecture Notes in Computer Science 2417:108-117. Springer-Verlag. 2002.