Research Article

Bounded Model Checking of ETL Cooperating with Finite and Looping Automata Connectives

Rui Wang, Wanwei Liu, Tun Li, Xiaoguang Mao, and Ji Wang

College of Computer Science, National University of Defense Technology, Changsha, Hunan 410073, China

Correspondence should be addressed to Wanwei Liu; wwliu@nudt.edu.cn

Received 8 March 2013; Accepted 14 June 2013

Academic Editor: Xiaoyu Song

Copyright © 2013 Rui Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As a complementary technique of the BDD-based approach, bounded model checking (BMC) has been successfully applied to LTL symbolic model checking. However, the expressiveness of LTL is rather limited, and some important properties cannot be captured by such logic. In this paper, we present a semantic BMC encoding approach to deal with the mixture of ETL and ETL. Since such kind of temporal logic involves both finite and looping automata as connectives, all regular properties can be succinctly specified with it. The presented algorithm is integrated into the model checker ENuSMV, and the approach is evaluated via conducting a series of imperative experiments.

1. Introduction

A crucial bottleneck of model checking is the state-explosion problem, and the symbolic model checking technique has proven to be an applicable approach to alleviate it. In the early 1990s, McMillan presented the BDD [1] based model checking technique [2]. It is first applied to CTL model checking and is later adapted to deal with LTL. With the rapid evolvement of SAT solvers, an entirely new approach, namely, bounded model checking (BMC), is presented in [3]. It represents the problem “there is a path (with bounded length) violating the specification in the model” with a Boolean formula and then tests its satisfiability via a SAT solver. Usually, BMC is considered to be a complementary approach of the BDD-based approach: BMC is normally used for hunting bugs not for proving their absence. It performs better when handling a model having a large reachable state set but involving (relatively) shallow error runnings.

BMC has been successfully employed in LTL model checking. However, LTL has the drawback of limited expressiveness. Wolper was the first to complain about this by addressing the fact that some counting properties such as " holds at every even moment" cannot be expressed by any LTL formula [4]. Indeed, LTL formulae are just as expressive as star-free expressions, that is, -regular expressions disallowing arbitrary (in a star-free expression, Kleene-closure operators ( and ) can only be applied upon , which is the whole set of alphabet) use of Kleene-closure operators.

As pointed in [5, 6], it is of great importance for a specification language to have the power to express all -regular properties—as an example, it is a necessary requirement to support modular model checking. Actually, such specification language like PSL [7] has been accepted as industrial standard.

For temporal logics within linear framework, there are several ways to pursue such an expressiveness.

(1) The first way is to add fixed-point operators or propositional quantifiers to the logic, such as linear -calculus [8] and QLTL [9].

(2) An alternative choice is to add regular expressions to LTL-like logics, as done in RLTL [10], FTL [11, 12], and PSL [7].

(3) The third approach is to cooperate infinitely many temporal connectives with the logic, just like various of ETLs [4, 9, 13].

The first extension requires finitely many operators in defining formulae. Meanwhile, the use of fixed-point operators and higher-order quantifiers tends to rise difficulties in understanding. In contrast, using regular expressions or automata as syntactical ingredients is much more intuitive in
comprehension. To some extent, since nesting of automata connectives is allowed, the third approach generalizes the second one.

In [4], Wolper suggested using right linear grammars as connectives. Later, Wolper, Vardi, and Sistla consider taking various \( \omega \)-automata [9, 13]. Depending on the type of automata used as temporal connectives, we may obtain various ETLs. As a result, ETLs employing \( \omega \)-automata with looping, finite, and repeating (alternatively, Büchi [14]) acceptance are, respectively, named ETL\(_\omega\), ETL\(_f\), and ETL\(_r\), and all of them are known to be as expressive as \( \omega \)-regular expressions [13].

We have presented a bounded model checking algorithm for ETL\(_f\) in [15] and an algorithm for BDD-based model checking of an invariant of PSL in [16]. Jehle et al. present a bounded model checking algorithm for linear \( \mu \)-calculus in [17]. And in [18], a tester based symbolic model checking approach is proposed by Pnueli and Zacks to deal with PSL properties. Meanwhile, a modular symbolic Büchi automata construction is presented in [19] by Cimatti et al.

In this paper, we present a semantic BMC encoding for ETL employing both finite acceptance and looping acceptance automata connectives (we in the following refer to it as ETL\(_{l+f}\)). The reason that we study BMC algorithm for such kind of logic is for the following considerations.

(1) The BDD-based symbolic model checking technique for ETL\(_f\) has been established in [15] by extending LTL construction [20]. Nevertheless, in a pure theoretical perspective, looping and finite acceptance, respectively, correspond to safety and liveness properties, and looping acceptance automata can be viewed as the counterparts of finite acceptance automata. Actually, both similarities and differences could be found in compiling the semantic models and translating Boolean representations when dealing with these two types of connectives. Since ETL\(_{l+f}\) has a rich set of fragments, such as LTL, it is hopeful to develop a unified semantic BMC framework of such logics.

(2) Practically, things would usually be much more succinct when employing both types of automata connectives, in comparison to merely using finite or looping ones. As an example, there is no direct encoding for the temporal operator \( G \) just with finite acceptance automata—to do this with ETL\(_f\), we need to use a two-state and two-letter connective to represent the operator \( F \) and then to dualize it. In contrast, with looping automata, we just need to define a one-state and one-letter connective. It would save much space overhead in building tableaux.

(3) Lastly, unlike syntactic BMC encodings (such kind of encodings give inductive Boolean translations with the formulae’s structure, cf. [21, 22] for a survey), the semantic fashion [22] yields a natural completeness threshold computation approach, and it describes the fair path finding problem over the product model with Boolean formulae. In this paper, we give a linear semantic encoding approach (opposing to the original quadratic semantic encoding) for ETL\(_{l+r}\). Moreover, the technique can also be tailored to semantic LTL BMC.

We have implemented the presented algorithm with our model checker ENuSMV (Ver. 1.2), and this tool allows end users to customize temporal connectives by defining automata. We have also justified the algorithm by conducting a series of comparative experiments.

The paper is structured as follows: Section 2 briefly revisits its basic notions. Section 3 introduces semantic BMC encoding technique for ETL\(_{l+f}\). In Section 4, experimental results of ETL\(_{l+f}\) BMC are given. Finally, we conclude the whole paper with Section 5.

### 2. Preliminaries

An infinite word \( w \) over the alphabet \( \Sigma \) is a mapping from \( \mathbb{N} \) to \( \Sigma \); hence we may use \( w(i) \) to denote the \( i \)th letter of \( w \). For the sake of simplicity, we usually write \( w \) as the sequence \( w(0)w(1)\cdots \). A finite prefix of \( w \) with length \( n \) is a restriction of \( w \) to the domain \( \{0,\ldots,n-1\} \), denoted by \( w[n] \).

A (nondeterministic) automaton is a tuple \( A = (\Sigma, Q, \delta, q, F) \), where:

(i) \( \Sigma \) is a finite alphabet,

(ii) \( Q \) is a finite set of states,

(iii) \( \delta : Q \times \Sigma \to 2^Q \) is a transition function,

(iv) \( q \in Q \) is an initial state, and

(v) \( F \subseteq Q \) is a set of accepting states.

An infinite run of \( A = (\Sigma, Q, \delta, q, F) \) over an infinite word \( w \) is an infinite sequence \( \sigma = q_0q_1\cdots \in Q^\omega \), where \( q_0 = q \) and \( q_{i+1} \in \delta(q_i, w(i)) \) for each \( i \). In addition, we say that each prefix \( q_0 \cdots q_{n+1} \) is a finite run over \( w[n] \).

In this paper, we are concerned with two acceptance types for \( \omega \)-automata.

**Looping.** An infinite word \( w \) is accepted if it has an infinite run over \( w \).

**Finite.** An infinite word \( w \) is accepted if it has a finite prefix \( w[n] \), over which there is a finite run \( q_0 \cdots q_{n+1} \) and \( q_{n+1} \) is an accepting state (call such a prefix accepting prefix).

In both cases, we denote by \( \mathcal{L}(A) \) the set of infinite words accepted by \( A \).

Given an automaton \( A = (\Sigma, Q, \delta, q, F) \) and a state \( r \in Q \), we denote by \( A^r \) the automaton \( (\Sigma, Q, \delta, r, F) \). That is, \( A^r \) is almost identical to \( A \), except for that its initial state is replaced by \( r \). Hence, \( A \) and \( A^r \) are the same.

Given a set of atomic propositions \( AP \), the class of ETL\(_{l+f}\) formulae can be inductively defined as follows.

(i) Both \( T \) and \( \bot \) are ETL\(_{l+f}\) formulae.

(ii) Each proposition \( p \in AP \) is an ETL\(_{l+f}\) formula.

(iii) If \( \varphi \) is an ETL\(_{l+f}\) formula, then \( \neg \varphi \) and \( \varphi \oplus \varphi \) are ETL\(_{l+f}\) formulae.
When \( \phi_1, \phi_2 \) are ETL \(_{4+}\) formulae, then both \( \phi_1 \land \phi_2 \) and \( \phi_1 \lor \phi_2 \) are ETL \(_{4+}\) formulae.

(5) If \( \mathcal{A} \) is an automaton with the alphabet \( \Sigma = \{ a_1, \ldots, a_n \} \) and \( \phi_1, \ldots, \phi_n \) are ETL \(_{4+}\) formulae, then \( \mathcal{A} (\phi_1, \ldots, \phi_n) \) is also an ETL \(_{4+}\) formula.

Remark 1. In the original definition of various ETLs (say ETL\(_3\), ETL\(_4\), and ETL\(_4+\)), the “next operator” \((\text{o})\) is not explicitly declared. However, this operator is extremely important in building the semantic BMC encodings for ETL\(_{4+}\). Hence, we explicitly use this operator in our definition, and it would not change the expressiveness of the logic.

Remark 2. Since we employ both finite and looping acceptance automata connectives, our logic is a mixture of ETL\(_3\) and ETL\(_4\). On the one hand, ETL\(_{4+}\) generalizes both of these two logics; on the other hand, it can be embedded into ETL\(_3\); hence this logic is also as expressive as omega-regular expressions.

To make a better understanding of ETL\(_{4+}\) in place of \( \pi, i \models \neg \phi \) if and only if: there is an infinite word \( \pi \in (2^{\text{AP}})^{\omega} \) and a position \( i \in \mathbb{N} \) is inductively given as follows.

(i) \( \pi, i \models \top \) and \( \pi, i \not\models \bot \).
(ii) \( \pi, i \models p \) if and only if \( p \in \pi(i) \).
(iii) \( \pi, i \models \neg \phi \) if and only if \( \pi, i \not\models \phi \).
(iv) \( \pi, i \models \phi \lor \psi \) if and only if \( \pi, i + 1 \models \phi \).
(v) \( \pi, i \models \phi \land \psi \) if and only if \( \pi, i \models \phi \) and \( \pi, i \models \psi \).
(vi) \( \pi, i \models \phi \lor \psi \) if and only if \( \pi, i \models \phi \) or \( \pi, i \models \psi \).
(vii) If \( \mathcal{A} \) is a looping acceptance automaton with the alphabet \( \{ a_1, \ldots, a_n \} \), then \( \pi, i \models \mathcal{A} (\phi_1, \ldots, \phi_n) \) if and only if: there is an infinite word \( \omega \in \mathcal{L}(\mathcal{A}) \), and, for each \( j \in \mathbb{N} \), \( \omega(j) = a_k \) implies \( \pi, i + j \models \phi_k \).
(viii) If \( \mathcal{A} \) is a finite acceptance automaton with the alphabet \( \{ a_1, \ldots, a_n \} \), then \( \pi, i \models \mathcal{A} (\phi_1, \ldots, \phi_n) \) if and only if: there is an infinite word \( \omega \in \mathcal{L}(\mathcal{A}) \) with an accepting prefix \( \omega[n] \), such that, for each \( j < n \), \( \omega(j) = a_k \) implies \( \pi, i + j \models \phi_k \).

As usual, we directly use \( \pi \models \phi \) in place of \( \pi, 0 \models \phi \).

To make a better understanding of ETL\(_{4+}\) formulas, we here give some examples of the use of automata connectives.

(1) Considering the LTL formula \( \phi_1 \lor \phi_2 \), it can be described with an ETL\(_{4+}\) formula \( \mathcal{A}_U (\phi_1, \phi_2) \), where \( \mathcal{A}_U \) is the finite acceptance automaton \( \{ a_1, a_2 \}, \{ q_1, q_2 \}, \delta_U, q_1, \{ q_2 \} \) and we let \( \delta_U (q_1, a_1) = \{ q_1 \}, \delta_U (q_1, a_2) = \{ q_2 \}, \delta_U (q_2, a_1) = \delta_U (q_2, a_2) = \emptyset \).

(2) The LTL formula \( \phi \) is equivalent to the ETL\(_{4+}\) formula \( \mathcal{A}_G (\phi) \), where \( \mathcal{A}_G = \{ a_1, \{ q \}, \delta_G, q, \emptyset \} \) is a looping acceptance automaton and \( \delta_G (q, a) = \{ q \} \).

Remark 3. The order of letters is important in defining automata connectives. Hence, the alphabet should be considered as a vector, rather than a set.
(iii) \(\text{el}(\varphi_1 \land \varphi_2) = \text{el}(\varphi_1 \lor \varphi_2) = \text{el}(\varphi_1) \cup \text{el}(\varphi_2)\).

(iv) \(\text{el}(\varphi) = \text{el}(\varphi) \cup \{\varphi\}\).

(v) If \(\varphi = \mathcal{A}^i\varphi_1, \ldots, \varphi_n\) or \(\varphi = \neg \mathcal{A}^i\varphi_1, \ldots, \varphi_n\) and the states set of \(\mathcal{A}\) is \(Q\), then

\[
\text{el}(\varphi) = \bigcup_{1 \leq k \leq n} \text{el}(\varphi_k) \cup \{\text{el}^i(\varphi_{1},\ldots,\varphi_{n}) | q' \in Q\}.
\]

Hence, if \(\psi \in \text{el}(\varphi)\), then \(\psi\) is either an atomic proposition or a formula rooted at the next operator.

Subsequently, we define the function sat, which maps each subformula \(\psi\) of \(\varphi\) to a set of members in \(2^{\text{el}(\varphi)}\). Inductively the following hold.

(i) \(\text{sat}(\top) = 2^{\text{el}(\varphi)}; \text{sat}(\bot) = \emptyset\).

(ii) \(\text{sat}(p) = \{\Gamma \subseteq \text{el}(\varphi) \mid p \in \Gamma\} \text{ and } \text{sat}(\neg p) = \{\Gamma \subseteq \text{el}(\varphi) \mid p \notin \Gamma\}\).

(iii) \(\text{sat}(\varphi \land \psi) = \{\Gamma \subseteq \text{el}(\varphi) \mid \psi \in \Gamma\} \text{ and } \text{sat}(\varphi \lor \psi) = \{\Gamma \subseteq \text{el}(\varphi) \mid \psi \not\in \Gamma\}\).

(iv) \(\text{sat}(\varphi_1 \land \varphi_2) = \text{sat}(\varphi_1) \cap \text{sat}(\varphi_2) \text{ and } \text{sat}(\varphi_1 \lor \varphi_2) = \text{sat}(\varphi_1) \cup \text{sat}(\varphi_2)\).

(v) Suppose that \(\Delta = \{[a_1, \ldots, a_n], Q, \delta, q, F\}\).

(1) If \(\mathcal{A}\) is a looping acceptance automaton or a finite acceptance automaton and \(q \notin F\), then

\[
\text{sat}(\mathcal{A}^i(\varphi_1, \ldots, \varphi_n)) = \bigcup_{1 \leq k \leq n} \left( \text{sat}(\varphi_k) \cap \bigcup_{q' \in \delta(\varphi_{1:k})} \text{sat}^i(\varphi_{1:k}) \right).
\]

(2) If \(\mathcal{A}\) is a finite acceptance automaton and \(q \notin F\), then

\[
\text{sat}(\neg \mathcal{A}^i(\varphi_1, \ldots, \varphi_n)) = 2^{\text{el}(\varphi)} \setminus \text{sat}(\mathcal{A}^i(\varphi_1, \ldots, \varphi_n)).
\]

Likewise, for each NL-subformula \(\psi = \neg \mathcal{A}^i(\varphi_1, \ldots, \varphi_n)\) of \(\varphi\), we also define a relation \(\Delta^+_{\psi} \subseteq (2^{\text{el}(\varphi)} \times 2^Q) \times (2^{\text{el}(\varphi)} \times 2^Q)\). In detail, for any \(\Gamma, \Gamma' \subseteq \text{el}(\varphi)\) and \(P, P' \subseteq Q\), we have \(((\Gamma, P), (\Gamma', P')) \in \Delta^+_{\psi}\) if and only if the following hold.

(i) When \(P \neq \emptyset\), then, for each \(q \in P \setminus F\), there exists some \(1 \leq k \leq n\) such that \(\Gamma \in \text{sat}(\varphi_k)\) and \(P' \cap \delta(q, a) \neq \emptyset\).

(ii) When \(P = \emptyset\), then \(q \in P'\) if and only if \(\Gamma' \in \text{sat}(\mathcal{A}^i(\varphi_1, \ldots, \varphi_n))\) for each \(q \in Q\).

The below two theorems (Theorems 4 and 5) reveal the language property of ETL\(_{\mathcal{A}^i}\) tableaux. To remove the lengthiness, we here just provide the proof sketches, and rigorous proofs of them are postponed to the appendices.

**Theorem 4.** For each \(\pi \in (2^{2\mathcal{A}^i})^\omega\), if \(\pi \in \mathcal{L}(\mathcal{T}_\varphi)\), then \(\pi \vdash \varphi\).

**Proof (sketch).** Just assume that \(\sigma = s_0s_1 \ldots \in S^\omega\) is the corresponding fair path of \(\mathcal{T}_\varphi\) such that \(\pi = L_{\sigma^i}(\sigma)\), where \(s_i = (\Gamma_i; P_{i,1}, \ldots, P_{i,m}; R_{i,1}, \ldots, R_{i,n})\). We may inductively prove the following claim.

For each \(\psi \in \text{sub}(\varphi) \cup \text{el}(\varphi)\), we have: \(\Gamma_i \in \text{sat}(\psi)\) implies \(\pi, i \vdash \varphi\).

Because we require that \(\Gamma_0 \in \text{sat}(\varphi)\), hence we have \(\pi, 0 \vdash \varphi\).

**Theorem 5.** For each \(\pi \in (2^{2\mathcal{A}^i})^\omega\), if \(\pi \vdash \varphi\), then \(\pi \in \mathcal{L}(\mathcal{T}_\varphi)\).

**Proof (sketch).** Suppose that \(\pi \vdash \varphi\); to show \(\pi \in \mathcal{L}(\mathcal{T}_\varphi)\), we need to first construct an infinite state sequence \(\sigma = s_0s_1 \ldots \in S^\omega\) guided by \(\pi\) (the detailed construction is given in Section A.2), and then we will subsequently show that \(\sigma\) is a fair path of \(\mathcal{T}_\varphi\) and \(\pi = L_{\sigma^i}(\sigma)\).
The following theorem is immediate from Theorems 4 and 5.

**Theorem 6.** The model \( \mathcal{M} \) violates the ETL \( \mathcal{L}_f \) property \( \varphi \) if and only if \( \mathcal{L}(\mathcal{M}) \cap \mathcal{L}(\mathcal{L}_f \neg) \neq \emptyset \), equivalently; there exists some path fair in \( \mathcal{M} \times \mathcal{L}_f \neg \).

**Theorem 7.** For an ETL \( \mathcal{L}_f \) formula \( \varphi \), its tableau \( \mathcal{T} \) has at most \( 4^{\text{el}(\varphi)} \) states.

Proof. Observe that a state should be of the form \( \langle \Gamma; p_1, \ldots, p_m; R_1, \ldots, R_n \rangle \). For \( \Gamma \), there are \( 2^{\text{el}(\varphi)} \) possible choices. Suppose that \( \psi_j = \mathcal{A} \mathcal{F}(\varphi_1, \ldots, \varphi_k) \) (resp., \( \eta_j = \neg \mathcal{A} \mathcal{F}(\varphi_1, \ldots, \varphi_k) \)) and the state set of \( \mathcal{A} \mathcal{F} \) is \( Q_j \) (resp., \( \mathcal{A} \mathcal{F} \) with \( Q'_j \)). According to the construction, each \( q \in Q_j \) (resp., \( q \in Q'_j \)) corresponds to a unique elementary formula \( \circ \mathcal{A} \mathcal{F}(\varphi_1, \ldots, \varphi_k) \) (resp., \( \circ \neg \mathcal{A} \mathcal{F}(\varphi_1, \ldots, \varphi_k) \)), and such a mapping is an injection. Hence we have

\[
\left( \sum_{1 \leq j \leq m} |Q_j| + \sum_{1 \leq j \leq n} |Q'_j| \right) \leq |\text{el}(\varphi) \setminus \text{AP}|.
\]

Note that \( Q_j \subseteq \varphi_j \) (resp., \( R_j \subseteq \varphi'_j \)), and hence we have \(|S\varphi| \leq (2^{\text{el}(\varphi)})^2 = 4^{\text{el}(\varphi)} \). \( \square \)

### 3.2. The Linear Semantic Encoding.

Practically, a model's state space is determined by the evaluation of a set of variables. Further, we may assume that each of them is a "Boolean variable" (which corresponds to a proposition belonging to \( \text{AP} \)), because every variable over finite domain could be encoded with several Boolean variables.

Let \( \mathcal{C} = \langle S, \rho, I, L, \mathcal{F} \rangle \) be an arbitrary LTS, and we also assume that the corresponding variable set is \( V = \{ p_1, \ldots, p_n \} \); then each state \( s \in S \) uniquely corresponds to an assignment of such \( p_s \).

If we use \( s(p_j) \) to denote the value of \( p_j \) at \( s \), then each subset \( Z \subseteq S \) can be represented by a Boolean formula \( \Phi_Z \) over \( V \). In detail, it fulfills

\[
s \in S \iff s \models \Phi_Z,
\]

where \( s \models \Phi_Z \) means that \( \Phi_Z \) is evaluated to be true if we assign each \( p_j \) with the value \( s(p_j) \).

Let \( V' = \{ p_1', \ldots, p_n' \} \), and each binary relation \( \lambda \subseteq S \times S \) also has a Boolean representation \( \Phi_\lambda \) over the variable set \( V \cup V' \). That is,

\[
(s_1, s_2) \in \lambda \iff (s_1, s_2) \models \Phi_\lambda,
\]

where \( (s_1, s_2) \models \Phi_\lambda \) means that \( \Phi_\lambda \) is evaluated to be true if we assign each \( p_j \) with \( s_1(p_j) \) and assign each \( p_j' \) with \( s_2(p_j) \).

Hence, all components of \( \mathcal{M} \) can be encoded: \( I \) and \( \rho \) can be represented by two Boolean formulae \( \Phi_I \) and \( \Phi_\rho \), respectively; we subsequently create a Boolean formula \( \Phi_F \) for each \( F \in \mathcal{F} \); note that the labeling function \( L \) is not concerned any longer, because the states labeled with \( p \) can be captured by the Boolean formula \( \Phi_F \).

For example, from Theorem 7, we have that the symbolic representation of \( \mathcal{T} \) requires \( 2 \times (\text{el}(\varphi) \setminus \text{AP}) \) new Boolean variables—because variables in \( \text{el}(\varphi) \cap \text{AP} \) can be shared with the encoding of the original model.

A canonical Boolean encoding of fair path existence detection upon LTSs is presented in [22]: given a model \( \mathcal{C} = \langle S, \rho, I, L, \mathcal{F} \rangle \) and a bound \( k \in \mathbb{N} \), one may use the formula

\[
\Phi_F^{(0)} \wedge \bigwedge_{0 \leq i < k} \Phi_F^{(i+1)} \wedge \bigvee_{0 \leq i < k} \left( \Phi_F^{(k)} \wedge \bigwedge_{F \in \mathcal{F}} \bigvee_{0 \leq j < k} \Phi_F^{(j)} \right),
\]

where \( \Phi_F^{(i)} \) and \( \Phi_F^{(j)} \) are, respectively, the Boolean formulae obtained from \( \Phi_I \) and \( \Phi_\rho \) by replacing each variable \( p \) with a new copy \( p^{(i)} \), and \( \Phi_F^{(j)} \) is obtained from \( \Phi_F \) by replacing each \( p^{(j)} \) and replacing each \( p' \) with \( p^{(j)} \).

It can be seen that this formula is satisfiable if and only if \( \mathcal{C} \) involves a fair path of the form \( s_1 s_2 \cdots s_{i-1} (s_j \cdots s_k)^{\infty} \) (call it of the lasso shape). Since that \( \mathcal{L}(\mathcal{C}) \neq \emptyset \) if and only if \( \mathcal{C} \) contains some lasso path fair (note that from each fair path we may derive another fair path of lasso shape), hence we may convert the fair path detection into the satisfiability problem of the above Boolean formula.

However, a closer look shows that the size of such encoding is quadratic with the bound. To reduce the blow-up in size, we need to introduce the following new variables (the linearization can also be done with the syntactic fashion presented in [23, 24]). We would draw a comparison of these two approaches in Section 4.

1. For each \( 0 \leq \ell \leq k \), we introduce a new variable \( r_\ell \). Intuitively, \( r_\ell \) indicates that \( s_\ell \) is a successor of \( s_k \).
2. For each fairness constraint \( F \in \mathcal{F} \) and each \( 0 \leq \ell \leq k \), we introduce a variable \( f^{(i)}_F \), and this variable is evaluated to be true only if there is some \( \Phi_F^{(i)} \) which is evaluated to true, where \( \ell \leq j \leq k \).

And the new encoding (with the bound \( k \in \mathbb{N} \)) can be formulated as

\[
\Psi_{\mathcal{C}}^{(k)} \equiv \wedge_{0 \leq \ell < k} \left( r_\ell \rightarrow \left( \Phi_I^{(k)} \wedge \bigwedge_{F \in \mathcal{F}} f^{(\ell + 1)}_F \right) \right) \wedge \left( \bigwedge_{F \in \mathcal{F}} \left( f^{(0)}_F \rightarrow f^{(k)}_F \right) \right)
\]

Hence, both the number of variables and the size of this encoding are linear with \( k \). Moreover, the following theorem guarantees the correctness of such encoding.

**Theorem 8.** \( \mathcal{L}(\mathcal{C}) \neq \emptyset \) if and only if \( \Psi_{\mathcal{C}}^{(k)} \) is satisfiable for some \( k \).
Proof. We begin with the “if” direction: suppose that the variable set is \( \{p_1, \ldots, p_m\} \); if there is some \( k \) such that \( \Psi_{\mathcal{E}}^{(k)} \) is evaluated to \( 1 \) (i.e., true) under the assignment \( e \), then we denote \( s_i = (e(p_1^{(i)}), \ldots, e(p_m^{(i)})) \) for each \( 1 \leq i \leq m \). Hence, each \( s_i \) is a state of \( \mathcal{E} \).

(i) Since the truth value of \( \Phi_{\mathcal{F}}^{(0)} \) is 1 under \( e \), then we have \( s_0 \models \Phi_{\mathcal{F}}^{(0)} \); this implies that \( s_0 \in I \).

(ii) For each \( 0 \leq i < k \), we have \( (s_i, s_{i+1}) \models \Phi_{\mathcal{F}}^{(i+1)} \), and thus \( (s_i, s_{i+1}) \in \rho \).

(iii) Because we have the conjunct \( \bigvee_{0 \leq \ell \leq \ell'} r_\ell \), then there is some \( 0 \leq \ell \leq k \) such that \( e(r_\ell) = 1 \). In the following, we fix this specific value \( \ell \) for the discussion.

(iv) According to the constraint \( r_\ell \rightarrow \langle \Phi_{\mathcal{F}}^{(k,\ell)} \land \bigwedge_{F \in \mathcal{F}} f_F^{(\ell)} \rangle \), we have the following.

\[
(1) \quad e(\Phi_{\mathcal{F}}^{(k,\ell)}) = 1, \text{ which indicates that } (s_\ell, s_{\ell+1}) \in \rho.
\]

(2) For each \( F \in \mathcal{F} \), we have \( e(f_F^{(\ell)}) = 1 \).

(v) For each fairness constraint \( F \in \mathcal{F} \) and \( 0 \leq i \leq k \), we now inductively show that “\( e(f_F^{(i)}) = 1 \) implies \( e(\Phi_{\mathcal{F}}^{(i)}) = 1 \)” (alternatively, \( s_i \in F \)) for some \( i \leq j \leq k \).

First of all, it holds in the case of \( i = k \), because we have the constraint \( f_{\mathcal{F}}^{(k)} \rightarrow \Phi_{\mathcal{F}}^{(k)} \). In addition, the fact of “when \( i = c \), it holds” can be immediately inferred from the hypothesis “when \( i = c + 1 \), it holds,” according to the conjunct \( f_{\mathcal{F}}^{(0)} \rightarrow (f_{\mathcal{F}}^{(i+1)} \lor \Phi_{\mathcal{F}}^{(i)}) \). Since we have shown that \( e(f_F^{(\ell)}) = 1 \), we can conclude that there exists some \( 0 \leq j \leq k \) such that \( s_j \in F \).

The above shows that \( s_0 s_1 \cdots s_{\ell-1} (s_\ell \cdots s_k) \) is a path of \( \mathcal{E} \), and hence \( \mathcal{E}(\mathcal{E}) \neq \emptyset \).

Conversely, for the “only if” direction, it suffices to find some \( k \) and some assignment \( e \) evaluating \( \Psi_{\mathcal{E}}^{(k)} \) to be true since \( \mathcal{E}(\mathcal{E}) \neq \emptyset \), there must exist some path of lasso shape in \( \mathcal{E} \). Without loss of generality, assume that \( \sigma = s_0 s_1 \cdots s_{\ell-1} (s_\ell \cdots s_k) \) is such a path; just let \( k \) be this value, and we now illustrate how the assignment \( e \) is constructed.

(i) For each variable \( p_j \) and each \( 1 \leq i \leq k \), we let \( e(p_j^{(i)}) = s_i(p_j) \). Since \( s_0 \) is an initial state, according to the definition, we have \( e(\Phi_{\mathcal{F}}^{(0)}) = 1 \). Meanwhile, because each \( (s_i, s_{i+1}) \in \rho \), we have that the conjunction \( \bigwedge_{0 \leq i \leq k} \Phi_{\mathcal{F}}^{(i+1)} \) is satisfied under \( e \).

(ii) For each \( 0 \leq i \leq k \), we let

\[
e(r_i) = \begin{cases} 1, & i = \ell, \\ 0, & i \neq \ell. \end{cases}
\]

Then it can be seen that \( e(\bigvee_{0 \leq i \leq k} r_i) = 1 \).

(iii) For each \( F \in \mathcal{F} \) and each \( 0 \leq i \leq k \), we let

\[
e(f_F^{(i)}) = \begin{cases} 1, & \text{if there is some } j \geq i \text{ such that } s_j \in F, \\ 0, & \text{otherwise.} \end{cases}
\]

Then it can be directly checked that the conjunct

\[
\bigwedge_{0 \leq i \leq k} \left( f_{\mathcal{F}}^{(i)} \rightarrow (f_{\mathcal{F}}^{(i+1)} \lor \Phi_{\mathcal{F}}^{(i)}) \right)
\]

is evaluated to be true under \( e \).

(iv) Since \( (s_\ell, s_{\ell+1}) \in \rho \), we have \( e(\Phi_{\mathcal{F}}^{(k,\ell)}) = 1 \). Also note that \( \sigma \) is a fair path; then for each \( F \in \mathcal{F} \) there is some \( s_j \in F \), where \( 0 \leq j \leq k \). According to the previous definition, we can infer that \( e(\bigwedge_{F \in \mathcal{F}} f_F^{(\ell)}) = 1 \). Thus the conjunct

\[
\bigwedge_{0 \leq i \leq k} \left( r_i \rightarrow \left( \Phi_{\mathcal{F}}^{(k,i)} \land \bigwedge_{F \in \mathcal{F}} f_F^{(i)} \right) \right)
\]

is also satisfied under \( e \) (recall that we have assigned \( e(r_i) = 0 \) in the case of \( i \neq \ell \)).

Thus, the formula \( \Psi_{\mathcal{E}}^{(k)} \) is satisfiable.

For bounded model checking, an important issue is the completeness threshold, which is the specific value \( k \) such that we may declare \( \mathcal{E}(\mathcal{E}) = \emptyset \) in the case that \( \Psi_{\mathcal{E}}^{(k)} \) is not satisfiable, and we denote it by \( \text{CT}(\mathcal{E}) \) in this paper.

Since we need only to concern about fair paths of the form \( \sigma_1 (\sigma_2) \), as pointed in [22], a possible candidate for the completeness threshold \( \text{CT}(\mathcal{E}) \) is

\[
D^f(\mathcal{E}) + |\mathcal{F}| \times D(\mathcal{E}),
\]

where \( D \) and \( D^f \) are, respectively, the diameter and the initialized diameter (cf. [22]). Since [22] just considers LTSs having only one fairness constraint, we here add the factor \( |\mathcal{F}| \).

Observe that the part \( \sigma_1 \) must be enclosed in some SCC (i.e., strongly connected component) of \( \mathcal{E} \), and we may replace \( D(\mathcal{E}) \) with \( D(\mathcal{E}) \), where \( \mathcal{E} \) is the largest SCC that intersects all fairness constraints. Therefore, we may get a more compact upper bound of the completeness threshold.

Thus, for a given LTS \( \mathcal{M} \) and the given ETL-\( f \) formula \( \varphi \), since we have shown that \( \mathcal{M} \not\models \varphi \) if and only if \( \mathcal{M} \times \mathcal{F} \) involves some fair path, we now just need to test if there is some \( k \) making \( \Psi_{\mathcal{E} \times \mathcal{F} \rightarrow \mathcal{E}}^{(k)} \) satisfiable, where \( k \leq \text{CT}(\mathcal{M} \times \mathcal{F} \rightarrow \mathcal{E}) \).

Remark 9. In the case that \( \mathcal{E} = \mathcal{E}_1 \times \mathcal{E}_2 \), where \( \mathcal{E} = (S, p, l, \mathcal{F}) \) and \( \mathcal{E}_i = (S_i, p_i, l_i, \mathcal{F}_i) \) for \( i = 1, 2 \), we have \( \Phi_{\mathcal{F}} = \Phi_{\mathcal{F}_1} \lor \Phi_{\mathcal{F}_2}, \Phi_\varphi = \Phi_{\varphi_1} \lor \Phi_{\varphi_2}, \Phi_{\Delta} = \Phi_{\Delta_1} \lor \Phi_{\Delta_2}, \) and in addition, the variable set of \( \mathcal{E} \) is just the union of the variable sets of \( \mathcal{E}_1 \) and \( \mathcal{E}_2 \).

Remark 10. Actually, for an ETL-\( f \) formula \( \varphi \), the symbolic representation of \( \mathcal{F} \) can be directly given without the detour of explicit construction of the LTS. Because, the relation sat could be inductively constructed if we introduce corresponding new variables in \( e(\varphi) \). Subsequently, encodings of \( \Phi_{\Delta_\varphi} \) could be naturally obtained from the underlying Boolean variables corresponding to states of automata connectives. Hence, the Boolean representation of \( \Phi_{\varphi_\Delta} \) is obtained. And, encodings of other components are as routine.
CONNECTIVE A (a1, a2) : FIN
STATES >q1, q2, q3<;
TRANSITIONS(q1)
case a1 : q2; a2 : q3; esac;
TRANSITIONS(q2)
case
a1 : f(q2, q3); g;
a2 : q3;
esac;
TRANSITIONS(q3)
case a1 :
f(q1, q2); g;
esac;

Figure 1: An automata connective declaration in ENuSMV.

Table 1: Comparative results of BDD-based and BMC approaches.

<table>
<thead>
<tr>
<th>Cells</th>
<th>BDD Time (s)</th>
<th>BDD Memory (MB)</th>
<th>BMC Time (s)</th>
<th>BMC Memory (MB)</th>
<th>C.L.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>170.56</td>
<td>68.32</td>
<td>13.50</td>
<td>34.98</td>
<td>39</td>
</tr>
<tr>
<td>6</td>
<td>513.06</td>
<td>209.17</td>
<td>29.14</td>
<td>42.37</td>
<td>39</td>
</tr>
<tr>
<td>7</td>
<td>2372.28</td>
<td>976.14</td>
<td>41.32</td>
<td>58.48</td>
<td>39</td>
</tr>
<tr>
<td>8</td>
<td>8074.05</td>
<td>1482.15</td>
<td>52.45</td>
<td>67.32</td>
<td>39</td>
</tr>
<tr>
<td>9</td>
<td>≥5h</td>
<td>≥2G</td>
<td>68.17</td>
<td>82.13</td>
<td>39</td>
</tr>
<tr>
<td>10</td>
<td>≥5h</td>
<td>≥2G</td>
<td>85.13</td>
<td>101.21</td>
<td>39</td>
</tr>
</tbody>
</table>

4. Experimental Results

To justify our idea, we have integrated (the tool is available at https://sourceforge.net/projects/enusmv12/) the ETL_{t+f} BMC algorithm into ENuSMV (Ver. 1.2). This tool is completely compatible with NuSMV [25], and it allows end-users to customize new temporal connectives by defining automata.

For example, Figure 1 illustrates how to declare a finite acceptance automata connective (to define a looping acceptance automata connective, just replace the keyword FIN with LOOP), namely, A. Since it has three states q1, q2, and q3 (where q1 is the initial state and q3 is an accepting state), then A[q1], A[q2], and A[q3] are also connectives—for example, A[q2] just replaces the initial state with q2. Subsequently, one may define ETL_{t+f} specifications; for example,

ETLSPEC A(p, A[q2] (q, p))

is a proper declaration.

In this redistribution, both BDD-based and bounded model checkings for ETL_{t+f} are supported. To perform (semantic encoding based) BMC, we need to use the command option bmc.tab.

We have conducted some experiments to test the correctness and efficiency of our algorithm. In this paper, we are especially concerned with the following issues.

1. The comparison of BDD-based symbolic model checking and bounded model checking.
2. The overhead contrast in verifications of ETL_f and ETL_{t+f} upon both star-free and non-star-free properties.
3. The comparison of performances with syntactic/semantic BMC of LTL and semantic BMC of ETL_{t+f}.

To compare the efficiencies between BDD-based MC and BMC, we chose the (distributed mutual exclusion) DME circuit as the model (which involves a buggy design), as described in [3]. It consists of n cells for n users that want to have exclusive access to a shared resource. We conducted the experiment by describing the liveness property that “a request for using the resource will eventually be acknowledged” (with ETL_{t+f} formula). The max bound are set to 100, and the comparative results are shown in Table 1, where “C.L.” stands for the length of counterexample.

As a previous work, we have implemented the symbolic model checking algorithm for ETL_f in ENuSMV 1.0. To justify that in general ETL_{t+f} could be more effectively checked, we would make a comparison of BMC for ETL_f and ETL_{t+f}.

To draw the comparison upon non-start-free regular properties, we use a “mod 2^n” counter as the model. The model consists of n “cells” bit_0, ..., bit_{n-1}. Each cell is a (mod 2) counter having an input carry_in and an output signal carry_out. These cells are connected in a serial manner; that is, bit_0’s carry_in is set to 1, and bit_i’s carry_in is connected to bit_{i-1}’s carry_out as described in Figure 2.
We, respectively, describe the periodicity property that “bit_0 carries out at every even (except for 0) moment” with ETL_f and ETL_{t_0,f}. We set the time bound to 1 hour, and Table 2 provides the max bounds (together with related information) which can be handled by the SAT solver within the time bound. From it, we can see that a deeper search could be done when specifications are described with ETL_{t_0,f}.

To compare the overhead of ETL_f and ETL_{t_0,f} upon star-free properties, we would first use the DME model to check the safety property: “no two cells will be simultaneously acknowledged.” The results are shown in Table 3, and the time bound is also set to 1 hour.

At the same time, we can also compare the verification performances of the DME model upon the aforementioned liveness property that “each request will be acknowledged in the further.” Note that for this property, the verification could be accomplished within the given time bound, and a counterexample could be detected at the bound k = 39. The comparative results are given in Table 4.

The last group of experiments aims at comparing the efficiencies of (syntactic/semantic) LTL BMC and ETL BMC. First of all, for LTL BMC, we are also concerned with two types of encoding approaches.

(1) The Syntactic Approach. We here adopt the linear incremental syntactic encoding proposed in [26]—to the best of our knowledge, this is the most effective syntactic encoding for full LTL.

(2) The Semantic Approach. ENuSMV 1.2 also supports semantic encoding for LTL—this is tailored from our linear encoding presented in Section 3.2.

We still use the DME circuit as the model and the liveness property as specification; Table 5 provides the experimental results on LTL BMC based on syntactic and semantic encodings. From that, we can see that, with semantic encoding, it tends to generate less clauses and tends to terminate earlier than that with the syntactic encoding, whereas the latter requires fewer variables.

Meanwhile, we can also make a comparison between Tables 4 and 5; we may find that the variable numbers of semantic ETL BMC and LTL BMC are almost at a fixed ratio—for this experiment, the ratio is 1.09 (approximately).

5. Concluding Remarks

The logic ETL_{t+1,f} is a variant of extended temporal logic, it employs both finite and looping acceptance automata connectives, and it can be considered a mixture of ETL_f and ETL_{t_0,f}. Thus, any omega-regular properties can be succinctly described with this kind of logic, particularly for safety and liveness properties.

We have presented the semantic bounded model checking algorithm for ETL_{t+1,f}. The central part of this approach is the tableau construction. Meanwhile, we also illustrate how to give a linear BMC encoding for it. To justify it, we have implemented the presented algorithm (in ENuSMV 1.2). Experimental results show that ETL_{t+1,f} could be more efficiently verified via BMC (in comparison to our previous implementation for ETL_{t_0,f}).

In this paper, verification of ETL_f, namely, extended temporal logic using Büchi (alternatively, repeating) automata as connectives, has not been studied. This is partly because of the inherent difficulties of Büchi complementation [27]. Indeed, we may mimic the ranking complementing technique of Büchi automata [28–30]. However, this would cause an asymptotically quadratic blow-up of variable number in building the tableaux. Hence, a further work is about to study the semantic BMC encodings of ETL_f.

Appendix

A. Omitted Proofs

A.1. Proof of Theorem 4. Just assume that \( \sigma = s_0 \delta_1 \cdots s_1 \in S^\omega_b \) is the corresponding fair path of \( \mathcal{T}_p \) having \( \pi = L_p(\sigma) \), where \( s_i = (\Gamma_i; P_{j_1}, \ldots, P_{j_m_i}; R_{i_1}, \ldots, R_{i_n}) \). We now inductively prove the following claim.

“For each \( \psi \in \text{sub}(\phi) \cup \text{el}(\phi) \), we have: \( \Gamma_i \in \text{sat}(\psi) \) implies \( \pi, i \models \psi \).

(i) The basic cases are trivial.

(a) \( \psi = \top \), or \( \psi = \bot \). Since \( \pi, i \models \top \) holds trivially, the case \( \Gamma_i \in \text{sat}(\bot) \) never happens.
(b) $\psi = p$, or $\psi = \neg p$, where $p \in AP$. Because $\Gamma_i \in \text{sat}(p)$ if and only if $\pi_i \in p$, similar to when $\psi = \neg p$.

(ii) The following inductions are as routine.

(a) For the first case, $\psi = \psi_1 \land \psi_2$, or $\psi_1 \lor \psi_2$. For the first case, we have $\Gamma_i \in \text{sat}(\psi)$ if and only if $\Gamma_i \in \text{Sat}(\psi_1)$ and $\Gamma_i \in \text{sat}(\psi_2)$; by induction, we have $\pi, i \models \psi_1$ and $\pi, i \models \psi_2$; hence $\pi, i \models \psi$; similar to the case of $\psi_1 \lor \psi_2$.

(b) In the case of $\psi = \varphi'$, we have $\Gamma_i \in \text{sat}(\psi)$ if and only if $\Gamma_i \in \text{sat}(\varphi')$, according to the definition of $\Delta_\varphi$ (note that in this case $\psi$ is an elementary formula of $\varphi$). By induction, we have $\pi, i + 1 \models \psi'$; hence $\pi, i \models \psi$.

(iii) If $\psi = \varphi^0(\varphi_1, \ldots, \varphi_k)$ is a PF-subformula, without loss of generality, assume $\psi = \varphi_j$ (i.e., the $j$th PF-subformula of $\varphi$), where $1 \leq j \leq m$ and $\varphi^0 = \langle (a_1, \ldots, a_k), Q, \delta, \delta, F \rangle$, then the proof is given as follows.

Since $\sigma$ is a fair path, there exists $i_2 > i_1 > i$, such that $P_{j, i} = P_{j, i_1} = \emptyset$ and $P_{j, i} \neq \emptyset$ for each $i < c < i_2$.

(a) First, let $q_0 = q$; then we have $\Gamma_i \in \text{sat}(\varphi_i(\varphi_1, \ldots, \varphi_k))$.

(b) For each $i_1 \leq i < i$, if $q_i \notin F$ and $\Gamma_i \in \text{sat}(\varphi_i(\varphi_1, \ldots, \varphi_k))$, then, according to the definition of $\Delta_\varphi$, there is some $1 \leq k_i \leq k$ and a $q'$ such that $\Gamma_i \in \text{sat}(q_k)$, $q' \in \delta(q_i, a_k)$, and $\Gamma_i \in \text{sat}(\varphi_i(\varphi_1, \ldots, \varphi_k))$ (equivalently, $\Gamma_{i+1} \in \text{sat}(\varphi_i(\varphi_1, \ldots, \varphi_k))$, as discussed above). Now, we let $q_{i+1} = q'$.

Till now, if there is some $t$ having $q_t \in F$, we stop the processing. Otherwise, we go on with the following processing.

(a) When $t = i_1 - i + 1$, we have $\Gamma_i \in \text{sat}(\varphi_i(\varphi_1, \ldots, \varphi_k))$. Since $P_{j, i} = \emptyset$, according to the definition of $\Delta_\varphi$, we have $q_i \in P_{j, i_1} = P_{j, i+1}$.

(b) For each $i_1 - i < t \leq i_2 - i$, if $q_t \in P_{j, i_2}$ and $q_t \notin F$, according to $\Delta_\varphi$, there is some $1 \leq k_t \leq k$, such that $\Gamma_{i_2} \in \text{sat}(q_k)$, and there is some $q' \in P_{j, i_2+t} \cap \delta(q_i, a_k)$. Now, we let $q_{i+1} = q'$.

Notice that $P_{j, i_2} = \emptyset$, and we have the inductive assertion that $q_t \notin F$ implies $q_t \in P_{j, i+1}$, hence there must exist some $q_t \in F$.

Thus, $\varphi$ have some accepting prefix $a_k a_k \ldots a_k$ of some infinite word. On the other hand, by induction, $\Gamma_{i_2+t} \in \text{sat}(q_k)$ implies $\pi, i + t \models q_k$. Hence, we have $\pi, i \models \psi$ by definition.

(iv) If $\psi = \neg \varphi^0(\varphi_1, \ldots, \varphi_k)$ is an NF-subformula of $\varphi$ and assume that $\varphi^0 = \langle (a_1, \ldots, a_k), Q, \delta, \delta, F \rangle$, then the proof is given as below.

Assume by contradiction it is not the case; then, $\pi, i \in \text{sat}(\varphi_1(\varphi_1, \ldots, \varphi_k))$. Therefore, there exist some accepting prefix $a_k a_k \ldots a_k$, some of infinite word and a word sequence $q_0 q_1 \ldots q_{t+1}$ such that

(a) $q_0 = q$, $q_{t+1} \in F$ and each $q_{i+1} \in \delta(q_i, a_k)$,

(b) for each $t \leq \ell$, we have $\pi, i + t \models q_k$.

Then, we have the following facts.

(a) $\Gamma_{i+\ell+1} \in \text{sat}(\varphi_{i+\ell+1}(\varphi_1, \ldots, \varphi_k)) = 2^c(\varphi)$, because $q_{t+1} \in F$.

(b) Assume that $\Gamma_{i+\ell} \in \text{sat}(\varphi_{i+\ell}(\varphi_1, \ldots, \varphi_k))$, where $t \leq \ell$; then $\Gamma_{i+\ell} \in \text{sat}(\varphi_{i+\ell}(\varphi_1, \ldots, \varphi_k))$. In addition, $\pi, i + t \models q_k$ implies that $\Gamma_{i+t} \in \text{sat}(q_k)$ (otherwise, $\pi, i + t \models \neg q_k$, by induction (contradicts!)

Hence, by the definition of sat, we have $\Gamma_{i+t} \in \text{sat}(\varphi_{i+\ell}(\varphi_1, \ldots, \varphi_k))$.

Then, we have $\Gamma_i \in \text{sat}(\varphi_{i+\ell}(\varphi_1, \ldots, \varphi_k))$ when $t = 0$. However, it contradicts the premiss $\Gamma_i \in \text{sat}(\varphi_{i+\ell}(\varphi_1, \ldots, \varphi_k))$, because $q_0 = q$. Therefore, the assumption $\pi, i \models \varphi_{i+\ell}(\varphi_1, \ldots, \varphi_k)$ is a falsity, and this implies that $\pi, i \models \psi$ holds.

(v) If $\psi = \varphi^0(\varphi_1, \ldots, \varphi_k)$ is a PL-subformula of $\varphi$, where $\varphi^0 = \langle (a_1, \ldots, a_k), Q, \delta, \delta, F \rangle$, then we have the following proof in the case that $\Gamma_i \in \text{sat}(\psi)$.

(a) Let $q_0 = q$, and $\Gamma_i \in \text{sat}(\varphi^0(\varphi_1, \ldots, \varphi_k))$.

(b) For each $t \geq 0$, assume that $\Gamma_{i+t} \in \text{sat}(\varphi^0(\varphi_1, \ldots, \varphi_k))$; then by the definition of sat, there is some $1 \leq k_t \leq k$, such that $\Gamma_{i+t} \in \text{sat}(q_k)$, and there is

<table>
<thead>
<tr>
<th>Cells</th>
<th>Max bound</th>
<th>Memory (MB)</th>
<th>Variables</th>
<th>Clauses</th>
<th>Max-bound</th>
<th>Memory (MB)</th>
<th>Variables</th>
<th>Clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>68</td>
<td>29.14</td>
<td>4002</td>
<td>8091</td>
<td>101</td>
<td>38.03</td>
<td>4958</td>
<td>9898</td>
</tr>
<tr>
<td>4</td>
<td>62</td>
<td>34.76</td>
<td>4864</td>
<td>9764</td>
<td>87</td>
<td>40.34</td>
<td>5712</td>
<td>11512</td>
</tr>
<tr>
<td>5</td>
<td>54</td>
<td>40.12</td>
<td>5170</td>
<td>10313</td>
<td>73</td>
<td>45.37</td>
<td>6072</td>
<td>12088</td>
</tr>
</tbody>
</table>

Table 3: Comparison of ETL$_f$ and ETL$_{i,f}$ with safety properties.

<table>
<thead>
<tr>
<th>Cells</th>
<th>Time (s)</th>
<th>Variables</th>
<th>Clauses</th>
<th>Memory (MB)</th>
<th>Time (s)</th>
<th>Variables</th>
<th>Clauses</th>
<th>Memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>25.09</td>
<td>2400</td>
<td>4796</td>
<td>81.66</td>
<td>29.14</td>
<td>4002</td>
<td>9764</td>
<td>23.48</td>
</tr>
<tr>
<td>4</td>
<td>31.42</td>
<td>3120</td>
<td>6200</td>
<td>38.61</td>
<td>29.14</td>
<td>4002</td>
<td>9764</td>
<td>23.48</td>
</tr>
<tr>
<td>5</td>
<td>42.56</td>
<td>3840</td>
<td>7604</td>
<td>44.28</td>
<td>29.14</td>
<td>4002</td>
<td>9764</td>
<td>23.48</td>
</tr>
</tbody>
</table>

Table 4: Comparison of ETL$_f$ and ETL$_{i,f}$ with liveness properties.
Table 5: Comparison of syntactic/semantic LTL BMC with liveness properties.

<table>
<thead>
<tr>
<th>Cells</th>
<th>Time (s)</th>
<th>Syntactic LTL BMC encoding</th>
<th>Semantic LTL BMC encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Variables</td>
<td>Clauses</td>
</tr>
<tr>
<td>3</td>
<td>9.34</td>
<td>2052</td>
<td>5474</td>
</tr>
<tr>
<td>4</td>
<td>16.71</td>
<td>2736</td>
<td>6806</td>
</tr>
<tr>
<td>5</td>
<td>24.06</td>
<td>3420</td>
<td>8138</td>
</tr>
</tbody>
</table>

For each $i \geq 0$, we let $\Gamma_i = \{ \psi \in \text{el}(\varphi) \mid \pi, i \vdash \psi \}$. We will now show the following claim.

“For each $\psi \in \text{sub}(\varphi) \cup \text{el}(\varphi)$, we have $\Gamma_i \in \text{sat}(\psi)$ if and only if $\pi, i \vdash \psi$.”

By induction of the structure of $\psi$ the following hold.

(a) Cases are trivial when $\psi = \bot$ or $\psi = \top$.

(b) If $\psi = p \in A\varphi$, then $\psi \in \text{el}(\varphi)$. Thus, $\Gamma_i \in \text{sat}(p)$ if and only if $p \in \pi(i)$ if and only if $\pi, i \vdash p$. Similar to show the case of $\psi = \neg p$.

(c) Another basic case is $\psi = \circ \psi'$: note that in this case we also have $\psi \in \text{el}(\varphi)$; hence $\Gamma_i \in \text{sat}(\psi')$ if and only if $\psi \in \Gamma_i$ if and only if $\pi, i \vdash \psi$.

(d) If $\psi = \psi_1 \land \psi_2$, then $\Gamma_i \in \text{sat}(\psi)$ if and only if $\Gamma_i \in \text{sat}(\psi_1)$ and $\Gamma_i \in \text{sat}(\psi_2)$.

(e) If $\psi = \mathcal{A}(\varphi_1, \ldots, \varphi_k)$, where $\mathcal{A} = \{a_1, \ldots, a_k\}$, $Q, \delta, q, F$ and if $\mathcal{A}$ is a looping acceptance automaton or a finite acceptance automaton, but $q \notin F$, it is not difficult to show that

\[
\mathcal{A}(\varphi_1, \ldots, \varphi_k) \rightarrow \bigvee_{1 \leq i \leq k} \left( \varphi_i \land \bigwedge_{q \in \delta(q_i)} \mathcal{A}(\varphi_1, \ldots, \varphi_k) \right).
\]  

Then by induction and according to the scheme $\text{sat}(\psi' \lor \psi'') = \text{sat}(\psi') \cup \text{sat}(\psi'')$ and $\text{sat}(\psi' \land \psi'') = \text{sat}(\psi') \cap \text{sat}(\psi'')$, we have $\pi, i \vdash \psi$ if and only if

\[
\Gamma_i \in \bigcup_{1 \leq i \leq k} \left( \text{sat}(\varphi_i) \cap \bigcup_{q \in \delta(q_i)} \text{sat}(\mathcal{A}(\varphi_1, \ldots, \varphi_k)) \right)
\]

or

\[
\text{sat}(\mathcal{A}(\varphi_1, \ldots, \varphi_k)).
\]

Otherwise, if $\mathcal{A}$ is a finite acceptance automaton and $q \in F$, then

\[
\mathcal{A}(\varphi_1, \ldots, \varphi_k) \rightarrow \top.
\]

In this case, $\text{sat}(\mathcal{A}(\varphi_1, \ldots, \varphi_k)) = 2^{\text{el}(\varphi)}$, and hence the claim “$\pi, i \vdash \psi$ if and only if $\Gamma_i \in \text{sat}(\psi)$” also holds.

(f) If $\psi = \neg \mathcal{A}(\varphi_1, \ldots, \varphi_k)$, then $\Gamma_i \in \text{sat}(\psi)$ if and only if $\pi, i \not\vdash \mathcal{A}(\varphi_1, \ldots, \varphi_k)$ if and only if $\Gamma_i \not\in \text{sat}(\mathcal{A}(\varphi_1, \ldots, \varphi_k))$ if and only if $\pi, i \not\vdash \mathcal{A}(\varphi_1, \ldots, \varphi_k)$.
Now, we have the following properties.

(1) Since \( \pi, 0 \models \varphi \), we have \( \Gamma_0 \models \text{sat}(\varphi) \), and hence \( s_0 \in I_\varphi \).

(2) For each \( i \) and each \( \omega \psi \in e(\varphi) \), we have \( \Gamma_i \models \text{sat}(\omega \psi) \) if and only if \( \pi, i \models \psi \) if and only if \( \pi, i + 1 \models \psi \) if and only if \( \Gamma_{i+1} \models \text{sat}(\psi) \).

(3) For each \( p \in AP \) and each \( i \), we have \( p \in \pi(i) \) if and only if \( \pi, i \models p \) if and only if \( p \in \Gamma_i \). Hence, \( \pi(i) = \Gamma_i \cap AP \).

(ii) The construction of \( P_{j,i} \), for \( 1 \leq j \leq m \) is as follows.

Assume that the \( j \)th PF-subformula \( \psi_j = \rho^j(\varphi_1, \ldots, \varphi_k) \) and the automata connective \( \rho^j = \langle \{a_1, \ldots, a_k\}, Q, \delta, q, F \rangle \); then the construction is as follows.

(a) We will find a series of “key positions” \( \ell_0, \ell_1, \ldots \), having \( P_{j,\ell_i} = \emptyset \) for each \( \ell_i \). First, let \( \ell_0 = 0 \), and, for each \( t \geq 0 \), once \( \ell_t \) is determined, we use the following steps to determine \( \ell_{t+1} \) and each \( P_{j,\ell_t} \) for \( \ell_t < c < \ell_{t+1} \).

(b) Let \( P_{j,\ell_i} = \emptyset \), and let \( P_{j,\ell_i+1} = \{q' | \pi, \ell_i + 1 \models \rho^i(\varphi_1, \ldots, \varphi_k)\} \). For each \( q' \in P_{j,\ell_i+1} \), there must exist a finite word \( w_{q'} = a_{q_0}a_{q_1} \cdots a_{q_t} \) and a finite state sequence \( \chi_{q_i} = q_0q_1 \cdots q_{t+1} \) such that \( q_0 = q' \), \( q_{t+1} \in \Gamma_i \), and, for each \( s \leq t \), we have \( q_{t+1} \in \rho(q_0, a_0) \) and \( \pi, q_t + c + 1 \models \varphi_k \) (equivalently, \( \Gamma_{q_t+1+1} \models \text{sat}(\varphi_k) \)).

(c) Now, let \( \ell_{t+1} = \ell_t + \max\{|\chi_{q'}| | q' \in P_{j,\ell_i+1}\} + 1 \). And we let

\[
P_{j,\ell_t} = \{\chi_{q'}(c - \ell_t - 1) | q' \in P_{j,\ell_t+1}\} - 1
\]

for each \( \ell_t + 1 < c < \ell_{t+1} \), where \( \chi_{q'}(s) \) is the \( s \)th element of the \( \chi_{q'} \).

Now, it is not hard to check that \( (\Gamma_i, P_{j,i}, (\Gamma_{i+1}, P_{j,i+1})) \in \Delta^\psi_i \) for each \( i \). Moreover, for each \( t \), we have \( P_{j,\ell_t} = \emptyset \).

(iii) The construction of \( R_{j,i} \) for each \( 1 \leq j \leq n \) is as follows.

Assume that the \( j \)th NL-subformula \( \neg \eta_j = \neg \rho^j(\varphi_1, \ldots, \varphi_k) \) and the automata connective \( \rho^j = \langle \{a_1, \ldots, a_k\}, Q, \delta, q, F \rangle \); then the construction is as follows.

(a) First, let \( \delta_0 = 0 \), and, for each \( t \geq 0 \), once \( \delta_t \) is decided, we use the following steps to determine \( \delta_{t+1} \) and each set \( R_{j,t} \) for \( \delta_t < c < \delta_{t+1} \).

(b) Let \( R_{j,\delta_t} = \emptyset \) and \( R_{j,\delta_t+1} = \{q' | (\Gamma_{\delta_t+1} \in \text{sat}(\neg \rho^j(\varphi_1, \ldots, \varphi_k))\} \). For each \( c \geq \delta_t + 1 \), let

\[
R_{j,\delta_t+1} = \bigcup_{q' \in R_{j,\delta_t}} \left\{ \delta(q', a_i) | 1 \leq s \leq k, \Gamma_c \models \text{sat}(\varphi_i) \right\}.
\]

(c) Since we have shown that \( \pi, i \models \psi \) if and only if \( \pi, i \models \psi \) for every \( \psi \in \text{sub}(\varphi) \cup e(\varphi) \), it is not hard to show that if \( R_{j,\delta_{t+1}} = \emptyset \), then there is some \( c > \delta_t + 1 \) such that \( R_{j,c} = \emptyset \) — otherwise (because, if \( R_{j,c} \neq \emptyset \) for each \( t > \delta_t \)), using König’s lemma, we may find a run of \( \rho^j \) over some infinite word), we can show


Submit your manuscripts at http://www.hindawi.com