Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 462532, 12 pages
http://dx.doi.org/10.1155/2013/462532

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

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_f and ETL_l. 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 imperial 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 rerpresents 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 "p holds at every even moment" cannot be expressed by any LTL formula [4]. Indeed, LTL formulae are just as expressive as *star-free* ω-expressions, that is, ω-regular expressions disallowing arbitrary (in a star-free expression, Kleene-closure operators ($\bullet^*$ and $\bullet^\omega$) can only be applied upon Σ, 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 ω-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 μ-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 ω-automata [9, 13]. Depending on the type of automata used as temporal connectives, we may obtain various ETLs. As a result, ETLs employing ω-automata with looping, finite, and repeating (alternatively, Büchi [14]) acceptance are, respectively, named ETL_l, ETL_f, and ETL_r, and all of them are known to be as expressive as ω-regular expressions [13].

We have presented a BDD-based 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 μ-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+f}. 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 Σ is a mapping from $\mathbb{N}$ to Σ; hence we may use $w(i)$ to denote the ith 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 $\mathscr{A} = \langle \Sigma, Q, \delta, q, F\rangle$, where:

(i) Σ 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 $\mathscr{A} = \langle \Sigma, Q, \delta, q, F\rangle$ over an infinite word w is an infinite sequence $\sigma = q_0 q_1 \cdots \in Q^\omega$, 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 ω-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 $\mathscr{L}(\mathscr{A})$ the set of infinite words accepted by $\mathscr{A}$.

Given an automaton $\mathscr{A} = \langle \Sigma, Q, \delta, q, F\rangle$ and a state $r \in Q$, we denote by $\mathscr{A}^r$ the automaton $\langle \Sigma, Q, \delta, r, F\rangle$. That is, $\mathscr{A}^r$ is almost identical to $\mathscr{A}$, except for that its initial state is replaced by r. Hence, $\mathscr{A}$ and $\mathscr{A}^q$ 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 $\top$ and $\bot$ are ETL_{l+f} formulae.

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

(iii) If φ is an ETL_{l+f} formula, then $\neg\varphi$ and $\circ\varphi$ are ETL_{l+f} formulae.

(iv) If φ_1, φ_2 are ETL_{l+f} formulae, then both $\varphi_1 \wedge \varphi_2$ and $\varphi_1 \vee \varphi_2$ are ETL_{l+f} formulae.

(v) If $\mathscr{A}$ is an automaton with the alphabet $\Sigma = \{a_1, \ldots, a_n\}$ and $\varphi_1, \ldots, \varphi_n$ are ETL_{l+f} formulae, then $\mathscr{A}(\varphi_1, \ldots, \varphi_n)$ is also an ETL_{l+f} formula.

Remark 1. In the original definition of various ETLs (say ETL_l, ETL_f, and ETL_r), the "next operator" ($\circ$) is not explicitly declared. However, this operator is extremely important in building the semantic BMC encodings for ETL_{l+f}. 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_l and ETL_f. On the one hand, ETL_{l+f} generalizes both of these two logics; on the other hand, it can be embedded into ETL_r; hence this logic is also as expressive as omega-regular expressions.

The *satisfaction* relation of an ETL_{l+f} formula φ with respect to an infinite word $\pi \in (2^{AP})^\omega$ and a position $i \in \mathbb{N}$ is inductively given as follows.

(i) $\pi, i \vDash \top$ and $\pi, i \nvDash \bot$.

(ii) $\pi, i \vDash p$ if and only if $p \in \pi(i)$.

(iii) $\pi, i \vDash \neg\varphi$ if and only if $\pi, i \nvDash \varphi$.

(iv) $\pi, i \vDash \circ\varphi$ if and only if $\pi, i+1 \vDash \varphi$.

(v) $\pi, i \vDash \varphi_1 \wedge \varphi_2$ if and only if $\pi, i \vDash \varphi_1$ and $\pi, i \vDash \varphi_2$.

(vi) $\pi, i \vDash \varphi_1 \vee \varphi_2$ if and only if $\pi, i \vDash \varphi_1$ or $\pi, i \vDash \varphi_2$.

(vii) If $\mathscr{A}$ is a looping acceptance automaton with the alphabet $\{a_1, \ldots, a_n\}$, then $\pi, i \vDash \mathscr{A}(\varphi_1, \ldots, \varphi_n)$ if and only if: there is an infinite word $w \in \mathscr{L}(\mathscr{A})$, and, for each $j \in \mathbb{N}$, $w(j) = a_k$ implies $\pi, i+j \vDash \varphi_k$.

(viii) If $\mathscr{A}$ is a finite acceptance automaton with the alphabet $\{a_1, \ldots, a_n\}$, then $\pi, i \vDash \mathscr{A}(\varphi_1, \ldots, \varphi_n)$ if and only if: there is an infinite word $w \in \mathscr{L}(\mathscr{A})$ with an accepting prefix $w[n]$, such that, for each $j < n$, $w(j) = a_k$ implies $\pi, i+j \vDash \varphi_k$.

As usual, we directly use $\pi \vDash \varphi$ in place of $\pi, 0 \vDash \varphi$.

To make a better understanding of ETL_{l+f} formulas, we here give some examples of the use of automata connectives.

(1) Considering the LTL formula $\varphi_1 \mathbf{U} \varphi_2$, it can be described with an ETL_{l+f} formula $\mathscr{A}_{\mathbf{U}}(\varphi_1, \varphi_2)$, where $\mathscr{A}_{\mathbf{U}}$ is the finite acceptance automaton $\langle \{a_1, a_2\}, \{q_1, q_2\}, \delta_{\mathbf{U}}, q_1, \{q_2\}\rangle$, and we let $\delta_{\mathbf{U}}(q_1, a_1) = \{q_1\}$, $\delta_{\mathbf{U}}(q_1, a_2) = \{q_2\}$, and $\delta_{\mathbf{U}}(q_2, a_1) = \delta_{\mathbf{U}}(q_2, a_2) = \emptyset$.

(2) The LTL formula $\mathbf{G}\varphi$ is equivalent to the ETL_{l+f} formula $\mathscr{A}_{\mathbf{G}}(\varphi)$, where $\mathscr{A}_{\mathbf{G}} = \langle \{a\}, \{q\}, \delta_{\mathbf{G}}, q, \emptyset\rangle$ is a looping acceptance automaton and $\delta_{\mathbf{G}}(q, a) = \{q\}$.

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.

We use $\mathrm{sub}(\varphi)$ to denote the set of subformulae of φ. A formula φ is in *negation normal form* (NNF) if all negations in φ are adjacent to atomic propositions or automata connectives. One can achieve this by repeatedly using De Morgan's law and the schemas of $\neg\circ\varphi \equiv \circ\neg\varphi$ and $\neg\neg\varphi \equiv \varphi$. In addition, we call a formula φ being of the form $\mathscr{A}(\varphi_1, \ldots, \varphi_n)$ an *automaton formula*.

Given a formula φ (in NNF), we use a two-letter-acronym to designate the type of an automaton subformula of φ: the first letter is either "P" or "N," which means "positive" or "negative"; and the second letter can be "F" or "L," which describes the acceptance type. For example, NL-subformulae stand for "negative automata formulae with looping automata connectives," such as $\neg\mathscr{A}^q(\varphi_1, \varphi_2)$, where $\mathscr{A}$ is a two-letter looping automaton.

A *model* or interchangeably a *labeled transition system* (LTS) is a tuple $\mathscr{M} = \langle S, \rho, I, L, \mathscr{F}\rangle$, where:

(i) S is a finite set of states,

(ii) $\rho \subseteq S \times S$ is a transition relation (usually, we require ρ to be total; that is, for each $s \in S$, there is some $s' \in S$ having $(s, s') \in \rho$),

(iii) $I \subseteq S$ is the set of initial states,

(iv) $L : S \to 2^{AP}$ is the labeling function, and

(v) $\mathscr{F} \subseteq 2^S$ is a set of fairness constraints.

A *path* of $\mathscr{M}$ is an infinite sequence $\sigma = s_0 s_1 \cdots \in S^\omega$, where $s_0 \in I$ and $(s_i, s_{i+1}) \in \rho$ for each $i \in \mathbb{N}$. In addition, σ is a *fair path* if σ visits each $F \in \mathscr{F}$ infinitely often. Formally, σ is a fair path if $\inf(\sigma) \cap F \neq \emptyset$ for each $F \in \mathscr{F}$, where $\inf(\sigma)$ denotes the set of states occurring infinitely many times in σ.

An infinite word $\pi = a_0 a_1 \cdots$ is *derived* from a path σ of $\mathscr{M}$ (denoted by $\pi = L(\sigma)$) if $a_i = L(s_i)$ for each $i \in \mathbb{N}$. We use $\mathscr{L}(\mathscr{M})$ to denote the set of infinite words derived from fair paths of $\mathscr{M}$.

Given an ETL_{l+f} formula φ and an LTS $\mathscr{M}$, we denote by $\mathscr{M} \vDash \varphi$ if $\pi \vDash \varphi$ for each $\pi \in \mathscr{L}(\mathscr{M})$. The *model checking problem* of ETL_{l+f} is just to verify if $\mathscr{M} \vDash \varphi$ holds for the given LTS $\mathscr{M}$ and the given ETL_{l+f} formula φ.

3. Semantic BMC Encoding for ETL_{l+f}

In this section, we will give a detailed description of the semantic BMC encoding for ETL_{l+f}. Firstly, we show how to extend the tableau construction of LTL [20] to that of ETL_{l+f}, and hence a product model can also be constructed. Subsequently, we interpret the fairness path finding problem (upon the product model) into SAT, and the size blow-up of this encoding is linear with the bound.

For the sake of convenience, in this section, we always assume that the given ETL_{l+f} formulae have been normalized into NNF.

3.1. The Tableaux of ETL_{l+f} Formulae. Given an ETL_{l+f} formula φ, we first inductively define its *elementary formula set* $\mathrm{el}(\varphi)$ as follows.

(i) $\mathrm{el}(\top) = \mathrm{el}(\bot) = \emptyset$.

(ii) $\mathrm{el}(p) = \mathrm{el}(\neg p) = \{p\}$ for each $p \in AP$.

(iii) $\mathrm{el}(\varphi_1 \wedge \varphi_2) = \mathrm{el}(\varphi_1 \vee \varphi_2) = \mathrm{el}(\varphi_1) \cup \mathrm{el}(\varphi_2)$.

(iv) $\mathrm{el}(\circ\varphi) = \mathrm{el}(\varphi) \cup \{\circ\varphi\}$.

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

$$\mathrm{el}(\varphi) = \bigcup_{1\le k\le n} \mathrm{el}(\varphi_k) \cup \left\{\circ\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_n) \mid q' \in Q\right\}. \quad (1)$$

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

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

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

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

(iii) $\mathrm{sat}(\circ\psi) = \{\Gamma \subseteq \mathrm{el}(\varphi) \mid \circ\psi \in \Gamma\}$.

(iv) $\mathrm{sat}(\varphi_1 \wedge \varphi_2) = \mathrm{sat}(\varphi_1) \cap \mathrm{sat}(\varphi_2)$ and $\mathrm{sat}(\varphi_1 \vee \varphi_2) = \mathrm{sat}(\varphi_1) \cup \mathrm{sat}(\varphi_2)$.

(v) Suppose that $\mathscr{A} = \langle\{a_1, \ldots, a_n\}, Q, \delta, q, F\rangle$.

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

$$\mathrm{sat}\left(\mathscr{A}^q(\varphi_1, \ldots, \varphi_n)\right) = \bigcup_{1\le k\le n}\left(\mathrm{sat}(\varphi_k) \cap \bigcup_{q'\in\delta(q,a_k)} \mathrm{sat}\left(\circ\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_n)\right)\right). \quad (2)$$

(2) If $\mathscr{A}$ is a finite acceptance automaton and $q \in F$, then $\mathrm{sat}(\mathscr{A}^q(\varphi_1, \ldots, \varphi_n)) = 2^{\mathrm{el}(\varphi)}$.

(vi) $\mathrm{sat}(\neg\mathscr{A}^q(\varphi_1, \ldots, \varphi_n)) = 2^{\mathrm{el}(\varphi)} \setminus \mathrm{sat}(\mathscr{A}^q(\varphi_1, \ldots, \varphi_n))$.

Recall the tableau construction for LTL [20], an "until subformula" would generate a fairness constraint to the tableau. Indeed, such a subformula corresponds to a "least-fixpoint subformula" if we translate the specification into a logic employing higher-order quantifiers, such as μ-calculus. Similarly, for ETL_{l+f}, the PF- and NL-subformulae also impose fairness constraints. For this reason, we need to define the following two auxiliary relations before giving the tableau construction.

For a PF-subformula $\psi = \mathscr{A}(\varphi_1, \ldots, \varphi_n)$ of φ, where $\mathscr{A} = \langle\{a_1, \ldots, a_n\}, Q, \delta, q, F\rangle$, we define a relation $\Delta^+_\psi \subseteq (2^{\mathrm{el}(\varphi)} \times 2^Q) \times (2^{\mathrm{el}(\varphi)} \times 2^Q)$ as follows: suppose that $\Gamma, \Gamma' \subseteq \mathrm{el}(\varphi)$ and $P, P' \subseteq Q$; then $((\Gamma, P), (\Gamma', P')) \in \Delta^+_\psi$ if and only if the following hold.

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

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

Likewise, for each NL-subformula $\psi = \neg\mathscr{A}(\varphi_1, \ldots, \varphi_n)$ of φ, we also define a relation $\Delta^-_\psi \subseteq (2^{\mathrm{el}(\varphi)} \times 2^Q) \times (2^{\mathrm{el}(\varphi)} \times 2^Q)$. In detail, for any $\Gamma, \Gamma' \subseteq \mathrm{el}(\varphi)$ 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 $q \in P$ and $1 \le k \le n$, we have: $\Gamma \in \mathrm{sat}(\varphi_k)$ implies $\delta(q, a_k) \subseteq P'$.

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

We now describe the tableau construction for φ. Suppose that $\psi_1, \ldots, \psi_m$ and $\neg\eta_1, \ldots, \neg\eta_n$ are, respectively, all the PF-subformulae and NL-subformulae occurring in φ then the tableau $\mathscr{T}_\varphi$ is such an LTS $\langle S_\varphi, \rho_\varphi, I_\varphi, L_\varphi, \mathscr{F}_\varphi\rangle$, where:

(i) S_φ consists of tuples like $\langle\Gamma; P_1, \ldots, P_m; R_1, \ldots, R_n\rangle$, where $\Gamma \subseteq \mathrm{el}(\varphi)$ and each P_i (resp., R_i) is a subset of ψ_i's (resp., η_i's) connective's state set.

(ii) For two states $s = \langle\Gamma; P_1, \ldots, P_m; R_1, \ldots, R_n\rangle$ and $s' = \langle\Gamma'; P'_1, \ldots, P'_m; R'_1, \ldots, R'_n\rangle$, $(s, s') \in \rho_\varphi$ if and only if the following three conditions hold.

(1) $\Gamma \in \mathrm{sat}(\circ\psi)$ if and only if $\Gamma' \in \mathrm{sat}(\psi)$ for each $\circ\psi \in \mathrm{el}(\varphi)$.

(2) $((\Gamma, P_i), (\Gamma', P'_i)) \in \Delta^+_{\psi_i}$ for each $1 \le i \le m$.

(3) $((\Gamma, R_j), (\Gamma', R'_j)) \in \Delta^-_{\neg\eta_j}$ for each $1 \le j \le n$.

(iii) $I_\varphi = \{\langle\Gamma; P_1, \ldots, P_m; R_1, \ldots, R_n\rangle \in S_\varphi \mid \Gamma \in \mathrm{sat}(\varphi)\}$.

(iv) $L_\varphi(\langle\Gamma; P_1, \ldots, P_m; R_1, \ldots, R_n\rangle) = \Gamma \cap AP$.

(v) $\mathscr{F}_\varphi = \{F^+_i \mid 1 \le i \le m\} \cup \{F^-_j \mid 1 \le j \le n\}$, where

$$F^+_i = \left\{\langle\Gamma; P_1, \ldots, P_m; R_1, \ldots, R_n\rangle \in S_\varphi \mid P_i = \emptyset\right\},$$
$$F^-_j = \left\{\langle\Gamma; P_1, \ldots, P_m; R_1, \ldots, R_n\rangle \in S_\varphi \mid R_j = \emptyset\right\}. \quad (3)$$

The below two theorems (Theorems 4 and 5) reveal the *language property* of ETL_{l+f} 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 $\pi \in (2^{AP})^\omega$, if $\pi \in \mathscr{L}(\mathscr{T}_\varphi)$, then $\pi \vDash \varphi$.*

Proof (sketch). Just assume that $\sigma = s_0 s_1 \cdots \in S^\omega_\varphi$ is the corresponding fair path of $\mathscr{T}_\varphi$ such that $\pi = L_\varphi(\sigma)$, where $s_i = \langle\Gamma_i; P_{1,i}, \ldots, P_{m,i}; R_{1,i}, \ldots, R_{n,i}\rangle$. We may inductively prove the following claim.

"For each $\psi \in \mathrm{sub}(\varphi) \cup el(\varphi)$, we have: $\Gamma_i \in \mathrm{sat}(\psi)$ implies $\pi, i \vDash \psi$."

Because we require that $\Gamma_0 \in \mathrm{sat}(\varphi)$, hence we have $\pi, 0 \vDash \varphi$. □

Theorem 5. *For each $\pi \in (2^{AP})^\omega$, if $\pi \vDash \varphi$, then $\pi \in \mathscr{L}(\mathscr{T}_\varphi)$.*

Proof (sketch). Suppose that $\pi \vDash \varphi$; to show $\pi \in \mathscr{L}(\mathscr{T}_\varphi)$, we need to first construct an infinite state sequence $\sigma = s_0 s_1 \cdots \in S^\omega_\varphi$ guided by π (the detailed construction is given in Section A.2), and then we will subsequently show that σ is a fair path of $\mathscr{T}_\varphi$ and $\pi = L_\varphi(\sigma)$. □

The following theorem is immediate from Theorems 4 and 5.

Theorem 6. *The model $\mathscr{M}$ violates the ETL_{l+f} property φ if and only if $\mathscr{L}(\mathscr{M}) \cap \mathscr{L}(\mathscr{T}_{\neg\varphi}) \neq \emptyset$, equivalently; there exists some fair path in $\mathscr{M} \times \mathscr{T}_{\neg\varphi}$.*

Theorem 7. *For an ETL_{l+f} formula φ, its tableau $\mathscr{T}_\varphi$ has at most $4^{|el(\varphi)|}$ 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 Γ, there are $2^{|\mathrm{el}(\varphi)|}$ possible choices. Suppose that $\psi_j = \mathscr{A}_j(\varphi_1, \ldots, \varphi_k)$ (resp., $\neg\eta_i = \neg\mathscr{A}_i(\varphi'_1, \ldots, \varphi'_r)$) and the state set of $\mathscr{A}_j$ is Q_j (resp., $\mathscr{A}_i$ with Q'_i). According to the construction, each $q \in Q_j$ (resp., $q \in Q'_i$) corresponds to a unique elementary formula $\circ\mathscr{A}^q_j(\varphi_1, \ldots, \varphi_k)$ (resp., $\circ\mathscr{A}^q_i(\varphi'_1, \ldots, \varphi'_r)$), and such a mapping is an injection. Hence we have

$$\left(\sum_{1 \le j \le m} \left|Q_j\right| + \sum_{1 \le i \le n} \left|Q'_i\right| \right) \le \left|\mathrm{el}\,(\varphi) \setminus AP\right|. \tag{4}$$

Note that $P_j \subseteq Q_j$ (resp., $R_i \subseteq Q'_i$), and hence we have $|S_\varphi| \le (2^{|\mathrm{el}(\varphi)|})^2 = 4^{|\mathrm{el}(\varphi|)}$. □

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 AP), because every variable over finite domain could be *encoded* with several Boolean variables.

Let $\mathscr{C} = \langle S, \rho, I, L, \mathscr{F} \rangle$ 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_is.

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

$$s \in S \Longleftrightarrow s \Vdash \Phi_Z, \tag{5}$$

where $s \Vdash \Phi_Z$ means that Φ_Z is evaluated to be true if we assign each p_i with the value $s(p_i)$.

Let $V' = \{p'_1, \ldots, p'_n\}$, and each binary relation $\lambda \subseteq S \times S$ also has a Boolean representation Φ_λ over the variable set $V \cup V'$. That is,

$$(s_1, s_2) \in \lambda \Longleftrightarrow (s_1, s_2) \Vdash \Phi_\lambda, \tag{6}$$

where $(s_1, s_2) \Vdash \Phi_\lambda$ means that Φ_λ is evaluated to be true if we assign each p_i with $s_1(p_i)$ and assign each p'_i with $s_2(p_i)$.

Hence, all components of $\mathscr{M}$ can be encoded: I and ρ can be represented by two Boolean formulae Φ_I and Φ_ρ, respectively; we subsequently create a Boolean formula Φ_F for each $F \in \mathscr{F}$; note that the labeling function L is not concerned any longer, because the sates labeled with p can be captured by the Boolean formula p.

For example, from Theorem 7, we have that the symbolic representation of $\mathscr{T}_\varphi$ requires $2 \times |\mathrm{el}(\varphi) \setminus AP|$ *new* Boolean variables—because variables in $\mathrm{el}(\varphi) \cap AP$ 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 $\mathscr{C} = \langle S, \rho, I, L, \mathscr{F} \rangle$ and a *bound* $k \in \mathbb{N}$, one may use the formula

$$\Phi^{(0)}_I \wedge \bigwedge_{0 \le i < k} \Phi^{(i,i+1)}_\rho \wedge \bigvee_{0 \le \ell \le k} \left(\Phi^{(k,\ell)}_\rho \wedge \bigwedge_{F \in \mathscr{F}} \bigvee_{\ell \le j \le k} \Phi^{(j)}_F \right), \tag{7}$$

where $\Phi^{(j)}_I$ and $\Phi^{(j)}_F$ are, respectively, the Boolean formulae obtained from Φ_I and Φ_F by replacing each variable p with a new copy $p^{(j)}$, and $\Phi^{(i,j)}_\rho$ is obtained from Φ_ρ by replacing each p with $p^{(i)}$ and replacing each p' with $p^{(j)}$.

It can be seen that this formula is satisfiable if and only if $\mathscr{C}$ involves a fair path of the form $s_0 s_1 \cdots s_{\ell-1}(s_\ell \cdots s_k)^\omega$ (call it is of the *lasso shape*). Since that $\mathscr{L}(\mathscr{C}) \neq \emptyset$ if and only if $\mathscr{C}$ contains some lasso fair path (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 $0 \le \ell \le k$, we introduce a new variable r_ℓ. Intuitively, r_ℓ indicates that s_ℓ is a successor of s_k.

(2) For each fairness constraint $F \in \mathscr{F}$ and each $0 \le \ell \le k$, we introduce a variable $f^{(\ell)}_F$, and this variable is evaluated to be true only if there is some $\Phi^{(j)}_F$ which is evaluated to true, where $\ell \le j \le k$.

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

$$\begin{aligned} &\Phi^{(0)}_I \wedge \bigwedge_{0 \le i < k} \Phi^{(i,i+1)}_\rho \wedge \bigvee_{0 \le \ell \le k} r_\ell, \\ \Psi^{(k)}_{\mathscr{C}} = {} & \wedge \bigwedge_{0 \le \ell \le k} \left(r_\ell \longrightarrow \left(\Phi^{(k,\ell)}_\rho \wedge \bigwedge_{F \in \mathscr{F}} f^{(\ell)}_F \right) \right) \\ & \wedge \bigwedge_{F \in \mathscr{F}} \left(\bigwedge_{0 \le i < k} \left(f^{(i)}_F \longrightarrow \left(f^{(i+1)}_F \vee \Phi^{(i)}_F \right) \right) \right. \\ & \qquad \left. \wedge \left(f^{(k)}_F \longrightarrow \Phi^{(k)}_F \right) \right). \end{aligned} \tag{8}$$

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. *$\mathscr{L}(\mathscr{C}) \neq \emptyset$ if and only if $\Psi^{(k)}_{\mathscr{C}}$ 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 $\Psi_{\mathscr{C}}^k$ 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 $1 \le i \le m$. Hence, each s_i is a state of $\mathscr{C}$.

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

(ii) For each $0 \le i < k$, we have $(s_i, s_{i+1}) \Vdash \Phi_\rho^{(i,i+1)}$, and thus $(s_i, s_{i+1}) \in \rho$.

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

(iv) According to the constraint $r_\ell \longrightarrow (\Phi_\rho^{(k,\ell)} \wedge \bigwedge_{F\in\mathscr{F}} f_F^{(\ell)})$, we have the following.

(1) $e(\Phi_\rho^{(k,\ell)}) = 1$, which indicates that $(s_k, s_\ell) \in \rho$.

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

(v) For each fairness constraint $F \in \mathscr{F}$ and $0 \le i \le k$, we now inductively show that "$e(f_F^{(i)}) = 1$ implies $e(\Phi_F^{(j)}) = 1$ (alternatively, $s_j \in F$) for some $i \le j \le k$." First of all, it holds in the case of $i = k$, because we have the constraint $f^{(k)} \longrightarrow \Phi_F^{(k)}$. 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 $f_F^{(i)} \longrightarrow (f_F^{(i+1)} \vee \Phi_F^{(i)})$. Since we have shown that $e(f_F^{(\ell)}) = 1$, we can conclude that there exists some $\ell \le j \le k$ such that $s_j \in F$.

The above shows that $s_0 s_1 \cdots s_{\ell-1}(s_\ell \cdots s_k)^\omega$ is a fair path of $\mathscr{C}$, and hence $\mathscr{L}(\mathscr{C}) \ne \emptyset$.

Conversely, for the "only if" direction, it suffices to find some k and some assignment e evaluating $\Psi_{\mathscr{C}}^{(k)}$ to be true. Since $\mathscr{L}(\mathscr{C}) \ne \emptyset$, there must exist some fair path of lasso shape in $\mathscr{C}$. Without loss of generality, assume that $\sigma = s_0 s_1 \cdots s_{\ell-1}(s_\ell \cdots s_k)^\omega$ 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 $i \le k$, let $e(p_j^{(i)}) = s_i(p_j)$. Since s_0 is an initial state, according to the definition, we have $e(\Phi_I^{(0)}) = 1$. Meanwhile, because each $(s_i, s_{i+1}) \in \rho$, we have that the conjunction $\bigwedge_{0\le i\le k} \Phi_\rho^{(i,i+1)}$ is satisfied under e.

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

$$e(r_i) = \begin{cases} 1, & i = \ell, \\ 0, & i \ne \ell. \end{cases} \tag{9}$$

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

(iii) For each $F \in \mathscr{F}$ and each $0 \le i \le k$, we let

$$e\left(f_F^{(i)}\right) = \begin{cases} 1, & \text{if there is some } j \ge i \text{ such that } s_j \in F, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Then it can be directly checked that the conjunct

$$\bigwedge_{F\in\mathscr{F}} \left(\bigwedge_{0\le i<k} \left(f_F^{(i)} \longrightarrow \left(f_F^{(i+1)} \vee \Phi_F^{(i)}\right)\right) \wedge \left(f_F^{(k)} \longrightarrow \Phi_F^{(k)}\right) \right) \tag{11}$$

is evaluated to be true under e.

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

$$\bigwedge_{0\le i\le k} \left(r_i \longrightarrow \left(\Phi_\rho^{(k,i)} \wedge \bigwedge_{F\in\mathscr{F}} f_F^{(i)} \right) \right) \tag{12}$$

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

Thus, the formula $\Psi_{\mathscr{C}}^{(k)}$ is satisfiable.

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

Since we need only to concern about fair paths of the form $\sigma_1(\sigma_2)^\omega$, as pointed in [22], a possible candidate for the completeness threshold $\mathrm{CT}(\mathscr{C})$ is

$$D^I(\mathscr{C}) + |\mathscr{F}| \times D(\mathscr{C}), \tag{13}$$

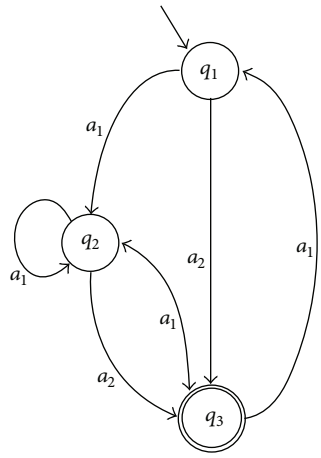
where D and D^I are, respectively, the *diameter* and the *initialized diameter* (cf. [22]). Since [22] just considers LTSs having only one fairness constraint, we here add the factor $|\mathscr{F}|$.

Observe that the part σ_2 must be enclosed in some SCC (i.e., *strongly connected component*) of $\mathscr{C}$, and we may replace $D(\mathscr{C})$ with $D(\mathscr{S})$, where $\mathscr{S}$ is the largest SCC that intersects all fairness constraints. Therefore, we may get a more compact upper bound of the completeness threshold.

Then, for a given LTS $\mathscr{M}$ and the given ETL_{l+f} formula φ, since we have shown that $\mathscr{M} \not\models \varphi$ if and only if $\mathscr{M} \times \mathscr{T}_{\neg\varphi}$ involves some fair path, we now just need to test if there is some k making $\Psi_{\mathscr{M}\times\mathscr{T}_{\neg\varphi}}^{(k)}$ satisfiable, where $k \le \mathrm{CT}(\mathscr{M} \times \mathscr{T}_{\neg\varphi})$.

Remark 9. In the case that $\mathscr{C} = \mathscr{C}_1 \times \mathscr{C}_2$, where $\mathscr{C} = \langle S, \rho, I, L, \mathscr{F}\rangle$ and $\mathscr{C}_i = \langle S_i, \rho_i, I_i, L_i, \mathscr{F}_i\rangle$ for $i = 1, 2$, we have $\Phi_I = \Phi_{I_1} \wedge \Phi_{I_2}$, $\Phi_\rho = \Phi_{\rho_1} \wedge \Phi_{\rho_2}$, and $\Phi_{\mathscr{F}} = \Phi_{\mathscr{F}_1} \cup \Phi_{\mathscr{F}_1}$, and in addition, the variable set of $\mathscr{C}$ is just the union of the variable sets of $\mathscr{C}_1$ and $\mathscr{C}_2$.

Remark 10. Actually, for an ETL_{l+f} formula φ, the symbolic representation of $\mathscr{T}_\varphi$ 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 $\mathrm{el}(\varphi)$. Subsequently, encodings of $\Phi_{\Delta_\psi^+}$ or $\Phi_{\Delta_\eta^-}$ could be naturally obtained from the underlying Boolean variables corresponding to states of automata connectives. Hence, the Boolean representation of Φ_{ρ_φ} 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 : {q2, q3};
   a2 : q3;
esac;
TRANSITIONS(q3)
case a1 : {q1, q2}; esac;
```

Figure 1: An automata connective declaration in ENuSMV.

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

Cells	BDD			BMC				
	Time (s)	Memory (MB)	C.L.	Variables	Clauses	Time (s)	Memory (MB)	C.L.
5	170.56	68.32	78	3939	8126	13.50	34.98	39
6	513.06	209.17	84	4677	9620	29.14	42.37	39
7	2372.28	976.14	90	5415	11113	41.32	58.48	39
8	8074.05	1482.15	96	6154	12609	52.45	67.32	39
9	≥5 h	≥2 G	—	6892	14102	68.17	82.13	39
10	≥5 h	≥2 G	—	7630	15599	85.13	101.21	39

4. Experimental Results

To justify our idea, we have integrated (the tool is available at https://sourceforge.net/projects/enusmv12/) the ETL_{l+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_{l+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_{l+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_{l+f} upon both star-free and nonstar-free properties.

(3) The comparison of performances with syntactic/semantic BMC of LTL and semantic BMC of ETL_{l+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_{l+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_{l+f} could be more effectively checked, we would make a comparison of BMC for ETL_f and ETL_{l+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.

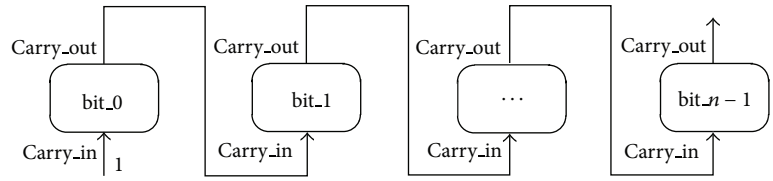


Figure 2: The circuit of MOD 2^n counter.

Table 2: Comparison of ETL_f and ETL_{l+f} with periodicity properties.

Cells	ETL_f				ETL_{l+f}			
	Max bound	Memory (MB)	Variables	Clauses	Max bound	Memory (MB)	Variables	Clauses
10	11032	279.44	154461	1180431	13009	382.784	182140	1417990
11	5138	148.22	77084	631981	6037	173.59	90570	754461
12	3702	129.571	3702	518287	4316	143.802	69072	612888

We, respectively, describe the *periodicity* property that "`bit_0` carries out at every even (except for 0) moment" with ETL_f and ETL_{l+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_{l+f}.

To compare the overhead of ETL_f and ETL_{l+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_{l+f} 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_{l+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_l and ETL_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_{l+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_{l+f} could be more efficiently verified via BMC (in comparison to our previous implementation for ETL_{l+f}).

In this paper, verification of ETL_r, namely, extended temporal logic using Büchi (alternatively, repeating) automata as connectives, has not been studied. This is partly because of the inherited 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_r.

Appendix

A. Omitted Proofs

A.1. Proof of Theorem 4. Just assume that $\sigma = s_0 s_1 \cdots \in S_\varphi^\omega$ is the corresponding fair path of $\mathscr{T}_\varphi$ having $\pi = L_\varphi(\sigma)$, where $s_i = \langle \Gamma_i; P_{1,i}, \ldots, P_{m,i}; R_{1,i}, \ldots, R_{n,i} \rangle$. We now inductively prove the following claim.

> "For each $\psi \in \mathrm{sub}(\varphi) \cup \mathrm{el}(\varphi)$, we have: $\Gamma_i \in \mathrm{sat}(\psi)$ implies $\pi, i \vDash \psi$."

(i) The basic cases are trivial.

(a) $\psi = \top$, or $\psi = \bot$. Since $\pi, i \vDash \top$ holds trivially, the case $\Gamma_i \in \mathrm{sat}(\bot)$ never happens.

Table 3: Comparison of ETL_f and ETL_{l+f} with safety properties.

Cells	ETL_f				ETL_{l+f}			
	Max bound	Memory (MB)	Variables	Clauses	Max-bound	Memory (MB)	Variables	Clauses
3	68	29.14	4002	8091	101	38.03	4958	9898
4	62	34.76	4864	9764	87	40.34	5712	11512
5	54	40.12	5170	10313	73	45.37	6072	12088

Table 4: Comparison of ETL_f and ETL_{l+f} with liveness properties.

Cells	ETL_f				ETL_{l+f}			
	Time (s)	Variables	Clauses	Memory (MB)	Time (s)	Variables	Clauses	Memory (MB)
3	25.09	2400	4796	29.66	13.78	2340	4747	26.34
4	31.42	3120	6200	38.61	20.06	3042	6115	31.38
5	42.56	3840	7604	44.28	24.70	3744	7843	36.12

(b) $\psi = p$, or $\psi = \neg p$, where $p \in AP$. Because $\Gamma_i \in \mathrm{sat}(p)$ if and only if $p \in \Gamma_i$ if and only if $p \in \pi(i)$ if and only if $\pi, i \vDash p$. Similar to show it when $\psi = \neg p$.

(ii) The following inductions are as routine.

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

(b) In the case of $\psi = \circ\psi'$, we have $\Gamma_i \in \mathrm{sat}(\psi)$ if and only if $\Gamma_{i+1} \in \mathrm{sat}(\psi')$, according to the definition of Δ_φ (note that in this case ψ is an elementary formula of φ). By induction, we have $\pi, i+1 \vDash \psi'$; hence $\pi, i \vDash \psi$.

(iii) If $\psi = \mathscr{A}^q(\varphi_1, \ldots, \varphi_k)$ is a PF-subformula, without loss of generality, assume $\psi = \psi_j$ (i.e., the jth PF-subformula of φ), where $1 \le j \le m$ and $\mathscr{A}^q = \langle\{a_1, \ldots, a_k\}, Q, \delta, q, F\rangle$, then the proof is given as follows.

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

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

(b) For each $t \le i_1 - i$, if $q_t \notin F$ and $\Gamma_{i+t} \in \mathrm{sat}(\mathscr{A}^{q_t}(\varphi_1, \ldots, \varphi_k))$, then, according to the definition of sat, there is some $1 \le k_t \le k$ and a q' such that $\Gamma_{i+t} \in \mathrm{sat}(\varphi_{k_t})$, $q' \in \delta(q_t, a_{k_t})$, and $\Gamma_{i+t} \in \mathrm{sat}(\circ\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k))$ (equivalently, $\Gamma_{i+t+1} \in \mathrm{sat}(\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k))$, as discussed above). Now, we let $q_{t+1} = q'$.

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 $\Gamma_t \in \mathrm{sat}(\mathscr{A}^{q_t}(\varphi_1, \ldots, \varphi_k))$. Since $P_{j,i_1} = \emptyset$, according to the definition of $\Delta^+_{\psi_j}$, we have $q_t \in P_{j,i_1+1} = P_{j,i+t}$.

(b) For each $i_1 - i < t \le i_2 - i$, if $q_t \in P_{j,i+t}$ and $q_t \notin F$, according to $\Delta^+_{\psi_j}$, there is some $1 \le k_t \le k$, such that $\Gamma_{i+t} \in \mathrm{sat}(\varphi_{k_t})$, and there is some $q' \in P_{j,i+t+1} \cap \delta(q_t, a_{k_t})$. Now, we let $q_{t+1} = q'$.

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

Thus, $\mathscr{A}$ have some accepting prefix $a_{k_0} a_{k_1} \cdots a_{k_\ell}$ of some infinite word. On the other hand, by induction, $\Gamma_{i+t} \in \mathrm{sat}(\varphi_{k,t})$ implies $\pi, i+t \vDash \varphi_{k_t}$. Hence, we have $\pi, i \vDash \psi$ by definition.

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

Assume by contradiction it is not the case; then $\pi, i \vDash \mathscr{A}^q(\varphi_1, \ldots, \varphi_k)$ holds. Therefore, there exist some accepting prefix $a_{k_0} a_{k_1} \cdots a_{k_\ell}$ of some infinite word and a state sequence $q_0 q_1 \cdots q_\ell q_{\ell+1}$ such that

(a) $q_0 = q$, $q_{\ell+1} \in F$ and each $q_{t+1} \in \delta(q_t, a_{k_t})$,

(b) for each $t \le \ell$, we have $\pi, i+t \vDash \varphi_{k_t}$.

Then, we have the following facts.

(a) $\Gamma_{i+\ell+1} \in \mathrm{sat}(\mathscr{A}^{q_{\ell+1}}(\varphi_1, \ldots, \varphi_k)) = 2^{\mathrm{el}(\varphi)}$, because $q_{\ell+1} \in F$.

(b) Assume that $\Gamma_{i+t+1} \in \mathrm{sat}(\mathscr{A}^{q_{t+1}}(\varphi_1, \ldots, \varphi_k))$, where $t \le \ell$; then $\Gamma_{i+t} \in \mathrm{sat}(\circ\mathscr{A}^{q_{t+1}}(\varphi_1, \ldots, \varphi_k))$. In addition, $\pi, i+t \vDash \varphi_{k_t}$ implies that $\Gamma_{i+t} \in \mathrm{sat}(\varphi_{k_t})$—otherwise (it is not hard to show that if there is an infinite path in $\mathscr{T}_\varphi$ starting from s_{i+t}, then, for each η, either $\Gamma_{i+t} \in \mathrm{sat}(\eta)$ or $\Gamma_{i+t} \in \mathrm{sat}(\neg\eta)$ holds), $\Gamma_{i+t} \in \mathrm{sat}(\neg\varphi_{k_t})$; then we have $\pi, i+t \vDash \neg\varphi_{k_t}$ by induction contradicts! Hence, by the definition of sat, we have $\Gamma_{i+t} \in \mathrm{sat}(\mathscr{A}^{q_t}(\varphi_1, \ldots, \varphi_k))$.

Then, we have $\Gamma_i \in \mathrm{sat}(\mathscr{A}^{q_0}(\varphi_1, \ldots, \varphi_k))$ when $t = 0$. However, it contradicts the premiss $\Gamma_i \in \mathrm{sat}(\neg\mathscr{A}^q(\varphi_1, \ldots, \varphi_n))$, because $q_0 = q$. Therefore, the assumption $\pi, i \vDash \mathscr{A}^q(\varphi_1, \ldots, \varphi_k)$ is a falsity, and this implies that $\pi, i \vDash \psi$ holds.

(v) If $\psi = \mathscr{A}^q(\varphi_1, \ldots, \varphi_k)$ is a PL-subformula of φ, where $\mathscr{A}^q = (\{a_1, \ldots, a_k\}, Q, \delta, q, F)$, then we have the following proof in the case that $\Gamma_i \in \mathrm{sat}(\psi)$.

(a) Let $q_0 = q$, and $\Gamma_i \in \mathrm{sat}(\mathscr{A}^{q_0}(\varphi_1, \ldots, \varphi_k))$.

(b) For each $t \ge 0$, assume that $\Gamma_{i+t} \in \mathrm{sat}(\mathscr{A}^{q_t}(\varphi_1, \ldots, \varphi_k))$; then by the definition of sat, there is some $1 \le k_t \le k$, such that $\Gamma_{i+t} \in \mathrm{sat}(\varphi_{k_t})$, and there is

Table 5: Comparison of syntactic/semantic LTL BMC with liveness properties.

Cells	Syntactic LTL BMC encoding				Semantic LTL BMC encoding			
	Time (s)	Variables	Clauses	Memory (MB)	Time (s)	Variables	Clauses	Memory (MB)
3	9.34	2052	5474	32.66	8.59	2128	4291	24.79
4	16.71	2736	6806	35.74	10.87	2812	5623	28.44
5	24.06	3420	8138	37.02	15.90	3469	6955	32.28

some $q' \in \delta(q_t, a_{k_t})$ such that $sat(\circ\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k))$ (equivalently, $\Gamma_{i+t+1} \in \text{sat}(\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k))$ holds). Now, we let $q_{t+1} = q'$.

Hence, we have an accepting run of $\mathscr{A}^q$ over the infinite word $a_{k_0}a_{k_1}\cdots$. Meanwhile, by induction, $\Gamma_{i+t} \in \text{sat}(\varphi_{k_t})$ implies that $\pi, i+t \vDash \varphi_{k_t}$. By definition, we have $\pi, i \vDash \psi$ in this case.

(vi) Lastly, consider the case $\psi = \neg\mathscr{A}^q(\varphi_1, \ldots, \varphi_k)$, where $\mathscr{A}^q = \langle\{a_1, \ldots, a_k\}, Q, \delta, q, F\rangle$ is a looping acceptance automaton.

Suppose that $\Gamma_i \in \text{sat}(\psi)$, and assume by contradiction that $\pi, i \nvDash \psi$; then there is an infinite word $w = a_{k_0}a_{k_1}\cdots \in \mathscr{L}(\mathscr{A}^q)$ and $\pi, i+t \vDash \varphi_{k_t}$ for each $t \geq 0$. Also, let the corresponding run of $\mathscr{A}^q$ on w is $q_0q_1\cdots$, where $q_0 = q$.

Without loss of generality, suppose that $\psi = \neg\eta_j$ (i.e., the jth NL-subformula); since σ is a fair path, there must exist some $i_2 > i_1 > i$ such that $R_{j,i_1} = R_{j,i_2} = \emptyset$ and $R_{j,c} \neq \emptyset$ for each $i_1 < c < i_2$. Then

(a) $\Gamma_i \in \text{sat}(\neg\mathscr{A}^{q_0}(\varphi_1, \ldots, \varphi_k))$, because $q_0 = q$,

(b) as discussed before (in the case of NF-subformula), $\pi, i+t \vDash \varphi_{k_t}$ implies that $\Gamma_{i+t} \in \text{sat}(\varphi_{k_t})$. For each $t > 0$, if $\Gamma_{i+t} \in \text{sat}(\neg\mathscr{A}^{q_t}(\varphi_1, \ldots, \varphi_k))$ holds, according to the definition of sat, for each $q' \in \delta(q_t, a_{k_t})$, we have $\Gamma_{i+t} \notin \text{sat}(\circ\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k))$. Note that $q_{t+1} \in \delta(q_t, a_k)$; hence $\Gamma_{i+t+1} \in \text{sat}(\neg\mathscr{A}^{q_{t+1}}(\varphi_1, \ldots, \varphi_k))$.

Therefore, we have $\Gamma_{i+t} \in \text{sat}(\neg\mathscr{A}^{q_t}(\varphi_1, \ldots, \varphi_k))$ for each $t \geq 0$.

According to the definition of $\Delta^-_{\neg\eta_j}$, we have

(a) $q_{i_1+1-i} \in R_{j,i_1+1}$, since $R_{j,i_1} = \emptyset$,

(b) for each t having $q_t \in R_{j,i+t}$, since $\Gamma_{i+t} \in \text{sat}(\varphi_{k_t})$, then $q' \in \delta(q_t, a_{k_t})$ implies $q' \in R_{j,i+t+1}$. Therefore, $q_{t+1} \in R_{j,i+t+1}$.

The above induction implies that $q_t \in R_{j,i+t}$ for each $t \geq i_1 - i + 1$. However, it is impossible since $R_{j,i_2} = \emptyset$. Hence, the assumption $\pi, i \vDash \mathscr{A}^q(\varphi_1, \ldots, \varphi_k)$ is incorrect, which implies that $\pi, i \vDash \psi$ also holds in this case.

Clearly, the above induction is complete. Because we require that $\Gamma_0 \in \text{sat}(\varphi)$, we have $\pi, 0 \vDash \varphi$.

A.2. Proof of Theorem 5. Suppose that $\pi \vDash \varphi$ to show $\pi \in \mathscr{L}(\mathscr{T}_\varphi)$, we need to first construct an infinite state sequence $\sigma = s_0s_1\cdots$, where $s_i = \langle\Gamma_i; P_{1,i}, \ldots, P_{m,i}; R_{1,i}, \ldots, R_{n,i}\rangle \in S_\varphi$, and then we will show that σ is a fair path of $\mathscr{T}_\varphi$ and $\pi = L_\varphi(\sigma)$.

(i) The construction of Γ_i is as follows.

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 ψ the following hold.

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

(b) If $\psi = p \in AP$, then $\psi \in \text{el}(\varphi)$. Thus, $\Gamma_i \in \text{sat}(p)$ if and only if $p \in \Gamma_i$ 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 $\Gamma_i \in \text{sat}(\circ\psi')$ if and only if $\psi \in \Gamma_i$ if and only if $\pi, i \vDash \psi$.

(d) If $\psi = \psi_1 \wedge \psi_2$, then $\Gamma_i \in \text{sat}(\psi)$ if and only if $\Gamma_i \in \text{sat}(\psi_1)$ and $\Gamma_i \in \text{sat}(\psi_2)$. By induction, we have $\pi, i \vDash \psi_1$ and $\pi, i \vDash \psi_2$ that is, $\pi, i \vDash \psi$. Similar to show the case of $\psi = \psi_1 \vee \psi_2$.

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

$$\mathscr{A}^q(\varphi_1, \ldots, \varphi_k) \longleftrightarrow \bigvee_{1\leq t\leq k}\left(\varphi_t \wedge \bigvee_{q'\in\delta(q,a_t)} \circ\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k)\right). \tag{A.1}$$

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

$$\Gamma_i \in \bigcup_{1\leq t\leq k}\left(\text{sat}(\varphi_t) \cap \bigcup_{q'\in\delta(q,a_t)} \text{sat}\left(\circ\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k)\right)\right) = \text{sat}\left(\mathscr{A}^q(\varphi_1, \ldots, \varphi_k)\right). \tag{A.2}$$

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

$$\mathscr{A}^q(\varphi_1, \ldots, \varphi_k) \longleftrightarrow \top. \tag{A.3}$$

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

(f) If $\psi = \neg\mathscr{A}^q(\varphi_1, \ldots, \varphi_k)$, then $\Gamma_i \in \text{sat}(\psi)$ if and only if $\Gamma_i \notin \text{sat}(\mathscr{A}^q(\varphi_1, \ldots, \varphi_k))$ if and only if $\pi, i \nvDash \mathscr{A}^q(\varphi_1, \ldots, \varphi_k)$ if and only if $\pi, i \vDash \psi$.

Now, we have the following properties.

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

(2) For each i and each $\circ\psi \in \mathrm{el}(\varphi)$, we have $\Gamma_i \in \mathrm{sat}(\circ\psi)$ if and only if $\pi, i \vDash \circ\psi$ if and only if $\pi, i+1 \vDash \psi$ if and only if $\Gamma_{i+1} \in \mathrm{sat}(\psi)$.

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

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

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

(a) We will find a series of "*key positions*" $\ell_0, \ell_1, \ldots$, having $P_{j,\ell_t} = \emptyset$ for each ℓ_t. First, let $\ell_0 = 0$, and, for each $t \ge 0$, once ℓ_t has been determined, we use the following two steps to determine ℓ_{t+1} and each $P_{j,c}$ for $\ell_t < c < \ell_{t+1}$.

(b) Let $P_{j,\ell_t} = \emptyset$, and let $P_{j,\ell_t+1} = \{q' \mid \pi, \ell_t + 1 \vDash \mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k)\}$. For each $q' \in P_{j,\ell_t+1}$, there must exit a finite word $w_{q'} = a_{k_0} a_{k_1} \cdots a_{k_s}$ and a finite state sequence $\chi_{q'} = q_0 q_1 \cdots q_{s+1}$ such that $q_0 = q'$, $q_{s+1} \in F$, and, for each $c \le s$, we have $q_{c+1} \in \delta(q_c, a_{k_c})$ and $\pi, \ell_t + c + 1 \vDash \varphi_{k_s}$ (equivalently, $\Gamma_{\ell_t+c+1} \in \mathrm{sat}(\varphi_{k_s})$).

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

$$P_{j,c} = \left\{\chi_{q'}\,(c - \ell_t - 1) \,\middle|\, \chi_{q'}\right| > c - \ell_t - 1\right\} \tag{A.4}$$

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

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

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

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

(a) First, let $\ell_0 = 0$, and, for each $t \ge 0$, once ℓ_t is decided, we use the following steps to determine ℓ_{t+1} and each set $R_{j,c}$ for $\ell_t < c < \ell_{t+1}$.

(b) Let $R_{j,\ell_t} = \emptyset$, and let $R_{j,\ell_t+1} = \{q' \mid \Gamma_{\ell_t+1} \in \mathrm{sat}(\neg\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k))\}$. For each $c \ge \ell_t + 1$, let

$$R_{j,c+1} = \bigcup_{q' \in R_{j,c}} \left\{\delta\left(q', a_s\right) \mid 1 \le s \le k, \Gamma_c \in \mathrm{sat}\left(\varphi_s\right)\right\}. \tag{A.5}$$

(c) Since we have shown that $\Gamma, i \vDash \psi$ if and only if $\pi, i \vDash \psi$ for every $\psi \in \mathrm{sub}(\varphi) \cup \mathrm{el}(\psi)$, it is not hard to show that if $R_{j,\ell_t+1} \ne \emptyset$, then there is some $c > \ell_t + 1$ such that $R_{j,c} = \emptyset$—otherwise (because, if $R_{j,c} \ne \emptyset$ for each $t > \ell_t$, using König's lemma, we may find a run of $\mathscr{A}^{q'}$ over some infinite word), we can show $\pi, \ell_t + 1 \vDash \mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k)$ for some $q' \in R_{j,\ell_t+1}$, and this implies that $\Gamma_{\ell_t+1} \in \mathrm{sat}(\mathscr{A}^{q'}(\varphi_1, \ldots, \varphi_k))$, which is contradiction!

If $R_{j,\ell_t+1} = \emptyset$, let $\ell_{t+1} = \ell_t + 1$; otherwise, let $\ell_{t+1} = c$. According to the construction, it can be examined that $((\Gamma_i, R_{j,i}), (\Gamma_{i+1}, R_{j,i+1})) \in \Delta^-_{\neg\eta_j}$ for each i. In addition, we have $R_{j,\ell_t} = \emptyset$ for each $t \ge 0$.

Taking all the above into account, by definition, we may conclude that $(s_i, s_{i+1}) \in \rho_\varphi$ for every $i \ge 0$; hence σ is a fair path of $\mathscr{T}_\varphi$. According to the construction of Γ_is, we have that $\pi = L_\varphi(\sigma)$; hence $\pi \in \mathscr{L}(\mathscr{T}_\varphi)$.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments on a previous version of this paper. This work is supported by the NSFC in China under grant number 61103012, 61272335, 61133007, 91118007, 61120106006, the 863 Program 2011AA010106, 2012AA011201, and the Program for New Century Excellent Talents in University.

References

[1] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Transactions on Computers C*, vol. 35, no. 8, pp. 677–691, 1986.

[2] K. L. McMillan, *Symbolic model checking, an approach to the state explosion problem [Ph.D. thesis]*, Carnegie Mellon University, Pittsburgh, Pa, USA; Kluwer Academic, Boston, Mass, USA, 1993.

[3] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *Proceedings of the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '99)*, vol. 1579 of *Lecture Notes in Computer Science*, pp. 193–207, Springe, Berlin, Germany, 1999.

[4] P. Wolper, "Temporal logic can be more expressive," *Information and Control*, vol. 56, no. 1-2, pp. 72–99, 1983.

[5] A. Pnueli, "Linear and branching structures in the semantics and logics of reactive systems," in *International Colloquium on Automata, Language and Programming*, W. Brauer, Ed., vol. 194 of *Lecture Notes in Computer Science*, pp. 15–32, Springer, Berlin, Germany, 1985.

[6] O. Lichtenstein, A. Pnueli, and L. Zuck, "The glory of the past," in *Proceedings of the Workshop on Logics of Programs*, vol. 193 of *Lecture Notes in Computer Science*, pp. 97–107, Springer, Brooklyn, NY, USA, 1985.

[7] Accellera, "Accellera property languages reference manual," June 2004, http://www.eda.org/vfv/docs/PSL-v1.1.pdf.

[8] B. Banieqbal and H. Barringer, "Temporal logic with fixed points," in *Temporal Logic in Specification*, vol. 398 of *Lecture Notes in Computer Science*, pp. 62–74, Springer, Berlin, Germany, 1987.

[9] A. P. Sistla, M. Y. Vardi, and P. Wolper, "The complementation problem for Büchi automata with applications to temporal logic," *Theoretical Computer Science*, vol. 49, no. 2-3, pp. 217–237, 1987.

[10] M. Leucker and C. Sanchez, "Regular linear temporal logic," in *Proceedings of the 4th International Conference on Theoretical Aspects of Computing*, vol. 4711 of *Lecture Notes in Computer Science*, pp. 291–305, Springer, Berlin, Germany, 2007.

[11] R. Armoni, L. Fix, A. Flaisher et al., "The ForSpec temporal logic: a new temporal property-specification language," in *Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS '02)*, vol. 2280 of *Lecture Notes in Computer Science*, pp. 296–311, Springer, Berlin, Germany, 2002.

[12] I. Beer, S. Ben-David, C. Eisner, D. Fisman, A. Gringauze, and Y. Rodeh, "The temporal logic sugar," in *Proceedings of the 13th International Conference on Computer Aided Verification*, G. Berry, H. Comon, and A. Frinkel, Eds., vol. 2102 of *Lecture Notes in Computer Science*, pp. 363–367, Springer, London, UK, 2001.

[13] M. Y. Vardi and P. Wolper, "Reasoning about infinite computations," *Information and Computation*, vol. 115, no. 1, pp. 1–37, 1994.

[14] J. R. Büchi, "On a decision method in restricted second order arithmetic," in *Proceedings of the International Congresses in Logic, Methodology and Philosophy of Science 1960*, pp. 1–12, Stanford University Press, Palo Alto, Calif, USA, 1962.

[15] W. Liu, J. Wang, and Z. Wang, "Symbolic model checking of ETL," *Journal of Software*, vol. 20, no. 8, pp. 2015–2025, 2009.

[16] W. Liu, J. Wang, H. Chen, X. Ma, and Z. Wang, "Symbolic model checking APSL," *Frontiers of Computer Science in China*, vol. 3, no. 1, pp. 130–141, 2009.

[17] M. Jehle, J. Johannsen, M. Lange, and N. Rachinsky, "Bounded model checking for all regular properties," *Electronic Notes in Theoretical Computer Science*, vol. 144, no. 1, pp. 3–18, 2006.

[18] A. Pnueli and A. Zaks, "PSL model checking and run-time verification via testers," *Formal Methods*, Springer, Berlin, Germany, vol. 4085, pp. 573–586, 2006.

[19] A. Cimatti, M. Roveri, S. Semprini, and S. Tonetta, "From PSL to NBA: a modular symbolic encoding," in *Formal Methods in Computer Aided Design (FMCAD '06)*, Lecture Notes in Computer Science, pp. 125–133, Springer, 2006.

[20] E. M. Clarke, O. Grumberg, and K. Hamaguchi, "Another look at LTL model checking," in *Computer Aided Verification, 6th International Conference (CAV '94)*, vol. 818 of *Lecture Notes in Computer Science*, pp. 415–427, Springer, Berlin, Germany, 1994.

[21] A. Biere, K. Heljanko, T. Junttila, T. Latvala, and V. Schuppan, "Linear encodings of bounded LTL model checking," *Logical Methods in Computer Science*, vol. 2, no. 5, article 5, 2006.

[22] E. Clarke, D. Kroening, J. Ouaknine, and O. Strichman, "Completeness and complexity of bounded model checking," in *Verification, Model Checking, and Abstract Interpretation (VMCAI '04)*, vol. 2937 of *Lecutre Notes in Computer Science*, pp. 85–96, Springer, Berlin, Germany, 2004.

[23] T. Latvala, A. Biere, K. Heljanko, and T. Junttila, "Simple bounded LTL model checking," in *Formal Methods in Computer-Aided Design (FMCAD '04)*, A. Hu and A. Martin, Eds., vol. 3312 of *Lecture Notes in Computer Science*, pp. 186–200, Springer, Berlin, Germany, 2004.

[24] T. Latvala, A. Biere, K. Heljanko, and T. Junttila, "Simple is better: efficient bounded model checking for past LTL," in *Verification, Model Checking, and Abstract Interpretation (VMCAI '05)*, vol. 3385 of *Lecture Notes in Computer Science*, pp. 380–395, Springer, Berlin, Germany, 2005.

[25] R. Cavada, A. Cimatti, C. A. Jochim et al., "NuSMV 2. 5 user manual," April 2010, http://nusmv.fbk.eu/NuSMV/userman/v25/nusmv.pdf.

[26] K. Heljanko, T. Junttila, and T. Latvala, "Incremental and complete bounded model checking for full PLTL," in *Proceedings of the 17th International Conference of Computer Aided Verification (CAV '05)*, K. Etessami and S. K. Rajamani, Eds., vol. 3576 of *Lecture Notes in Computer Science*, pp. 98–111, Springer, Berlin, Germany, 2005.

[27] Q. Yan, "Lower bounds for complementation of ω-automata via the full automata technique," *Journal of Logical Methods in Computer Science*, vol. 4, no. 1, article 5, 2008.

[28] O. Kupferman and M. Y. Vardi, "Weak alternating automata are not that weak," *ACM Transactions on Computational Logic*, vol. 2, no. 3, pp. 408–429, 2001.

[29] E. Friedgut, O. Kupferman, and M. Y. Vardi, "Büchi complementation made tighter," in *Automated Technology for Verification and Analysis (ATVA '06)*, vol. 3299 of *Lecture Notes in Computer Science*, pp. 64–78, Springer, Berlin, Germany, 2004.

[30] S. Schewe, "Büchi complementation made tight," in *STACS 2009: 26th International Symposium on Theoretical Aspects of Computer Science*, vol. 3, pp. 661–672, IBFI, 2009.