Verify LTL with Fairness Assumptions Efficiently

Yong Li*, Lei Song*, Yuan Feng‡, Lijun Zhang§

*State Key Laboratory of Computer Science, Institute of Software, CAS, China
‡University of Chinese Academy of Sciences, China
§University of Technology Sydney, Australia

Abstract—This paper deals with model checking problems with respect to LTL properties under fairness assumptions. We first present an efficient algorithm to deal with a fragment of fairness assumptions and then extend the algorithm to handle arbitrary ones. Notably, by making use of some syntactic transformations, our algorithm avoids to construct corresponding Büchi automata for the whole fairness assumptions, which can be very large in practice. We implement our algorithm in NuSMV and consider a large selection of formulas. Our experiments show that in many cases our approach exceeds the automata-theoretic approach up to several orders of magnitude, in both time and memory.

I. INTRODUCTION

Linear Temporal Logic (LTL) [24] has been shown to be a proper specification language. As a result, for verifying reactive systems, model checkers for LTL, like Spin [13] and NuSMV [7], have been applied in practice successfully. To verify whether or not a system satisfies an LTL formula, the classical automata-theoretic approach [32] is usually adopted: Firstly, a Büchi automaton is built which accepts all executions violating the LTL formula; Secondly, a product system is built from the original system and the Büchi automaton; Finally, the problem is reduced to finding an accepting path in the product system. Since in the worst case the constructed Büchi automaton can be exponentially larger than the LTL formula, both time and space complexity of the algorithm in [32] is exponential with respect to the size of the LTL formula. The complexity of this algorithm is shown to be PSPACE-complete in [30]. Even if we restrict to a small subset of LTL formulas (those only containing eventual modality F), it is still NP-complete. On the other side, due to the popularity of LTL, many ideas have been proposed optimizing the construction of Büchi automata, see e.g. [10], [14], [16], [31], [19], [28].

The classification of properties into different categories is pivotal for efficient verification of reactive systems. In the seminal paper [22], Lamport introduced the notions of safety and liveness properties, where “safety” properties assert something “bad” never happens, while “liveness” properties require something “good” will eventually happen. These notions were later formalized by Alpern and Schneider in [11]. Properties were further classified into strong safety and absolute liveness in [29], and fair properties. The notion of fairness is important for verifying liveness in reactive systems to remove unrealistic behaviors [15].

In practice, fairness assumptions can have a great impact on the performance in many cases. For instance in the binary semaphore protocol [17], the fairness assumption that whenever a process is ready, it will have a chance to enter the critical

section, is given by: \( \bigwedge_{1 \leq i \leq n} (\text{GF}_{\text{enter}_i} \rightarrow \text{GF}_{\text{critical}_i}) \) (G and F denote “always” and “eventually”, respectively) with \( n \) being the number of processes. When \( n = 5 \), the corresponding Büchi automaton generated by LTL3BA [16] has more than 300 states and 1 million transitions. Therefore, model checking formulas under such an assumption will be time and memory consuming even when given formulas are simple.

In this paper we propose a novel algorithm to verify fairness as well as general properties with fairness assumptions. We do not only consider simple fairness formulas as mentioned above, but also consider more complex fairness with nested modalities like \( \text{FG}(a \lor \text{F}b) \). Moreover, we extend the notion of fairness assumptions to full LTL formulas, which allows us to specify some fairness assumption like “a and \( \text{X}(b \lor c) \) holds infinitely often”. Notably, our algorithm relies on a syntactic transformation and avoids to construct a Büchi automaton for the whole fairness. The approach is presented in three steps:

- We first restrict to fairness with only F and G modalities, for which our syntactic transformation can completely avoid Büchi automata construction. For this setting our approach achieves a speedup of four orders of magnitudes in some examples.
- We then extend the algorithm to deal with fair formulas of full LTL. The idea is to transform a fair formula into an equivalent one in disjunctive norm form, each sub-formula of which can be handled by specific and efficient algorithms. Even though we may still resort to the automata-theoretic approach for some sub-formulas, they are often much smaller than the original one.
- Finally, we show our approach can be adapted to accelerate the verification of generic LTL formulas under fairness assumptions.

We have implemented the algorithm in NuSMV and compared it with the classical algorithm. Our experimental results show that for many cases while NuSMV runs out of time or memory quickly, our algorithm terminates within seconds using memory less than 100 MB. The main reason is that after the syntactical transformation, we can avoid constructing Büchi automata for many sub-formulas. Even for those where Büchi automata construction is inevitable, their corresponding automata are relatively small and can be constructed efficiently. It should be pointed out, however, that the syntactical transformation may also cause exponential blow-ups in the

Interested readers can try the online LTL translator available at:
http://spot.lip6.fr/ltl2tgba.html
length of given formulas. Hence, as the experimental results show, our algorithm may be much slower than NuSMV in some cases. We then further discuss and characterize the formulas for which our approach provides better performance.

a) Related Work: There is a plenty of work on optimizing verification of LTL (or its sub-logic). Here we only briefly recall a few closely related works. In [4], specialized algorithms are proposed to deal with LTL properties, which can be represented by either terminal or weak automata. Compared to general algorithms, specialized algorithms improve the worst-case time complexity by a constant factor. This result is further formalized in [6], where it is shown that terminal and weak automata correspond to guarantee properties (something happens eventually) and persistence properties (something always happens eventually), respectively. For guarantee properties, model checking algorithm reduces to the reachability of an accepting state, while for persistence properties, it reduces to finding a fully accepting cycle, i.e., all states on it are accepting. Furthermore, a decision algorithm is proposed in [6] to check whether an LTL formula is a guarantee or persistence property. For properties which are neither guaranteed nor persistent, the general algorithm has to be used. One exception is [26], where a decomposition algorithm is proposed for strong automata, which are neither terminal nor weak. The idea is to decompose a strong automaton into three sub-automata, which are terminal, weak, and strong, respectively. Then specialized algorithms can be used to check the terminal and weak sub-automata. Since the strong sub-automaton is smaller than the original automaton, decomposition always speeds up the verification according to the experiment in [26].

Differently, our algorithm performs decomposition syntactically on given formulas, hence we do not need to build their corresponding Büchi automata at the beginning. While the specialized algorithm in [4], [6], [26] is automata-based, Büchi automata have to be built beforehand, which may take a significant amount of time and memory, especially when the given formula is long [17]. Moreover, our algorithm works for arbitrary fairness including those which are neither guaranteed nor persistent.

b) Organization of the paper: Section II introduces some definitions and notations used throughout the paper. The algorithm is presented in Section III. We demonstrate the efficiency of our algorithm via experiment in Section IV. Finally, we conclude our paper in Section VI.

All missing proofs can be found in the appendix.

II. PRELIMINARIES

We shall first introduce some preliminary definitions and notations and then present the syntax and semantics of LTL.

Let $X$ be a finite set of elements and $\xi = x_0x_1... \in X^*$ a finite sequence of elements in $X$. For each $\xi \in X^*$, we let $|\xi| = i + 1$ denote its length. An infinite sequence $\xi \in X^\omega$ is cyclic if there exists $\xi' \in X^*$ for some $i$ such that $\xi = (\xi')^i$, i.e., repeating $\xi'$ for infinite times. Let $\xi[i] = x_i$ denote the $(i + 1)$-th element on $\xi$ if it exists. We shall write $\xi'[i]$ to denote the prefix of $\xi$ ending at the $(i + 1)$-th element, while $\xi[i]$ the suffix of $\xi$ starting from the $(i + 1)$-th element. Let $\xi \in X^*$ and $\xi' \in X^\omega$. Then $\xi \cdot \xi'$ denotes the infinite sequence obtained by attaching $\xi'$ to the end of $\xi$.

We will fix a finite set of atomic propositions, denoted $AP$ and ranged over by $a, b, c, ...$, throughout the remainder of the paper. The syntax of LTL is given by the following grammar:

$$\phi, \psi ::= a | \neg a | \phi_1 \land \phi_2 | \phi_1 \lor \phi_2 | X \phi | \phi_1 U \phi_2 | \phi_1 W \phi_2$$

where $a \in AP$ and $\phi, \psi, \phi_1$, and $\phi_2$ range over LTL formulas. As usual, we introduce some abbreviations: $1 = a \lor \neg a$ and $0 = a \land \neg a$ denote True and False respectively, while $F \phi = 1U\phi$ (eventually $\phi$), $G \phi = \phi W0$ (always $\phi$), and $(\phi_1 = \Rightarrow \phi_2) = (\neg \phi_1 \lor \phi_2)$. Let $l, l_1, l_2, ...$ range over propositional formulas, i.e., formulas defined by: $l ::= a | \neg a | l_1 \land l_2 | l_1 \lor l_2$.

Given an infinite sequence of sets of atomic propositions $\rho = \rho_0 \rho_1 ... \in (2^{AP})^\omega$ and an LTL formula $\phi$, we say $\rho$ satisfies $\phi$, written as $\rho \models \phi$, if:

$$\begin{align*}
\rho &\models a & \text{iff} & a \in \rho[0] \\
\rho &\models X \phi & \text{iff} & \rho[1] \models \phi \\
\rho &\models \phi_1 U \phi_2 & \text{iff} & \exists i \geq 0.(\rho[i] \models \phi_2 \land \forall 0 \leq j < i.\rho[j] \models \phi_1) \\
\rho &\models \phi_1 W \phi_2 & \text{iff} & (\forall i \geq 0.\rho[i] \models \phi_1) \lor (\rho \models \phi_1 U \phi_2)
\end{align*}$$

All other connectives are defined in a standard way. For formulas $\phi$ and $\psi$, we say that $\phi$ and $\psi$ are semantically equivalent, denoted $\phi \equiv \psi$, if $\rho \models \phi$ iff $\rho \models \psi$ for any $\rho \in (2^{AP})^\omega$.

Here we only define LTL formulas in positive normal form, in the sense that the negation operator can only be applied to atomic propositions. However, it is well-known that any LTL formula can be transformed into an equivalent one in positive normal form, using $\neg(X \psi) \equiv X(\neg \psi)$ and the following duality laws:

$$\begin{align*}
\neg(\phi_1 U \phi_2) &\equiv (\phi_1 \land \neg \phi_2)W(\neg \phi_1 \land \neg \phi_2) \\
\neg(\phi_1 W \phi_2) &\equiv (\phi_1 \land \neg \phi_2)U(\neg \phi_1 \land \neg \phi_2)
\end{align*}$$

Fairness assumptions are critical to rule out unrealistic behaviors when performing verification; see for instance [23], [15]. Formally, fairness is a fragment of LTL, which can be defined as follows:

**Definition 1** ([29]). An LTL formula $\phi$ is a fairness iff for any $\rho \in (2^{AP})^\omega$,

1. the set of sequences satisfying $\phi$ is closed under suffixes, i.e., $\rho \models \phi$ implies $\rho[i] \models \phi$ for any $i \geq 0$;
2. the set of sequences satisfying $\phi$ is closed under prefixes, i.e., $\rho \models \phi$ implies $\rho[1] \cdot \rho \models \phi$ for any $\rho_1 \in (2^{AP})^*$. 

We shall refer properties defined in Definition 1 as fair formulas or fairness in the following. According to Definition 1, the following lemma is straightforward:

**Lemma 1.** $\phi$ is a fairness iff $\phi \equiv G\phi$ and $\phi \equiv F\phi$.

As a result of Lemma 1, we can add any number of $F$ and $G$ in front of a fairness without changing its semantics. For instance, fairness $Fa \lor G \neg a$ is equivalent to $GF(Fa \lor G \neg a)$. 
As usual we consider models given as Kripke structures, which are formally defined as follows:

**Definition 2.** A Kripke structure is a tuple $\mathcal{K} := (S, \bar{s}, T, AP, L)$ where $S$ is a finite set of states, $\bar{s} \in S$ is the initial state, $T \subseteq S \times S$ is a set of transitions, and $L : S \rightarrow 2^{AP}$ is a labeling function. We assume that for each $s \in S$, there exists $s' \in S$ such that $(s, s') \in T$.

We fix a Kripke structure $\mathcal{K} = (S, \bar{s}, T, AP, L)$ throughout the remainder of the paper. Let $r, s, t, \ldots$ range over $S$. Let $\text{Paths}^i(s) \subseteq S^i$ denote the set of infinite paths starting from $s$ such that $\pi \in \text{Paths}^i(s)$ iff $\pi[0] = s$ and for any $i \geq 0$, $(\pi[i], \pi[i+1]) \in T$. Similarly, we can define $\text{Paths}^i(\mathcal{K})$, i.e., finite paths in $\mathcal{K}$ starting from $s$. Let $\text{Paths}^i(\mathcal{K}) = \text{Paths}^i(\bar{s})$ and $\text{Paths}^*(\mathcal{K}) = \text{Paths}^*(\bar{s})$. Given $\pi \in S^i$, let $\text{trace}(\pi)$ denote the trace of $\pi$ such that $\text{trace}(\pi)[i] = L(\pi[i])$ for all $i \geq 0$, i.e., $\text{trace}(\pi)$ denotes the sequence of labels of states in $\pi$. For an LTL formula $\varphi$, we write $\pi \models \varphi$ iff $\text{trace}(\pi) \models \varphi$; $s \models \varphi$ iff $\varphi$ for all $\pi \in \text{Paths}^i(s)$; $\mathcal{K} \models \varphi$ iff $\bar{s} \models \varphi$. Given an LTL formula $\varphi$ and a fairness $\varphi_f$, $\mathcal{K}$ satisfies $\varphi$ under the assumption $\varphi_f$, i.e., $\mathcal{K} \models (\varphi_f \implies \varphi)$.

Moreover, we directly conclude the corollary below:

**Corollary 1.** Let $\pi \in \text{Paths}^i(\mathcal{K})$ and $\varphi$ a fairness. Then for any index $j \geq 0$, $\pi \models \varphi$ iff $\pi|_j \models \varphi$.

**Proof:** Since $\varphi$ is a fairness, $\varphi \equiv \text{GF} \varphi$. Thus $\pi \models \varphi$ implies $\pi|_j \models \varphi$ for any index $j$. On the other hand, $\pi|_j \models \varphi$ implies $\pi \models \text{FG} \varphi$. Then we conclude $\pi \models \varphi$ by $\text{FG} \varphi \equiv \varphi$. $\square$

Intuitively, we can safely consider only the suffixes of the paths when the given formula is a fairness.

## III. Model Checking Fairness

In this section, we present an algorithm for model checking fair formulas. We start with those expressible using only $F$ and $G$ modalities, then we extend the algorithm to deal with general fairness. Finally, we handle all LTL formulas with fairness assumptions.

### A. Fairness in LTL($F$, $G$)

In this subsection we focus on a fragment of LTL formulas, denoted LTL($F$, $G$), which only contains $F$ and $G$ modalities, i.e., it is defined by the following grammar:

$$\varphi := a | \neg a | \varphi \land \varphi | \varphi \lor \varphi | F \varphi | G \varphi.$$  

For each fairness in LTL($F$, $G$), we shall show that it can be transformed into an equivalent formula where all propositional formulas are directly preceded by precisely two modalities, either $FG$ or $GF$, which we call the normal form of a fairness. Such a transformation is purely syntactical and enables us to avoid the automata construction totally. We call the transformation procedure the flatten operation, denoted by $\text{flt}$.

### Theorem 1. If $\varphi \in \text{LTL}(F, G)$ and $\varphi$ is a fairness, then $\varphi \equiv \text{flt}(\varphi) \equiv \bigvee_{i=1}^{m} (\text{FG} l_i \land (\bigwedge_{j=1}^{n} \text{GF} l_{ij}))$.

where $l_i$ and $l_{ij}$ are propositional formulas.

We below give the intuition behind the syntactic transformation. Detailed proof can be found in the appendix.

For trivial fair formula like $Fa \lor G \neg a$, it seems impossible to get the normal form. However, due to Lemma 1 we first add $\text{GF}$ in front of the original formula and then apply the flatten operation, which gives us the normal form $\text{GF} a \lor \text{FG} \neg a$.

Given a fairness $\varphi \in \text{LTL}(F, G)$, our goal is to obtain an equivalent formula of the normal form. To that end, we first make sure that there exists at least one $FG$ or $GF$ in front of every propositional formula, which is guaranteed by safely adding $\text{GF}$ in front of $\varphi$. After that, we are going to push every $FG$ and $GF$ directly in front of all propositional formulas. To achieve this, one needs to discuss the distributivity of $\text{GF}$ and $\text{FG}$ over $\lor$ and $\land$.

Suppose $\varphi_1, \varphi_2 \in \text{LTL}$, our goal is pushing $FG$ and $FG$ inside. We consider following four cases:

1. $\text{GFG} \equiv \text{FG} \text{GFF} \equiv \text{GF} \text{FGG} \equiv \text{FG} \text{GFG} \equiv \text{GF} \text{FGF} \equiv \text{GF} \text{FGF} \equiv \text{GFG}$ are trivial according to the semantics of LTL. This insures that we have only $GF$ and $FG$ modalities since we first add $\text{GF}$ in front of $\varphi$.

2. $\text{GF} (\varphi_1 \lor \varphi_2) \equiv \text{GF} \varphi_1 \lor \text{GF} \varphi_2$ and $\text{FG} (\varphi_1 \land \varphi_2) \equiv \text{FG} \varphi_1 \land \text{FG} \varphi_2$ hold since $\text{GF}$ and $\text{FG}$ are distributive over $\lor$ and $\land$ operator respectively.

3. $\text{GF} (\varphi_1 \land \varphi_2) \equiv \text{GF} \varphi_1 \land \text{GF} \varphi_2$, $\text{FG} (\varphi_1 \lor \varphi_2) \equiv \text{FG} \varphi_1 \lor \text{FG} \varphi_2$, $\text{GF} (\varphi_1 \land \varphi_2) \equiv \text{GF} \varphi_1 \land \text{GF} \varphi_2$ and $\text{FG} (\varphi_1 \lor \varphi_2) \equiv \text{FG} \varphi_1 \lor \text{FG} \varphi_2$. Intuitively, if the operands of $\text{GF}$ and $\text{FG}$ are not propositional formulas, they must be the four cases we listed here after we go through case 2 and case 3.

4. $\text{GF} (\varphi_1 \lor \varphi_2)$ and $\text{FG} (\varphi_1 \lor \varphi_2)$. This is the most challenging part since $\text{GF}$ ($\text{FG}$) is not distributive over $\land$ ($\lor$) operator. The following procedure relies on the structure of the formula. If the operand of $\text{GF}$ or $\text{FG}$ is propositional formula, then it is already the formula we desire; otherwise, if they are not case 3, we transform $\varphi_1 \land \varphi_2$ and $\varphi_1 \lor \varphi_2$ to disjunctive normal form (DNF) and conjunctive normal form (CNF) respectively. After that, we apply case 2 and may use case 3 for further processing.

Once we get a formula where all propositional formulas are adjacent to $\text{FG}$ or $\text{GF}$, we transform it into DNF, which gives us the normal form in Theorem 1. We illustrate the procedure of $\text{flt}$ operator via an example as follows:

**Example 1.** Let $\varphi = \text{FG} (a \lor (Fb \land Gc))$. We show how to flatten $\varphi$ step by step, where numbers above $= \text{denote the corresponding cases we listed above}$.

$$
\begin{align*}
\text{GF\varphi} & \quad \text{(1)} \quad a \lor (Fb \land Gc) \\
\text{FG}(a \lor Fb) \land (a \lor Gc) & \quad \text{(2)} \\
\text{FG}(a \lor Fb) \land (a \lor Gc) & \quad \text{(3)} \\
\text{FG}(a \lor Fb) \land (a \lor Gc) & \quad \text{(4)} \\
\text{FG}(a \lor Fb) \land (a \lor Gc) & \quad \text{(5)} \\
\text{FG}(a \lor Fb) \land (a \lor Gc) & \quad \text{(6)} \\
\text{FG}(a \lor Fb) \land (a \lor Gc) & \quad \text{(7)} \\
\text{FG}(a \lor Fb) \land (a \lor Gc) & \quad \text{(8)}
\end{align*}
$$
Intuitively, it means whenever $\pi \models \varphi$, it must be the case that $\pi$ ends up with a loop such that either all states on the loop satisfy $a$ or all states satisfy $c$ and at least one state satisfies $b$. This can be verified by applying the semantics of LTL.

By Corollary 1 we only need to consider the infinite suffixes of the paths that all states will be visited infinitely often. That is to say, we only need to consider the strongly connected components (SCCs) of $\mathcal{X}$ that can be reached. An SCC $B$ of $\mathcal{X}$ is a state set such that for any $s, t \in B$, there exists a path from $s$ to $t$.

**Definition 3 (Accepting SCC).** Given a formula $\varphi = \text{FG}l \land \bigwedge_{i=1}^{m} \text{GF}l_{i}$ and an SCC $B$. If 1) for every state $s \in B$, $s \models l$ and 2) for each $k$, there exists $s_k \in B$, such that $s_k \models l_k$, then we say SCC $B$ is accepting for $\varphi$.

With the definition of accepting SCC, we have the following theorem:

**Theorem 2.** For any $\varphi = \text{FG}l \land \bigwedge_{i=1}^{m} \text{GF}l_{i}$, there exists an infinite path $\pi$ in $\mathcal{X}$ such that $\pi \models \varphi$ iff there exists a reachable SCC $B$ such that $B$ is accepting for $\varphi$.

**Proof:**

$\Rightarrow$ Since $\mathcal{X}$ is finite, for any $\pi \in \text{Paths}^{\omega}(\mathcal{X})$, there exists a smallest index $j$, such that all states in $\pi|_{j}$ will be visited by infinite times. By Corollary 1 it suffices to show that $\pi \models \varphi$ iff $\pi|_{j} \models \varphi$ since one can check that $\varphi$ is a fairness. For convenience, let $\pi_1 = \pi|_{j}$. Let $B_1$ be the set of states on $\pi_1$. Obviously, all states in $B_1$ are connected since all states will be visited by infinite times. $\pi_1 \models \text{FG}l$ means $s \models l$ for each $s \in B_1$ and $\pi \models \text{GF}l_{k}$ means that there exists $s_k \in B_1$ such that $s_k \models l_k$ for each $k$. Let $B = B_1 \subseteq S$, then $B$ is an SCC and is accepting for $\varphi$.

$\Leftarrow$ This direction is trivial, since we can always construct a path $\pi_2$ that starts from any $s \in B$ and visits all states in $B$ by infinite times. Since $B$ is reachable, we can find a finite path $\pi_1$ which starts from the initial state and reaches the first state of $\pi_2$. Let $\pi = \pi_1 \cdot \pi_2$. Obviously $\pi \models \bigwedge_{i=1}^{m} \text{GF}l_{k} \land \text{FG}l_{i}$, thus we complete the proof.

For fair formula $\varphi \in \text{LTL}(F, G)$, $\mathcal{X} \models \varphi$ means that for all infinite paths $\pi$ starting from initial state $\delta$, $\pi \models \varphi$. Conversely, if $\neg(\mathcal{X} \models \varphi)$, then there exists an infinite path $\pi$ such that $\pi \models \neg \varphi$. By Theorem 1 $\neg \varphi$, obviously in $\text{LTL}(F, G)$ as well, is equivalent to the formula of normal form $\bigvee_{i=1}^{m} \varphi_i$, which means there exists an infinite path $\pi$ such that $\pi \models \varphi_i$. In other words, there exists an SCC $B$ being accepted for $\varphi_i$ according to Theorem 2.

Based on Theorem 2, Algorithm 1 describes the procedure to determine whether all paths in $\mathcal{X}$ satisfy a given fair formula $\varphi$ in $\text{LTL}(F, G)$. For this, the algorithm first syntactically transforms $\neg \varphi$ into an equivalent formula of the form $\bigvee_{i=1}^{m} (\bigwedge_{j=1}^{n_i} \text{GF}l_{i,j} \land \text{FG}l_{i})$. For each $1 \leq i \leq m$, we then try to find an accepting SCC $B'$ such that all states in $B'$ satisfy $l_i$ and at least one state in $B'$ satisfies $l_{i,j}$ for each $1 \leq j \leq n_i$. In case an accepting SCC is found, there exists a path in $\mathcal{X}$ violating $\varphi$, hence $\mathcal{X} \not\models \varphi$; otherwise we conclude that $\mathcal{X} \models \varphi$.

The soundness and completeness of Algorithm 1 is guaranteed by the following theorem.

**Theorem 3.** Algorithm 1 is sound and complete.

Let $|\mathcal{X}|$ denote the size of the given model, i.e., the total number of states and transitions. The complexity of Algorithm 1 is shown in the following theorem.

**Theorem 4.** Algorithm 1 runs in time $O(|\mathcal{X}| \times 2^{\|\varphi\|})$ and in space $O(|\mathcal{X}| + |\varphi| \times 2^{\|\varphi\|})$.

Due to case 4 in the proof of Theorem 1 and the transformation that gives a formula of DNF, the resulting formula length can be $O(2^{\|\varphi\|})$ in the worst case. Suppose $n_1$ is the number of literals first preceded by $F$, and $n_2$ for number of literals first preceded by $G$, obviously $n_1 + n_2 \in O(\|\varphi\|)$. We then have $2^{n_1}$ options for $\text{FG}l$ formulas and $2^{n_2}$ for $\text{GF}l_{k}$ since the number of $l_k$ is $n_2$, so we will at most have $2^{n_1+n_2}$ formulas have the form $\bigwedge_{i=1}^{m} \text{GF}l_{k} \land \text{FG}l_{i}$ and each formula of that form at most has $n_2 + 1$ literals, which means that formula length can be $|\varphi| \times 2^{\|\varphi\|}$ at worst case. That is, we will at most have $2^{n_1+n_2}$ formulas with the form $\bigwedge_{i=1}^{m} \text{GF}l_{k} \land \text{FG}l_{i}$, and the time for model checking $\bigwedge_{i=1}^{m} \text{GF}l_{k} \land \text{FG}l_{i}$ will be $|\mathcal{X}|$ to traverse all SCCs. Comparing to the classical algorithm presented in [32], Algorithm 1 has the same time complexity. However, experiment shows that our algorithm achieves much better performance comparing to the classical one. Furthermore, Algorithm 1 reduces the space complexity from $O(|\mathcal{X}| \times 2^{\|\varphi\|})$ to $O(|\mathcal{X}| + 2^{\|\varphi\|})$ for fairness in $\text{LTL}(F, G)$.

**B. Fairness in LTL**

In this subsection we deal with arbitrary fair formulas including those not expressible in $\text{LTL}(F, G)$. More notations are needed. Given $B \subseteq S$ and $s \in B$, let $\mathcal{X}_{s} := (B, s, T_B, L_B)$ where $T_B = T \cap (B \times B)$ and $L_B : B \rightarrow 2^{\Delta}$ such that $L_B(t) = L(t)$ for any $t \in B$. In other words, $\mathcal{X}_{s}$ is a sub-model of $\mathcal{X}$ where only states in $B$ and transitions between states in $B$ are kept.

We have discussed the model checking algorithms of fair formulas in $\text{LTL}(F, G)$. But for any $\varphi \in \text{LTL}$ which is a fairness, the model checking problem is more involved. Take

---

**Algorithm 1.**

The procedure $\text{fairMC}$ for checking whether $\mathcal{X} \models \varphi$, where $\varphi$ is a fair formula in $\text{LTL}(F, G)$. $\text{fairMC}(\varphi, \mathcal{X})$ returns True if $\mathcal{X} \models \varphi$, and False otherwise.

1: procedure $\text{fairMC}(\varphi, \mathcal{X})$
2: $\text{fl}(-\varphi) \equiv \bigvee_{i=1}^{m} \varphi_i = \bigwedge_{i=1}^{n_i} \text{GF}l_{i,j} \land \text{FG}l_{i}$;
3: for all $(1 \leq i \leq m)$ do
4: $B \leftarrow \{s \in S \mid s \models l_i\}$;
5: if $B \neq \emptyset$ then
6: for all $(\text{SCC} B' \subseteq B)$ do
7: if $(B' \text{ is accepting for } \varphi_i)$ then
8: return False;
9: return True;
\( \varphi = \mathrm{GF}(a \land X(b \lor c)) \) as an example, it is easy to see that \( \varphi \) is a fairness by Lemma \[1\] But it is obviously impossible to find an equivalent formula in LTL(F, G) to represent \( \varphi \) since the order of states in SCC matters. That is, we can not directly employ Algorithm \[1\] to handle fairness in LTL. In this subsection, we propose a solution to deal with the fairness in LTL.

Let LTL(U, X) denote the fragment of LTL only containing U and X modalities, namely, it is defined by the following grammar:

\[
\varphi ::= a \mid \neg a \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \mid X\varphi \mid \varphi_1 U \varphi_2.
\]

Formulas in LTL(U, X) are also known as co-safety in literature \[20, 3, 21\].

Similar as in Section \[III-A\] we shall show that any fair formula can be transformed into an equivalent one, where all U and X modalities can be separated from F and G such that the innermost formulas are all in LTL(U, X). Such a transformation is syntactical as well, after which a formula in DNF will be obtained and moreover, each sub-formula can be handled individually by specific and efficient algorithms.

**Theorem 5.** For any fair formula \( \varphi \) in LTL,

\[
\varphi \equiv \bigvee_{i=1}^{m} \left( \varphi_0 \land \mathrm{GF}\varphi_1 \land \left( \bigwedge_{j=2}^{n_i} \mathrm{GF}\varphi_j \right) \right),
\]

where \( \varphi_0 \in \text{LTL}(F, G) \) and \( \varphi_j \in \text{LTL}(U, X) \) for all \( 1 \leq i \leq m \) and \( 1 \leq j \leq n_i \).

As the model checking of an LTL formula \( \psi \) is essentially reduced to the problem of finding a path in \( \mathcal{K} \) violating \( \psi \), we shall focus on the procedure of finding a path in \( \mathcal{K} \) satisfying a given formula in the following. Let \( \varphi = \varphi_0 \land \mathrm{GF}\varphi_1 \land \mathrm{GF}\varphi_2 \land \cdots \land \mathrm{GF}\varphi_n \) with \( \varphi_0 \in \text{LTL}(F, G) \) and \( \varphi_j \in \text{LTL}(U, X) \) for all \( 1 \leq j \leq n \). We show how to optimize the procedure of finding a path satisfying \( \varphi \). Firstly, assume \( \varphi_0 \equiv \top \), hence the sub-formula \( \mathrm{GF}\varphi_1 \) can be omitted from \( \varphi \). The formal procedure for checking whether exists a path in \( \mathcal{K} \) satisfying \( \varphi \) is presented in Algorithm \[2\]. As \( \varphi \) is a fair formula, we can easily show that \( \varphi_0 \) must be also a fair formula. Since \( \varphi_0 \in \text{LTL}(F, G) \), a simple modification of Algorithm \[1\] can be applied to find all accepting SCCs with respect to \( \varphi_0 \) in \( \mathcal{K} \) (line 2). If no accepting SCC exists, we can terminate, as no path in \( \mathcal{K} \) can satisfy \( \varphi \); Otherwise, for each accepting SCC \( B \) and \( \varphi_j \) with \( 2 \leq j \leq n \), add a fresh atomic proposition \( a_j \) to \( A \) (line 7) iff there exists a state \( t \in B \) and \( \pi \in \text{Paths}^\omega(\mathcal{K}_B) \) such that \( \pi \models \varphi_j \) (line 6). This step can be done by launching classical algorithms: A path \( \pi \in \text{Paths}^\omega(\mathcal{K}_B) \) exists such that \( \pi \models \varphi_j \) iff \( \mathcal{K}_B \) does not satisfy \( \neg \varphi_j \). Finally, an SCC \( B \) is accepted by \( \varphi \) if at least one state in \( B \) is marked by \( a_j \) for each \( 2 \leq j \leq n \), namely, \( A = \{a_j\}_{2 \leq j \leq n} \) (line 9).

The key point behind Algorithm \[2\] is that \( \varphi_j \) \( (2 \leq j \leq n) \) is in LTL(U, X), the corresponding Büchi automaton of which is terminal \[4\]. Therefore, once a path \( \pi \) satisfies \( \varphi_j \), we can always find a finite fragment of \( \pi \) which suffices to conclude that \( \pi \models \varphi_j \) regardless of the remainder of \( \pi \). In other words, whenever \( \pi \models \varphi_j \), there exists \( i \geq 0 \) such that \( (\pi_1^i \mid \pi') \models \varphi_j \) for any infinite path \( \pi' \). Whenever Algorithm \[2\] returns \( \text{True} \) and finds an accepting \( B \) for \( \varphi \), we can construct a path satisfying \( \varphi \) as follows:

1) Let \( \pi_1 \) be a finite path in \( \mathcal{K}_B \) for any \( t \) such that all states in \( B \) appear in \( \pi_1 \) for at least once. Traversing all states in \( B \) is useful to witness \( \varphi_0 \in \text{LTL}(F, G) \).

2) Continue from the last state of \( \pi_1 \) and go to a state \( t_2 \) by following any path, where \( t_2 \) is a state in \( B \), from which a path satisfying \( \varphi_j \) exists. Let \( \pi_2' \) be the resultant path ending at \( t_2 \). Expand \( \pi_2' \) by following the path satisfying \( \varphi_j \) and stop whenever \( \varphi_j \) is for sure satisfied. Denote the resultant finite path by \( \pi_2 \).

3) Keep extending \( \pi_2 \) by repeating step 2) for each \( 3 \leq j \leq n \). Let \( \pi_n \) denote the resulting path.

4) Let \( \pi_n' \) denote an arbitrary extension of \( \pi_n \) such that \( t \) is a direct successor of the last state of \( \pi_n' \), namely, \( (\pi_n')^\omega \) is a cyclic path in \( \mathcal{K}_B \).

By construction, it is easy to check that \( (\pi_n')^\omega \models \varphi \), which also shows the soundness and correctness of Algorithm \[2\].

Secondly, when \( \varphi \neq \top \), we have to make sure that an accepting path also satisfies \( \mathrm{GF}\varphi_1 \). For this purpose, we first transform \( \mathrm{GF}\varphi_1 \) to a Büchi automaton, denoted \( A_1 \), and then build a product model \( \mathcal{K} \times \mathcal{A}_1 \) as in the classical algorithm. Let \( a_1 \) be a fresh atomic proposition such that \( a_1 \) holds at a state iff the state is accepting in \( \mathcal{K} \times \mathcal{A}_1 \). The remainder of the procedure is similar as the case when \( \varphi \equiv \top \) except \( \mathrm{GF}\varphi_1 \) is replaced by \( \mathrm{GF}a_1 \) in \( \varphi \) and the model under checked will be \( \mathcal{K} \times \mathcal{A}_1 \).

**c) Discussions:** As mentioned before, formulas in LTL(U, X) are guarantee properties according to the classification in \[6\]. Their corresponding Büchi automata are terminal, for which specific and efficient algorithms exist \[41\]. By separating a fair formula, we can identify sub-formulas belonging to different fragments, each of which will be handled by specific and efficient algorithms.
C. General Formulas with Fairness Assumptions

In this subsection we show how the model checking problem for general LTL formulas with fairness assumptions can be accelerated by the specific algorithms for fair formulas introduced in the above subsections.

Given a fair formula $\varphi_f$ and an LTL formula $\varphi$, the model checking problem of $\varphi$ under the assumption $\varphi_f$ reduces to checking whether $\mathcal{K} \models (\varphi_f \implies \varphi)$. In order to make use of our specific algorithm for fairness, the procedure can be divided into two steps:

1) $\neg \varphi$ is first transformed into a Büchi automaton, denoted $\mathcal{A}_{\neg \varphi}$, and the product of $\mathcal{A}_{\neg \varphi}$ and $\mathcal{K}$ is then constructed, where all accepting states are marked by a fresh atomic proposition $accepting$;

2) Then $\mathcal{K} \models (\varphi_f \implies \varphi)$ iff there is no path in the product satisfying $\varphi_f \land \mathcal{G} accepting$. Note $\varphi_f \land \mathcal{G} accepting$ is still a fair formula, for which our efficient algorithm can be applied.

Note that we can specify some fairness assumption like $\mathcal{G}F(a \land b \land c)$ which is not in $\mathcal{LTL}(F,G)$. Moreover, by making use of our algorithm for fairness, we expect some speed up in the model checking procedure.

IV. FORMULA CHARACTERIZATION

In this section, we specify some formula sets which are favourable to our algorithm as well as some formula sets for which our syntactic transformation leads to dramatic blow up of the formula lengths.

We first characterize some formula sets to which applying our transformation does not lead to dramatic growth of formula length, and we call them the fast $\mathcal{LTL}$ formulas.

Definition 4. Let $\Sigma_f$ be a subset of $\mathcal{LTL}$ formulas which is constructed by following rules. Then $\varphi_f, \varphi_e \in \Sigma_f$ where $\varphi_1 \in \mathcal{LTL(U,X)}$.

\[
\begin{align*}
\varphi_0 & ::= \varphi_1 \mid \mathcal{F}\varphi_0 \mid \mathcal{G}\varphi_0 \mid \varphi_0 \land \varphi_0 \\
\varphi_f & ::= \varphi_0 \mid \varphi_1 \lor \varphi_f \\
\varphi_e & ::= \varphi_1 \mid \varphi_e \lor \varphi_e \mid \varphi_e \land \varphi_e \mid \mathcal{F}\varphi_e \mid \mathcal{G}\varphi_e
\end{align*}
\]

By induction on the structure of formulas defined in Definition 4 and similar analysis from Theorem 5 it is straightforward to show that:

Corollary 2. Let $\varphi_f(\varphi_e)$ be a formula defined in Definition 4 and $\varphi'_f(\varphi'_e)$ be the resulting formula after the transformation defined in Theorem 5. Then $|\varphi'_f| = O(|\varphi_f|)$. Similarly, we have $|\varphi'_e| = O(2^{2|\varphi_e|})$.

In the following, we give the intuition why the transformation increase the formula length by the following example.

Example 2. Let

$\varphi = \psi_1 \cup \psi_2 = ((\mathcal{G}F a_1 \land \mathcal{G}F a_2) \lor \cdots \lor (\mathcal{G}F a_{p-1} \land \mathcal{G}F a_p))$

$\cup ((\mathcal{G}F b_1 \lor \mathcal{G}F b_2) \land \cdots \land (\mathcal{G}F b_{q-1} \lor \mathcal{G}F b_q))$

Clearly, $|\varphi| = O(p + q)$. We need first get all $\mathcal{F}$ and $\mathcal{G}$ modalities out of the scope of $\mathcal{U}$. To this end, by rules of $(\varphi_1 \lor \varphi_2) \cup \varphi_3 \equiv \varphi_1 \cup \varphi_3 \land \varphi_2 \cup \varphi_3$ and $\varphi_1 \cup (\varphi_2 \lor \varphi_3) \equiv \varphi_1 \cup \varphi_2 \lor \varphi_1 \cup \varphi_3$, it requires us to transform $\psi_1$ to CNF form and $\psi_2$ to DNF form. After that, we get a formula which is of size $O(2^{|\varphi|})$.

We remark that our transformation does not work when the formula contains $\mathcal{W}$ modalities, so we replace $\mathcal{W}$ with $\mathcal{G}$ modalities. As a result, it may increase the number of modalities after negating a formula. Take $\varphi = \mathcal{G}F \neg \varphi \lor \neg \mathcal{G}F U c$ for example, after negating $\varphi$, it gives us $\mathcal{G}F(a \land (\neg \mathcal{G}F \neg \varphi \lor \neg \mathcal{G}F U c))$, which is equivalent to $\mathcal{G}F(a \land \mathcal{G}F (\neg \mathcal{G}F \neg \varphi \lor \neg \mathcal{G}F U c))$. After applying the formula transformation, the resulting formula becomes $\mathcal{G}F (\mathcal{G}F \neg \varphi \lor \mathcal{G}F a) \lor \neg \mathcal{G}F U c)$. We notice that the reduction for $\mathcal{W}$ modality contributes to the growth of the formula length.

V. EXPERIMENT

In this section we first illustrate briefly how our algorithm is implemented symbolically in NuSMV and then compare the experiment results with existing algorithms. NuSMV is a Symbolic Model Verifier extending the first BDD-based model checker SMV [5]. Compared to tools based on explicit representations, NuSMV is able to handle relatively more complex formulas [27], which is the main reason for choosing NuSMV in our experiment.

A. A Symbolic Implementation

We implement our algorithm in NuSMV symbolically. The algorithm first decomposes a given formula syntactically to the specific form according to Theorems 1 and 5 and then uses the fair cycle detection algorithm proposed by Emerson and Lei [12] to find accepting SCCs. For instance, given a fair formula $\varphi \in \mathcal{LTL}(F,G)$ such that $\varphi \equiv \forall i \exists c \nu_{i-1} \left(\mathcal{F}G i \land (\bigwedge_{j=1}^{n_i} \mathcal{G}F h_{i,j})\right)$, the fair cycle detection algorithm can be applied to determine whether there exists an SCC in $\mathcal{K}$ satisfying $\mathcal{F}G i \land (\bigwedge_{j=1}^{n_i} \mathcal{G}F h_{i,j})$ for some $1 \leq i \leq m$. By doing so, we avoid to enumerating all SCCs one by one. We refer interested readers to [12] for details about the fair cycle detection algorithm.

B. Experiment Results

We adopt two well-known and scalable problems as our benchmarks: dining philosopher problem (PD) and binary semaphore protocol (BS). Their sizes are summarized in Table 8 where “Size” refers to the number of reachable states for each model, PDx denotes the PD model with $x$ philosophers, and similarly for BSx. All experiment results were obtained on a computer with an Intel(R) Core(TM) i7-2600 3.4GHz CPU running Ubuntu 14.04 LTS. We set time and memory limits to be 2 hours and 3 GB, respectively.

We consider three categories of formulas: The first category takes formulas often used in verification tasks. Specifically, for PD model we consider the following formula, saying that the first philosopher will eat eventually if no one will be
TABLE I
NUMBER OF REACHABLE STATES

| Model | PD6 | PD9 | PD12 | BS4 | BS8 | BS12 | BS16 |
|-------|-----|-----|------|-----|-----|------|------|
| Size  | 566 | 13605 | 524782 | 80 | 2304 | 53248 | 1114110 |

TABLE II
TIME (SECOND) AND MEMORY USAGE (MB) FOR FORMULAS IN THE FIRST CATEGORY

| Formula | Model | Time (second) | Memory (MB) |
|---------|-------|---------------|-------------|
| Spec₁  | PD6  | 2.65          | 19.49       |
| Spec₂  | BS4  | 0.30          | 0.15        |
| Spec₃  | BS8  | 0.04          | 172.63      |
| Spec₁  | BS12 | 0.11          | 33.85       |
| Spec₂  | BS16 | 1.06          | 139.58      |
| Spec₃  | BS4  | 0.02          | 12.22       |
| Spec₁  | BS8  | 0.03          | 101.25      |
| Spec₂  | BS12 | 0.12          | 36.31       |
| Spec₃  | BS16 | 1.14          | 337.55      |

TABLE III
FORMULAS IN SCALABLE PATTERNS GENERATED BY “GENLTL”.

| Pattern | genltl arguments | Formula |
|---------|------------------|---------|
| p₁      | ¬and-lg = n     | \(\bigwedge_{i=1}^{n} GFready_i \Rightarrow GFeat_i\) |
| p₂      | ¬and-gf = n     | \(\bigwedge_{i=1}^{n} GFG_i\) |
| p₃      | ¬¬p + x = n − 1 | \(\bigwedge_{i=1}^{n} (GF_i \lor FG_{i+1})\) |
| p₄      | ¬¬p ∧ ¬¬(a₁)    | \(\bigwedge_{i=1}^{n} (GF_i \lor FG_{i+1})\) |

formulas generated by “genltl” – a tool of Spot library [11] to generate formulas of scalable patterns; more details can be found at https://spot.lrde.epita.fr/genltl.html. These patterns and sample formulas are presented in Table III, where column “genltl arguments” denotes arguments used by “genltl” to generate corresponding formulas and \(n\) the number of philosophers in PD or the number of processes in BS. In Table III and the following formulas, we use \(a_1, b_1, \ldots\) as placeholders which will be replaced by proper atomic propositions during the experiment. To ease the presentation, we omit the details here.

starved (fairness assumption), namely, whenever a philosopher is ready, he/she will be able to eat eventually:

\[ Spec₁ = \left( \bigwedge_{i=1}^{n} (GFready_i \Rightarrow GFeat_i) \right) \Rightarrow GFeat₁ \]

For BS model, we consider the following two formulas:

\[ Spec₂ = \left( \bigwedge_{i=1}^{n} (GFenter_i \Rightarrow GFcritical_i) \right) \Rightarrow GFcritical₁ \]

\[ Spec₃ = \left( \bigwedge_{i=1}^{n} (GFenter_i \Rightarrow GFcritical_i) \right) \Rightarrow ((¬GFcritical₁ \land ¬GFcritical₂) \lor GFcritical₂) \]

\(Spec₂\) denotes a similar specification as \(Spec₁\), while \(Spec₃\) requires that the second process entering the critical part before the first and third processes. Notice that all given fairness assumptions are simple formulas in LTL(F, G).

In the following we write NuSMV to represent the automata-theoretic approach implemented in NuSMV. Table III shows both the time and memory spent by our algorithm and NuSMV to check formulas in the first category on PD and BS models, where T-O and M-O denote “timeout” and “out-of-memory”, respectively. From Table III we can see that our algorithm outperforms NuSMV in almost all cases. In particular, our algorithm terminates in seconds for some cases, while NuSMV runs out of time or memory.

The above formulas \(Spec_i\) (\(i = 1, 2, 3\)) are relatively simple. Therefore, we further consider the second category of fair formulas which can be converted to simple Streett/Rabin fairness conditions and alternative algorithms in [2] can be applied. We expect some speedup if those algorithms are implemented in NuSMV. Our algorithm for fairness in LTL(F, G) follows...
the same idea except that we first conduct a formula transformation so that we can handle fairness like $GF(a \land Gb)$.

Note all formulas in Table III are actually a subset of $LTL(F, G)$. This motivates us to consider the third category of formulas, which are summarized in Table IV. These formulas are often adopted to evaluate performance of an LTL model checker or planner in the literature; see for instance [31], [14], [23], [13]. The time consumption for checking these formulas is presented in Table V and VII while the memory consumption is shown in Table VI and VIII. From these results we observe similar phenomena as before for most cases except for “p11”, “p12”, and “p16”, where our algorithm uses more time and/or memory than NuSMV for certain cases, particularly when “p11” and PD models are concerned. We explain such differences in details in the following.

d) Discussions: As mentioned before, our algorithm relies on syntactical transformations in Theorems 1 and 5. These transformations can decompose a fair formula into smaller sub-formulas, whose corresponding Büchi automata are usually much smaller than the automaton of the original formula. This is the main reason that our algorithm achieves much better performance than the classical algorithm for most of the instances. However, the syntactic transformations adopted in Theorem 1 and 5 may cause exponential blow-up for certain cases; for instance formulas whose negations are in form of “p11” and “p12”. In order to lift all F and G modalities outside scopes of U modality, our transformation may need to transform back and forth between CNF and DNF of some formulas, especially for those where F, G, and U are alternatively nested for many times. Therefore, for such formulas, the syntactic transformation will be time-consuming and often result in formulas of exponentially longer than the original ones.

It is easy to check that many formulas considered in this paper are characterized by Definition 4. Transforming the negation of these formulas only leads to a linear increase in the formula length. The exceptions are “p6”, “p9”, “p11”, “p12”, and “p14–p17”. It is worthwhile to mention that even though for formulas such that the transformations result in formulas of exponential length, our algorithm is not necessarily slower than NuSMV, as the corresponding Büchi automata may be exponentially large as well; for instance “p8” and “p11”.

VI. Conclusion and Future Work

We presented a novel model checking algorithm for formulas in LTL with fairness assumptions. Our algorithm does not follow the automata-theoretic approach completely but tries to decompose a fair formula into several sub-formulas, each of which can be handled by specific and efficient algorithms. We showed by experiment that our algorithm in many cases exceeds NuSMV up to several orders of magnitudes.

As future work we would like to investigate whether or not our syntactic transformations for fair formulas can be extended to deal with more LTL formulas. Another interesting direction is to extend our approach to probabilistic systems including discrete-time Markov chains [9] and Markov decision processes [8]. In the future, we would also like to combine our method with on-the-fly Büchi construction.

REFERENCES

[1] B. Alpern and F. B. Schneider. Recognizing safety and liveness. Distributed Computing, 2(3):117–126, 1987.
[2] C. Baier and J.-P. Katoen. Principles of Model Checking. The MIT Press, 2008.
[3] A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Sampling-based motion planning with temporal goals. In ICRA, pages 2689–2696, 2010.
[4] R. Bloem, K. Ravi, and F. Somenzi. Efficient decision procedures for model checking of linear time logic properties. In CAV, volume 1633 of LNCS, pages 222–235. Springer, 1999.
[5] L. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: $10^{10}$ states and beyond. Inf. Comput., 98(2):142–170, 1992.
[6] I. Cerná and R. Pelánek. Relating hierarchy of temporal properties to model checking. In MFCS, volume 2747 of LNCS, pages 318–327. Springer, 2003.
[7] A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model checker. STTT, 2(4):410–425, 2000.
[8] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events (extended abstract). In ICALP, pages 336–349, 1990.
[9] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. J. ACM, 42(4):857–907, 1995.
[10] M. Daniele, F. Giunchiglia, and M. Y. Vardi. Improved automata generation for linear temporal logic. In CAV, volume 1633 of LNCS, pages 249–262, Springer, 1999.
[11] A. Duret-Lutz and D. Poitrenaud. SPOT: An extensible model checking library using transition-based generalized Büchi automata. In Proceedings of 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS ’04, pages 76–83. IEEE Press, 2004.
[12] E. A. Emerson and C. Lei. Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In Symposium on Logic in Computer Science, pages 267–278. IEEE Computer Society, 1986.
[13] J. Esparza and J. Kretinsky. From LTL to deterministic automata: A safraless compositional approach. In CAV, volume 8559 of LNCS, pages 192–208. Springer, 2014.
[14] K. Etessami and G. J. Holzmann. Optimizing Büchi automata. In CONCUR, volume 1877 of LNCS, pages 153–167. Springer, 2000.
[15] N. Francez. Fairness. Springer, 1986.
[16] P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In CAV, volume 2102 of LNCS, pages 53–65. Springer, 2001.
[17] M. Hammer, A. Knapp, and S. Merz. Truly on-the-fly LTL model checking. In TACAS, volume 3440 of LNCS, pages 191–205. Springer, 2005.
[18] G. J. Holzmann. The model checker SPIN. IEEE Trans. Softw. Eng., 24(5):279–295, 1997.
[19] D. Kini and M. Viswanathan. Limit Deterministic and Probabilistic Automata for LTL/GU. In TACAS, LNCS, pages 628–642, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
[20] O. Kupferman and M. Y. Vardi. Model checking of safety properties. Formal Methods in System Design, 19(3):291–314, 2001.
[21] B. Lacerda, D. Parker, and N. Hawes. Optimal and dynamic planning with temporal goals. In ICRA, pages 2689–2696, 2010.
[22] L. Lamport. Proving the correctness of multiprocess programs. IEEE Trans. Software Eng., pages 125–143, 1977.
[23] R. Pelánek. BEEM: benchmarks for explicit model checkers. In SPIN, volume 4595 of LNCS, pages 263–267. Springer, 2007.
A. Pnueli. The temporal logic of programs. In J. Queille and J. Sifakis. Fairness and related properties in transition systems - A temporal logic to deal with fairness. Acta Inf., 19:195–220, 1983.

J. Queille and J. Sifakis. Fairness and related properties in transition systems - A temporal logic to deal with fairness. Acta Inf., 19:195–220, 1983.

J. Queille and J. Sifakis. Fairness and related properties in transition systems - A temporal logic to deal with fairness. Acta Inf., 19:195–220, 1983.

E. Renault, A. Duret-Lutz, F. Kordon, and D. Poitrenaud. Strength-based decomposition of the property Büchi automaton for faster model checking. In TACAS, volume 7795 of LNCS, pages 580–593. Springer, 2013.

K. Y. Rozier and M. Y. Vardi. LTL satisfiability checking. In Proceedings of the 14th international SPIN conference on Model checking software, volume 4595 of LNCS, pages 149–167. Springer-Verlag, 2007.

S. Sickert, J. Esparza, S. Jaax, and J. Kretinsky. Limit-Deterministic Büchi Automata for Linear Temporal Logic. In CAV, LNCS, 2016. To appear.
APPENDIX

To present the transformation formally, we need to define another semantic equivalence between LTL formulas with restricted to cyclic sequences as follows:

Definition 5. Given two formulas \( \phi, \psi \), we write \( \phi \equiv_c \psi \) iff for any \( \sigma \in (2^A)^* \), \( \sigma^0 \models \phi \) iff \( \sigma^0 \models \psi \).

We call equivalence relation \( \equiv_c \) cyclic equivalence since we only care about the cyclic words. It is easy to show that \( \equiv \) and \( \equiv_c \) make no difference for fairness according to Corollary I. However, \( \equiv_c \) is still necessary, as some rules adopted by the transformation and presented in the following theorems only preserve \( \equiv_c \), but not \( \equiv \). For instance, \( \text{FG} \phi \equiv_c \text{G} \phi \) holds while \( \text{FG} \phi \equiv \text{G} \phi \) does not in general.

A. Proof of equations to explain Theorem 7

We first prove the proof of the equations listed to explain the intuition behind Theorem 1. In the following, we only prove the first equation, the second equation immediately follows by negating both sides of the first equation.

1) \( \text{GF}(\phi_1 \lor \phi_2) \equiv \text{GF}(\phi_1) \lor \text{GF}(\phi_2) \) and \( \text{FG}(\phi_1 \land \phi_2) \equiv \text{FG}(\phi_1) \land \text{FG}(\phi_2) \). The direction from \( \text{GF}(\phi_1) \lor \text{GF}(\phi_2) \) to \( \text{GF}(\phi_1 \lor \phi_2) \) is trivial. Consider the other direction, for any infinite word \( \sigma \) such that \( \sigma \models \text{GF}(\phi_1 \lor \phi_2) \), there exists infinite \( i \geq 0 \) such that \( \sigma^i \models \phi_1 \lor \phi_2 \), which implies that at least one formula out of \( \phi_1 \) and \( \phi_2 \) will be satisfied infinitely often.

2) \( \text{GF}(\phi_1 \land \phi_2) \equiv \text{GF}(\phi_1) \land \text{GF}(\phi_2) \) and \( \text{FG}(\phi_1 \lor \phi_2) \equiv \text{FG}(\phi_1) \lor \text{FG}(\phi_2) \). Obviously, \( \sigma \models \text{GF}(\phi_1 \land \phi_2) \) implies \( \sigma \models \text{GF}(\phi_1) \land \text{GF}(\phi_2) \) for any infinite word \( \sigma \). For the other direction, \( \sigma \models \text{GF}(\phi_1 \lor \phi_2) \) implies that there exist infinitely many \( i \) with \( i \geq 0 \) such that \( \sigma^i \models \phi_1 \lor \phi_2 \). For every such \( i \) from above, since \( \sigma \models \text{GF}(\phi_2) \), we have \( \sigma^i \models \phi_1 \land \text{GF}(\phi_2) \), hence \( \sigma \models \text{GF}(\phi_1 \lor \phi_2) \).

3) \( \text{GF}(\phi_1 \land \phi_2) \equiv \text{GF}(\phi_1) \land \text{GF}(\phi_2) \) and \( \text{FG}(\phi_1 \lor \phi_2) \equiv \text{FG}(\phi_1) \lor \text{FG}(\phi_2) \). The direction from \( \text{GF}(\phi_1 \land \phi_2) \) implies \( \text{GF}(\phi_1) \land \text{GF}(\phi_2) \) is straightforward. Now let \( \sigma \models \text{GF}(\phi_1) \land \text{GF}(\phi_2) \). Obviously, \( \sigma \models \text{GF}(\phi_2) \), which indicates that there exists some \( j \geq 0 \) such that for every \( i \geq j \), we have \( \sigma^i \models \phi_2 \). In addition, since \( \sigma \models \text{GF}(\phi_1 \lor \phi_2) \), we can find infinitely many \( k \geq j \) such that \( \sigma^k \models \phi_1 \lor \phi_2 \). Therefore, \( \sigma \models \text{FG}(\phi_1 \land \phi_2) \).

B. Proof of Theorem 7

Proof:

In the normal form, there are two kinds of modalities, namely \( \text{FG} \) and \( \text{GF} \). When we consider fairness, equivalence relation \( \equiv \) and \( \equiv_c \) coincide, we therefore use \( \text{G} \) and \( \text{F} \) to represent \( \text{FG} \) and \( \text{GF} \) respectively in the following flatten operation. We give the rules for flatten operation which we use to transfer any formula in LTL(\( F, G \)) to a formula of the normal form. We then prove that all transformation rules are sound. Formally, we define the flatten operator \( \text{flt} \) inductively as follows:

1) \( \text{flt}(l) = l \),
2) \( \text{flt}(\phi_1 \lor \phi_2) = \text{flt}(\phi_1) \lor \text{flt}(\phi_2) \),
3) \( \text{flt}(\phi_1 \land \phi_2) = \text{flt}_d \text{flt}(\phi_1) \land \text{flt}(\phi_2) \),
4) \( \text{flt}(\text{G} \phi) = \text{flt}_i (\text{flt}(\phi)) \),
5) \( \text{flt}(\text{F} \phi) = \text{flt}_d (\text{flt}_i (\text{flt}(\phi))) \),

where \( \text{flt}_d \) and \( \text{flt}_i \) denote transformations to equivalent formulas in conjunctive normal form (DNF) and conjunctive normal form (CNF), respectively, and

\[
\begin{align*}
\text{flt}_i(l) &= \text{Fl}_i\text{flt}(F_1) = F_1\text{flt}(G_1) = G_1, \\
\text{flt}_i(l) &= G_1;\text{flt}(F_1) = F_1;\text{flt}(G_1) = G_1, \\
\text{flt}(\phi_1 \lor \phi_2) &= \text{flt}(\phi_1) \lor \text{flt}(\phi_2) \quad \forall \phi \in \{ \land, \lor \},
\end{align*}
\]

Note that if \( \phi_1 \lor \phi_2 \) is a propositional formula, we consider it as one formula so that we do not apply \( \text{flt}_d \) or \( \text{flt}_i \) to \( \phi_1 \) and \( \phi_2 \) individually. As a result, we obtain a formula in form of \( \bigvee_{i=1}^{m} (G_i \land \bigwedge_{j=1}^{n_i} F_{i,j}) \). Since \( \equiv_c \) and \( \equiv \) coincide for fairness, we actually get an equivalent formula in the form of \( \bigvee_{i=1}^{m} (G_i \land \bigwedge_{j=1}^{n_i} F_{i,j}) \). We notice that rule 5 is more involved than other rules. Intuitively, after applying \( \text{flt} \) operator to \( \phi \), it gives us a formula, say \( \phi' \), which is in DNF. Further, since \( G \) is not distributive over \( \lor \) operator, we have to use \( \text{flt}_i \) to transform \( \phi' \) to a formula in CNF, say \( \phi'' \). By applying operator \( \text{flt}_d \) to \( \phi'' \), we are able to push \( G \) inside and then get a formula in DNF through \( \text{flt}_d \).

Next, we shall prove that rules 1 to 5 are sound. For this, it suffices to prove the following rules, where \( \phi_1, \phi_2 \), and \( \phi \) are arbitrary LTL formulas.

1) \( \text{FG} \equiv \text{G} \phi \), \( \text{GF} \equiv \text{F} \phi \). Trivial. That \( \text{G} \phi \) implies \( \text{FG} \phi \).

We show that whenever \( \sigma^0 \models \text{FG} \phi \), it is also the case that \( \sigma^0 \models \text{G} \phi \). This is also straightforward, as \( \sigma^0 \models \text{FG} \phi \) indicates any suffix of \( \sigma^0 \) satisfies \( \phi \).

2) \( \text{F}(\phi_1 \lor \phi_2) \equiv \text{F}(\phi_1) \lor \text{F}(\phi_2) \), \( G(\phi_1 \lor \phi_2) \equiv \text{G}(\phi_1) \lor \text{G}(\phi_2) \). Above equations can be proved by the equations in subsection A and together with \( \text{FG} \equiv \text{G} \) and \( \text{GF} \equiv \text{F} \).
3) \( F(\varphi_1 \land G\varphi_2) \equiv_c F\varphi_1 \land FG\varphi_2, \ G(\varphi_1 \lor F\varphi_2) \equiv_c G\varphi_1 \lor GF\varphi_2. \) Those equations can be proved by the equations in subsection \( [\mathcal{X}] \) and together with \( FG \equiv_c G \) and \( GF \equiv_c F. \n\)

4) \( FG\varphi_1 \land FG\varphi_2 \equiv FG(\varphi_1 \land \varphi_2). \) It has been proved before. This completes the proof.

Intuitively, the flatten operator \( flt \) takes a fair formula \( \varphi \in \text{LTL}(F, G) \) as an input and outputs a formula in DNF, where each sub-formula is a conjunction of formulas in form of \( I, Fl \) or \( Gl. \) We illustrate the definition of \( flt \) operator via an example as follows:

Example 3. Let \( \varphi = FG(a \lor Fb). \) We show how to flatten \( \varphi \) step by step, where numbers above = denote the corresponding rules in the above proof.

\[
\begin{align*}
\text{flt}(\text{Flt}(Fb)) & \Rightarrow \text{Flt}(a \lor Fb) & \Rightarrow \text{Flt}(a \lor Fb) \\
\text{flt}(\varphi) & \Rightarrow \text{Flt}(a \lor Fb) & \Rightarrow \varphi \lor Flt(Fa) \\
\text{flt}(FG(a \lor Fb)) & \Rightarrow \varphi \lor Flt(Fa) & \Rightarrow \varphi \lor Flt(Fa) \\
\text{flt}(FG(a \lor Fb)) & \Rightarrow \varphi \lor Flt(Fa) & \Rightarrow \varphi \lor Flt(Fa)
\end{align*}
\]

Intuitively, it means whenever \( \pi \models \varphi, \) it must be the case that \( \pi \) ends up with a loop such that either all states on the loop satisfy \( a \) or at least one state satisfies \( b. \) This can be verified by applying the semantics of \( \text{LTL}. \)

C. Proof of Theorem 3

Proof: Let \( \varphi_i \) with \( 1 \leq i \leq 4 \) be any \( \text{LTL} \) formula. We have the following equivalence relations with \( * \in \{ \land, \lor \}. \n\)

\[
\begin{align*}
FF\varphi & \equiv_c F\varphi \\
F(\varphi_1 \land \varphi_2) & \equiv_c F\varphi_1 \land F\varphi_2 \\
X(\varphi) & \equiv_c X\varphi \\
F(\varphi_1 \lor \varphi_2) & \equiv_c F\varphi_1 \lor F\varphi_2 \\
G(\varphi_1 \lor \varphi_2) & \equiv_c G\varphi_1 \lor G\varphi_2 \\
G(\varphi_1 \lor \varphi_2) & \equiv_c G\varphi_1 \lor G\varphi_2 \\
(\varphi_1 \lor \varphi_2) U(\varphi_3) & \equiv_c (\varphi_1 U\varphi_3) \lor (\varphi_2 U\varphi_3) \\
(\varphi_1 \lor \varphi_2) U(\varphi_3) & \equiv_c (\varphi_1 U\varphi_3) \lor (\varphi_2 U\varphi_3) \\
\varphi_1 U(\varphi_2 \lor \varphi_3) & \equiv_c (\varphi_1 U\varphi_2) \lor (\varphi_1 U\varphi_3)
\end{align*}
\]

Since \( F(\varphi_1 \land \varphi_2) \equiv_c F\varphi_1 \land F\varphi_2, \) and \( G(\varphi_1 \land \varphi_2) \equiv_c G\varphi_1 \land G\varphi_2, \) together with the distributive laws of \( \land \) and \( \lor, \) we can complete the proof. We only show the proofs of the following cases and omit others which are either similar or trivial.

- \( F(\varphi_1 U\varphi_2) \equiv_c F\varphi_2, \) \( \sigma \) is any infinite word.
- \( \sigma \in (2AP)^*. \)

\[
\begin{align*}
F(\varphi_1 U\varphi_2) & \equiv_c F\varphi_2, \ \text{if there exists } j \geq 0 \text{ such that } j \geq 0 \\
\sigma \models \varphi_1 U\varphi_2 & \Rightarrow \exists j \geq 0 \text{ such that } \sigma \models \varphi_1 U\varphi_2 \Rightarrow \exists j \geq 0 \text{ such that } \sigma \models \varphi_1 U\varphi_2.
\end{align*}
\]
Let $\varphi = \varphi_1 U (\varphi_2 \land G \varphi_3)$, $\psi = (\varphi_1 U \varphi_2) \land G \varphi_3$ and $\sigma \in (2^{AP^*})^c$.

\[ \Rightarrow: \sigma^o \models \varphi, \text{ then } \sigma^o \models \varphi_1 U \varphi_2. \]
Moreover, $\exists j \geq 0$ such that $\sigma^o|_j \models G \varphi_3$, which implies $\sigma^o \models G \varphi_3$. Therefore, $\sigma^o \models \psi$.

\[ \Leftrightarrow: \sigma^o \models (\varphi_1 U \varphi_2) \land G \varphi_3, \text{ then } \exists j \geq 0 \text{ such that } \sigma^o|_j \models \varphi_2 \text{ and } \forall 0 \leq i < j, \sigma^o|_i \models \varphi_1. \]
Since $\sigma^o \models G \varphi_3$ which implies $\sigma^o|_j \models G \varphi_3$. Thus $\sigma^o \models \varphi_1 U (\varphi_2 \land G \varphi_3)$.

- $\varphi_1 U (\varphi_2 \lor G \varphi_3) \equiv \varphi_1 U (\varphi_2 \lor (G \varphi_3))$.

Since $\varphi_1 U (\varphi_2 \lor G \varphi_3) \equiv (\varphi_1 U \varphi_2) \lor (\varphi_1 U (G \varphi_3))$, we only need to prove $\varphi_1 U (G \varphi_3) \equiv G \varphi_3$. Clearly, $G \varphi_3$ implies $\varphi_1 U G \varphi_3$. Since for any cyclic word $\sigma^o \models \varphi_1 U G \varphi_3$, $\exists j \geq 0$ such that $\sigma^o|_j \models G \varphi_3$, which implies $\sigma^o \models G \varphi_3$. Thus the claim holds.

- $(\varphi_1 \lor G \varphi_2) U \varphi_3 \equiv (G \varphi_2 \land F \varphi_3) \lor (\varphi_1 U \varphi_3)$.

Let $\varphi = (\varphi_1 \lor G \varphi_2) U \varphi_3$, $\psi = (G \varphi_2 \land F \varphi_3) \lor (\varphi_1 U \varphi_3)$ and $\sigma \in (2^{AP^*})^c$.

\[ \Rightarrow: \sigma^o \models (\varphi_1 \lor G \varphi_2) U \varphi_3, \text{ then } \exists j \geq 0 \text{ such that } \sigma^o|_j \models \varphi_3 \text{ and } \forall 0 \leq i < j, \sigma^o|_i \models \varphi_1 \lor G \varphi_2. \]
Case i), $\exists 0 \leq i < j$ such that $\sigma^o|_i \models G \varphi_2$, which implies $\sigma^o \models G \varphi_2$. Thus $\sigma^o \models G \varphi_2 \land F \varphi_3$. Case ii) is trivial, as $\forall 0 \leq i < j$, $\sigma^o|_i \models \varphi_2$, which directly conclude $\sigma^o \models \varphi_1 U \varphi_3$.

\[ \Leftrightarrow: \sigma^o \models \psi, \text{ we have either i) } \sigma^o \models (G \varphi_2 \land F \varphi_3) \text{ or ii) } \sigma^o \models (\varphi_1 U \varphi_3). \]
For i), note that $\sigma^o \models G \varphi_2$ implies all suffixes of $\sigma^o$ satisfy $G \varphi_2$. Together with the fact that $\sigma^o \models (G \varphi_3)$, we have $\sigma^o \models (G \varphi_2 U \varphi_3)$, which implies $\sigma^o \models \varphi$. Case ii) is trivial, as $\sigma^o \models (\varphi_1 U \varphi_3)$ implies $\sigma^o \models \varphi$.

- $(\varphi_1 \lor F \varphi_2) U \varphi_3 \equiv (F \varphi_2 \land F \varphi_3) \lor (\varphi_1 U \varphi_3)$.

Let $\varphi = (\varphi_1 \lor F \varphi_2) U \varphi_3$, $\psi = (F \varphi_2 \land F \varphi_3) \lor (\varphi_1 U \varphi_3)$ and $\sigma \in (2^{AP^*})^c$.

\[ \Rightarrow: \text{Since } \sigma^o \models \varphi, \text{ we have } \exists j \geq 0 \text{ such that } \sigma^o|_j \models \varphi_3 \text{ and } \forall 0 \leq i < j, \sigma^o|_i \models \varphi_1 \lor F \varphi_2. \]
Case i), $\exists 0 \leq i < j$ such that $\sigma^o|_i \models F \varphi_2$, then $\sigma^o|_k \models F \varphi_2 \land F \varphi_3$. Case ii), $\forall 0 \leq i < j$, $\sigma^o|_i \models \varphi_1$, which directly concludes $\sigma^o \models \varphi_1 U \varphi_3$. Thus $\sigma^o \models \psi$.

\[ \Leftrightarrow: \sigma^o \models \psi \text{ includes two cases. Case i), } \sigma^o \models F \varphi_2 \land F \varphi_3. \]
Since $\sigma^o$ is cyclic, $\exists j \geq 0$ such that $\sigma^o|_j \models \varphi_2$, then $\forall i \geq 0, \sigma^o|_i \models F \varphi_2$. Thus $\sigma^o \models (F \varphi_2) U \varphi_3$, which implies $\sigma^o \models (\varphi_1 \lor F \varphi_2) U \varphi_3$. Case ii), $\sigma^o \models \varphi_1 U \varphi_3$, which implies $\sigma^o \models (\varphi_1 \lor F \varphi_2) U \varphi_3$.

- $(\varphi_1 \land F \varphi_2) U \varphi_3 \equiv (F \varphi_2 \land (\varphi_1 U \varphi_3)) \lor \varphi_3$.

Let $\varphi = (\varphi_1 \land F \varphi_2) U \varphi_3$, $\psi = (F \varphi_2 \land (\varphi_1 U \varphi_3)) \lor \varphi_3$ and $\sigma \in (2^{AP^*})^c$.

\[ \Rightarrow: \text{Trivial since } \varphi \text{ implies } \varphi_3. \]
\[ \Leftrightarrow: \sigma^o \models \psi \text{ includes two cases. Case i), } \sigma^o \models \varphi_3, \text{ it is obvious that } \psi \text{ implies } \varphi. \]
Case ii), $\sigma^o \models F \varphi_2 \land (\varphi_1 U \varphi_3)$. Since $\exists j \geq 0$ such that $\sigma^o|_j \models \varphi_2$, then $\forall i \geq 0, \sigma^o|_i \models F \varphi_2$. Thus $\sigma^o \models (\varphi_1 \land F \varphi_2) U \varphi_3$.

- $(\varphi_1 \land G \varphi_2) U \varphi_3 \equiv (G \varphi_2 \land (\varphi_1 U \varphi_3)) \lor \varphi_3$.

Let $\varphi = (\varphi_1 \land G \varphi_2) U \varphi_3$, $\psi = (G \varphi_2 \land (\varphi_1 U \varphi_3)) \lor \varphi_3$ and $\sigma \in (2^{AP^*})^c$.

\[ \Rightarrow: \text{Trivial since } \varphi \text{ implies } \varphi_3. \]
\[ \Leftrightarrow: \sigma^o \models \psi \text{ consists of two cases. Case i), } \sigma^o \models \varphi_3, \text{ it is obvious that } \psi \text{ implies } \varphi. \]
Case ii), $\sigma^o \models G \varphi_2 \land (\varphi_1 U \varphi_3)$. Since $\sigma^o \models G \varphi_2$ implies $\forall i \geq 0, \sigma^o|_i \models G \varphi_2$. Thus $\sigma^o \models (\varphi_1 \land G \varphi_2) U \varphi_3$. This completes the proof. Note that using above equations, we can get all $F$ and $G$ modalities out of the scope of the modalities $U$. \hfill \blacksquare