State/Event-Based LTL Model Checking under Parametric Generalized Fairness

Kyungmin Bae and José Meseguer

Department of Computer Science,
University of Illinois at Urbana-Champaign, Urbana IL 61801
{kbae4,meseguer}@cs.uiuc.edu

Abstract. In modeling a concurrent system, fairness constraints are usually considered at a specific granularity level of the system, leading to many different variants of fairness: transition fairness, object/process fairness, actor fairness, etc. These different notions of fairness can be unified by making explicit their *parametrization* over the relevant entities in the system as *universal quantification*. We propose a *state/event-based* framework as well as an on-the-fly model checking algorithm to verify LTL properties under universally quantified parametric fairness assumptions, specified by *generalized* strong/weak fairness formulas. It enables verification of temporal properties under fairness conditions associated to dynamic entities such as new process creations. We have implemented our algorithm within the Maude system.

Keywords: Model checking, Parameterized Fairness, State/Event LTL.

1 Introduction

Fairness assumptions are often necessary to verify a liveness property of a concurrent system. Without fairness, unrealistic counterexamples can be produced, such as a process that is never executed even though the process is continuously enabled. A usual way to model check an LTL property under fairness assumptions is to re-formulate the property such that fairness requirements become a conjunction of premises implying the original property [6]. Since this method is impractical for model checking properties under complex fairness assumptions, several specialized algorithms have been proposed, e.g., [11,14,16,19].

In practice, however, descriptions of fairness are dependent on specific models or languages. There are many different variants of the fairness concepts, such as transition fairness, object/process fairness, actor fairness, etc [1,13]. In general, such variants do not coincide, even though their temporal behaviors like strong/weak fairness are all similar. It becomes difficult to represent fairness notions which are not directly supported by a specific modeling language.

Such different variants of fairness can be unified by making explicit the *parametrization* of fairness formulas over the relevant spatial entities in a system [20]. Fairness is then expressed by a *universally quantified* temporal formula, where variables in the formula range over the relevant system entities. We use a

G. Gopalakrishnan and S. Qadeer (Eds.): CAV 2011, LNCS 6806, pp. 132–148, 2011.
© Springer-Verlag Berlin Heidelberg 2011

state/event-based LTL (SE-LTL) because fairness notions usually involve both states and events. For example, weak process fairness can be expressed by the universally quantified formula: $\forall x \in \mathit{Process}.\ \Diamond\Box \mathit{enabled}(x) \rightarrow \Box\Diamond \mathit{execute}(x)$, where $\mathit{enabled}(x)$ is a state predicate and $\mathit{execute}(x)$ is an event predicate.

We present a framework to verify SE-LTL properties under parameterized fairness constraints, given by generalized strong (resp., weak) fairness formulas of the form $\forall \overline{x}\ \Box\Diamond\Phi \rightarrow \Box\Diamond\Psi$ (resp., $\forall \overline{x}\ \Diamond\Box\Phi \rightarrow \Box\Diamond\Psi$) in SE-LTL. For parameterized fairness, the number of entities in the system over which the parametrization ranges can be unbounded[1] and may change during execution. Instead, previous approaches require that the number of fairness conditions is already determined. For example, in process fairness with dynamic process creation, fairness is parametric over a number of processes unknown a priori. Our framework is based on the notion of *parameter abstraction* to make explicit the fact that, even though the domain of entities or parameters is infinite, only a *finite* number of parameters are meaningful in a single state for fairness purposes. In process algebra, meaningful parameters are the processes in the state, and strong/weak fairness is vacuously satisfied for the processes not existing in a system.[2]

We have developed an on-the-fly SE-LTL model checking algorithm (available at [2]) using a parameter abstraction that can directly handle universally quantified fairness formulas; its complexity is linear in the number of fairness instances (see Sec. 4). We have implemented our algorithm within the Maude system [7], which is a verification framework for concurrent systems where many concurrent systems, including object-based systems and process algebras, can be naturally described. This model checking algorithm can verify liveness properties of complex examples with dynamic entities having an unpredictable number of fairness assumptions (see Sec. 5). To the best of our knowledge, such dynamic parametric fairness assumptions cannot be easily handled by other existing model checkers.

Related Work. Parameterization has long been considered as a way to describe fairness of concurrent systems. The theorem proving of liveness properties commonly involves parameterized fairness properties. Fairness notions supported by usual modeling languages are parameterized, e.g., process fairness [13] is parameterized by processes. However, such fairness notions are parameterized *only* by specific entities, depending on the system modeling language. Localized fairness [20] was introduced as a unified notion to express different variants of fairness, depending on the chosen system granularity level. Localized fairness can be parameterized by *any* entity in a system, but generalized versions of strong/weak fairness were not discussed in [20]. Our work extends localized fairness to incorporate generalized fairness, and answers the question of how to model check LTL properties under such generalized fairness conditions.

To verify a property φ under parameterized fairness assumptions, the usual method is to construct the conjunction of corresponding instances of fairness,

[1] For finite-state systems the number is finite, but it may be impossible to determine such a number from the initial state without exploring the entire state space.

[2] E.g., $\mathit{enabled}(p)$ is false for all states if a process p does not exist in the system, and therefore, $\Diamond\Box \mathit{enabled}(p) \rightarrow \Box\Diamond \mathit{execute}(p)$ is vacuously satisfied.

and to apply either: (i) a standard LTL model checking algorithm for the reformulated property $fair \rightarrow \varphi$, or (ii) a specialized model checking algorithm which handles fairness, based on either explicit graph search [11,14,19], or a symbolic algorithm [16]. Approach (i) is inadequate for fairness, since the time complexity is exponential in the number of strong fairness conditions, while the latter is linear. Furthermore, compiling such a formula, expressing a conjunction of fairness conditions, into Büchi automata is usually not feasible in reasonable time [24]. There are several tools to support the specialized algorithms such as PAT [22] and Maria [19]. Our algorithm is related to the second approach to deal with fairness, but it does not require pre-translation of parameterized fairness, and can handle *dynamic* generalized fairness instances. There are also other methods to support parameterized fairness not based on standard model checking methods, such as regular model checking [4] and compositional reasoning [8].

This paper is organized as follows. Section 2 presents the necessary background about fairness and SE-LTL. Section 3 introduces a logical framework for parameterized fairness, and Section 4 presents the automata-based model checking algorithm under parameterized fairness. Section 5 illustrates case studies, Section 6 shows experimental results, and Section 7 presents some conclusions.

2 Fairness Expressed in a State/Event-Based Logic

Fairness generally means that, if a certain kind of choice is sufficiently often provided, then it is sufficiently often taken. For example, strong fairness means that, if a given choice is available infinitely often, then it is taken infinitely often. Similarly, weak fairness means that, if the choice is continuously available beyond a certain point, then it is taken infinitely often.

In order to express fairness using *only* logic formulas, we need a logic to specify properties involving both states and events. Neither state-based logics such as LTL nor event-based logics are usually sufficient to express fairness as logic formulas on the original system, although system transformations can be used to "encode" events in the state, typically at the price of a bigger state space. Many modeling languages using state-based logics incorporate specific kinds of fairness properties to avoid such problems, but the expressiveness of fairness is then limited to the given kind of fairness thus supported.

State/event linear temporal logic (SE-LTL) [5] is a simple state/event extension of linear temporal logic. The only syntactic difference between LTL and SE-LTL is that the latter can have both state propositions and event propositions. Given a set of state propositions AP and a set of event propositions ACT, the syntax of SE-LTL formulas over AP and ACT is defined by $\varphi ::= p \mid \delta \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \bigcirc\varphi \mid \varphi \mathbf{U} \varphi'$, where $p \in AP$ and $\delta \in ACT$. Other operators can be defined by equivalences, e.g., $\Diamond\varphi \equiv true\mathbf{U}\varphi$ and $\Box\varphi \equiv \neg\Diamond\neg\varphi$.

The semantics of SE-LTL is defined on a *labeled Kripke structure* (LKS), which is a natural extension of a Kripke structure with transition labels. The model checking problem of SE-LTL formulas can be characterized by automata-theoretic techniques on LKS similar to the LTL case [3,5], which use the Büchi

automaton $\mathscr{B}_{\neg\varphi}$ with size $O(2^{|\varphi|})$ associated to the negated formula $\neg\varphi$, where the alphabet of $\mathscr{B}_{\neg\varphi}$ is a set of subsets of the disjoint union $AP \uplus ACT$.

Definition 1. *A* labeled Kripke structure *is a 6-tuple* $(S, S_0, AP, \mathcal{L}, ACT, T)$ *with* S *a set of* states, $S_0 \subseteq S$ *a set of* initial states, AP *a set of* atomic state propositions, $\mathcal{L} : S \to \mathcal{P}(AP)$ *a* state-labeling function, ACT *a set of* atomic events, *and* $T \subseteq S \times \mathcal{P}(ACT) \times S$ *a* labeled transition relation.

Note that each transition of an LKS is labeled by a *set* A of atomic events, which enables us to describe a pattern of an event. A labeled transition $(s, A, s') \in T$ is often denoted by $s \xrightarrow{A} s'$. A *path* (π, α) of an LKS is an infinite sequence $\langle \pi(0), \alpha(0), \pi(1), \alpha(1), \ldots \rangle$ such that $\pi(i) \in S$, $\alpha(i) \subseteq ACT$, and $\pi(i) \xrightarrow{\alpha(i)} \pi(i+1)$ for each $i \geq 0$. $(\pi, \alpha)^i$ denotes the suffix of (π, α) beginning at position $i \in \mathbb{N}$. A SE-LTL formula φ is satisfied on an LKS $\mathcal{K}$, denoted by $\mathcal{K} \models \varphi$, if and only if for each path (π, α) of $\mathcal{K}$ starting from an initial state, the path satisfaction relation $\mathcal{K}, (\pi, \alpha) \models \varphi$ holds, which is defined inductively as follows:

- $\mathcal{K}, (\pi, \alpha) \models p$ iff $p \in \mathcal{L}(\pi(0))$
- $\mathcal{K}, (\pi, \alpha) \models \delta$ iff $\delta \in \alpha(0)$
- $\mathcal{K}, (\pi, \alpha) \models \neg\varphi$ iff $\mathcal{K}, (\pi, \alpha) \not\models \varphi$
- $\mathcal{K}, (\pi, \alpha) \models \varphi \wedge \varphi'$ iff $\mathcal{K}, (\pi, \alpha) \models \varphi$ and $\mathcal{K}, (\pi, \alpha) \models \varphi'$
- $\mathcal{K}, (\pi, \alpha) \models \bigcirc\varphi$ iff $\mathcal{K}, (\pi, \alpha)^1 \models \varphi$
- $\mathcal{K}, (\pi, \alpha) \models \varphi \,\mathbf{U}\, \varphi'$ iff $\exists k \geq 0.\ \mathcal{K}, (\pi, \alpha)^k \models \varphi',\ \forall 0 \leq i < k.\ \mathcal{K}, (\pi, \alpha)^i \models \varphi$

We can define fairness properties of an LKS as SE-LTL formulas. A strong fairness (resp. weak fairness) condition with respect to an event proposition α is expressed by the SE-LTL formula $\Box\Diamond \mathit{enabled}.\alpha \to \Box\Diamond\alpha$ (resp., $\Diamond\Box \mathit{enabled}.\alpha \to \Box\Diamond\alpha$). A special state proposition $\mathit{enabled}.\alpha$ is defined for each state s of $\mathcal{K}$ such that $\mathit{enabled}.\alpha \in \mathcal{L}(s)$ iff there exists a transition $s \xrightarrow{A} s' \in T$ with $\alpha \in A$. *Generalized* strong (resp., weak) fairness conditions are defined by SE-LTL formulas of the form $\Box\Diamond\Phi \to \Box\Diamond\Psi$ (resp, $\Diamond\Box\Phi \to \Box\Diamond\Psi$), where Φ and Ψ are Boolean formulas that do not contain any temporal operators. Many fairness notions, that arise in real examples, can be expressed by generalized strong/weak fairness formulas [18]. For example, the minimal progress of a set of events $\{\alpha_1, \ldots, \alpha_k\}$ can be expressed by $\Diamond\Box(\mathit{enabled}.\alpha_1 \vee \cdots \vee \mathit{enabled}.\alpha_k) \to \Box\Diamond(\alpha_1 \vee \cdots \vee \alpha_k)$. However, imposing such fairness conditions for each relevant entity, e.g., for each process, may require a large or even infinite set of such formulas.

3 Parameterized Fairness as Quantified SE-LTL

Besides a temporal perspective regarding frequency of a choice, fairness also has a spatial perspective depending on the relation between the choice and the system structure. The variants of fairness from such system structures can be unified by making explicit their *parametrization* over the chosen spatial entities [20]. To specify parameterized fairness conditions, we use first-order SE-LTL over parameterized propositions. Fairness is then expressed by a universally quantified SE-LTL formula, where variables range over the relevant entities in the system.

3.1 Quantification of Parametric SE-LTL Formulas

In order to define parametric SE-LTL formulas, we should make the state and event propositions parametric on the relevant entities. Such entities need not be states: they could be process names, messages, or other data structures. Therefore, we allow *parametric* state propositions $p \in \Pi$ (resp., event propositions $\delta \in \Omega$) of the form $p(x_1, \ldots, x_n)$ (resp., $\delta(x_1, \ldots, x_m)$).

Definition 2. *A* parameterized labeled Kripke structure *over a set of parameters $\mathcal{C}$ is a tuple $\mathcal{K} = (S, S_0, \Pi, \mathcal{L}, \Omega, T)$ such that $\mathcal{K}_{\mathcal{C}} = (S, S_0, AP_{\mathcal{C}}, \mathcal{L}, ACT_{\mathcal{C}}, T)$ is an ordinary LKS with state propositions $AP_{\mathcal{C}} = \{p(\overline{a}) \mid \overline{a} \in \mathcal{C}^n, p \in \Pi, n \in \mathbb{N}\}$ and event propositions $ACT_{\mathcal{C}} = \{\delta(\overline{b}) \mid \overline{b} \in \mathcal{C}^m, \delta \in \Omega, m \in \mathbb{N}\}$.*

We can now define the set of *universally quantified SE-LTL formulas* with respect to Π, Ω, and $\mathcal{C}$ as the set of formulas of the form $\forall \overline{x}\ \varphi$, where φ is a propositional SE-LTL formula over $AP_{\mathcal{C}\cup\mathcal{V}} = \{p(\overline{a}) \mid \overline{a} \in (\mathcal{C} \cup \mathcal{V})^n, p \in \Pi, n \in \mathbb{N}\}$ and $ACT_{\mathcal{C}\cup\mathcal{V}} = \{\delta(\overline{b}) \mid \overline{b} \in (\mathcal{C} \cup \mathcal{V})^m, \delta \in \Omega, m \in \mathbb{N}\}$, with $\mathcal{V}$ an infinite set of variables disjoint from $\mathcal{C}$, and $\overline{x} = vars(\varphi)$ the set of variables occurring in φ. The *satisfaction* of such formulas in a path (π, α) of a parameterized LKS $\mathcal{K} = (S, S_0, \Pi, \mathcal{L}, \Omega, T)$ is now defined in the obvious way:

$$\mathcal{K}, (\pi, \alpha) \models \forall \overline{x}\ \varphi \quad \Leftrightarrow \quad \forall (\theta : \overline{x} \to \mathcal{C}).\ \mathcal{K}_{\mathcal{C}}, (\pi, \alpha) \models \theta\varphi$$

where $\theta\varphi$ is the propositional SE-LTL formula obtained by applying the simultaneous substitution θ to the variables $\overline{x}$ in φ. Note that $\mathcal{K} \models \forall \overline{x}\ \varphi$ iff $\mathcal{K}, (\pi, \alpha) \models \forall \overline{x}\ \varphi$ for each path (π, α) starting from an initial state.

A parameterized strong (resp., weak) fairness formula from Φ to Ψ is a universally quantified SE-LTL formula $\forall \overline{x}\ \Box\Diamond\Phi \to \Box\Diamond\Psi$ (resp, $\forall \overline{x}\ \Diamond\Box\Phi \to \Box\Diamond\Psi$), where Φ and Ψ are Boolean formulas. Consider sets of strong (resp., weak) parameterized fairness formulas $\mathscr{F} = \{\forall \overline{x}_i\ \Box\Diamond\Phi_i \to \Box\Diamond\Psi_i \mid i \in I\}$ (resp., $\mathscr{J} = \{\forall \overline{x}_j\ \Diamond\Box\Phi_j \to \Box\Diamond\Psi_j \mid j \in J\}$), where I and J are index sets. A path (π, α) of $\mathcal{K}$ is *fair under* parameterized fairness $\mathscr{F} \cup \mathscr{J}$ iff $\mathcal{K}, (\pi, \alpha) \models \forall \overline{x}_i\ \Box\Diamond\Phi_i \to \Box\Diamond\Psi_i$ and $\mathcal{K}, (\pi, \alpha) \models \forall \overline{x}_j\ \Diamond\Box\Phi_j \to \Box\Diamond\Psi_j$ for each $i \in I$, $j \in J$. A SE-LTL formula φ is *fairly* satisfied on $\mathcal{K}$ under $\mathscr{F} \cup \mathscr{J}$, denoted by $\mathcal{K} \models_{\mathscr{F}\cup\mathscr{J}} \varphi$, iff $\mathcal{K}, (\pi, \alpha) \models \varphi$ for each fair path (π, α) under $\mathscr{F} \cup \mathscr{J}$ starting from any initial state of $\mathcal{K}$.

3.2 Finite Instantiation Property and Parameter Abstraction

Although the parameter set $\mathcal{C}$ is not a subset of the set S of states, there is however an implicit relation between $\mathcal{C}$ and S derived from an underlying LKS $\mathcal{K}$, in terms of a definable set. If $[\overline{x} \to \mathcal{C}]$ denotes the set of all substitutions $\theta : \overline{x} \to \mathcal{C}$, given a transition $s \xrightarrow{A} s'$ of $\mathcal{K}$, the *proposition-definable* sets $\mathcal{D}_{s,A}(p(\overline{x}))$ and $\mathcal{D}_{s,A}(\delta(\overline{x}))$ are the sets of substitutions that make the propositions satisfied:

$$\mathcal{D}_{s,A}(p(\overline{x})) = \{\theta \in [\overline{x} \to \mathcal{C}] \mid \theta p(\overline{x}) \in \mathcal{L}(s)\} \quad \mathcal{D}_{s,A}(\delta(\overline{x})) = \{\theta \in [\overline{x} \to \mathcal{C}] \mid \theta\delta(\overline{x}) \in A\}$$

In practice, the number of parameters $c \in \mathcal{C}$ that *occur* in a state is finite. For that reason, assuming that $\mathcal{L}(s)$, $\mathcal{L}(s')$, and A are finite for each transition $s \xrightarrow{A} s'$, the proposition-definable sets for all state and event propositions are finite. This is captured by the following finite instantiation property.

Definition 3. *A parameterized LKS* $\mathcal{K} = (S, S_0, \Pi, \mathcal{L}, \Omega, T)$ *over parameters* $\mathcal{C}$ *satisfies* finite instantiation property (FIP) *with respect to* $\Pi' \subseteq \Pi$ *and* $\Omega' \subseteq \Omega$ *if for each transition* $s \xrightarrow{A} s'$ *of* $\mathcal{K}$*, the sets* $\mathcal{D}_{s,A}(p(\overline{x}))$ *for each* $p \in \Pi'$ *and* $\mathcal{D}_{s,A}(\delta(\overline{x}))$ *for each* $\delta \in \Omega'$ *are finite.*

For an LKS $\mathcal{K}$ satisfying FIP with respect to Π' and Ω' over a parameter set $\mathcal{C}$ we can define abstraction of substitutions $\theta : \overline{x} \to \mathcal{C}$ with respect to proposition-definable sets of Π' and Ω'. The key idea is to collapse the cofinite[3] complement of each proposition-definable set into the abstracted substitution $\bot_{\overline{x}} : \overline{x} \to \{\bot\}$ with a fresh new constant $\bot$, which denotes a parameter that does *never* appear in the finite definable set. For example, for a state proposition $p(\overline{x})$, each substitution $\theta \notin \mathcal{D}_{s,A}(p(\overline{x}))$ is abstracted to $\bot_{\overline{x}} : \overline{x} \to \{\bot\}$. The extended parameter set $\mathcal{C}_\bot = \mathcal{C} \cup \{\bot\}$ involves the LKS $\mathcal{K}_{\mathcal{C}_\bot} = (S, S_0, AP_{\mathcal{C}_\bot}, \mathcal{L}, ACT_{\mathcal{C}_\bot}, T)$ naturally extending $\mathcal{K}_\mathcal{C} = (S, S_0, AP_\mathcal{C}, \mathcal{L}, ACT_\mathcal{C}, T)$ to $\mathcal{C}_\bot$. In this case, the negated satisfaction relation $\mathcal{K}_{\mathcal{C}_\bot}, (\pi, \alpha) \not\models \bot_{\overline{x}} p(\overline{x})$ holds, as $\bot_{\overline{x}} p(\overline{x}) \in \mathcal{L}(\pi(0))$ is impossible.

This abstraction relation can be extended to any SE-LTL formula using a natural ordering in the abstracted domain $[\overline{x} \to \mathcal{C}_\bot]$. A partial order relation $\preceq$ between substitutions $\theta_1, \theta_2 \in [\overline{x} \to \mathcal{C}_\bot]$ is defined by:

$$\theta_1 \preceq \theta_2 \;\Leftrightarrow\; \text{for each } x \in \overline{x},\ \theta_1(x) = \bot \text{ or } \theta_1(x) = \theta_2(x)$$

Given a pair θ_1, θ_2 of substitutions that have a common upper bound, i.e., $\theta_1 \preceq \theta$ and $\theta_2 \preceq \theta$ for some θ, there is the least upper bound defined by:

$$\theta_1 \vee \theta_2 = (\theta_1 \vee \theta_2)(x) = \theta_1(x) \vee \theta_2(x) \text{ for each } x \in \overline{x}$$

where $c \vee \bot = \bot \vee c = c \vee c = c$ for each $c \in \mathcal{C}$. For substitutions θ_1, θ_2 with possibly different domains, we can define the combined substitution $\theta_1 \oplus \theta_2$:

$$\theta_1 \oplus \theta_2(x) = \begin{cases} \theta_1(x) & \text{if } x \in \mathit{dom}(\theta_1) \setminus \mathit{dom}(\theta_2) \\ \theta_2(x) & \text{if } x \in \mathit{dom}(\theta_2) \setminus \mathit{dom}(\theta_1) \\ \theta_1(x) \vee \theta_2(x) & \text{otherwise} \end{cases}$$

The *abstraction function* $\varrho_{(\pi,\alpha),\varphi} : [\mathit{vars}(\varphi) \to \mathcal{C}_\bot] \to [\mathit{vars}(\varphi) \to \mathcal{C}_\bot]$ is then inductively defined for a SE-LTL formula φ and a path (π, α) as follows:

- $\varrho_{(\pi,\alpha),p(\overline{x})}(\theta) = \textbf{if } \theta \in \mathcal{D}_{\pi(0),\alpha(0)}(p(\overline{x})) \textbf{ then } \theta \textbf{ else } \bot_{\overline{x}} \textbf{ fi}$
- $\varrho_{(\pi,\alpha),\delta(\overline{x})}(\theta) = \textbf{if } \theta \in \mathcal{D}_{\pi(0),\alpha(0)}(\delta(\overline{x})) \textbf{ then } \theta \textbf{ else } \bot_{\overline{x}} \textbf{ fi}$
- $\varrho_{(\pi,\alpha),\neg\varphi}(\theta) = \varrho_{(\pi,\alpha),\varphi}(\theta)$
- $\varrho_{(\pi,\alpha),\varphi_1 \wedge \varphi_2}(\theta) = \varrho_{(\pi,\alpha),\varphi_1}(\theta{\restriction}_{\mathit{vars}(\varphi_1)}) \oplus \varrho_{(\pi,\alpha),\varphi_2}(\theta{\restriction}_{\mathit{vars}(\varphi_2)})$
- $\varrho_{(\pi,\alpha),\bigcirc\varphi}(\theta) = \varrho_{(\pi,\alpha)^1,\varphi}(\theta)$
- $\varrho_{(\pi,\alpha),\varphi_1 \mathbf{U} \varphi_2}(\theta) = \bigvee_{i \geq 0} \varrho_{(\pi,\alpha)^i,\varphi_1}(\theta{\restriction}_{\mathit{vars}(\varphi_1)}) \;\oplus\; \bigvee_{j \geq 0} \varrho_{(\pi,\alpha)^j,\varphi_2}(\theta{\restriction}_{\mathit{vars}(\varphi_2)})$

Note that $\varrho_{(\pi,\alpha),\varphi_1 \wedge \varphi_2}(\theta)$ and $\varrho_{(\pi,\alpha),\varphi_1 \mathbf{U} \varphi_2}(\theta)$ are well-defined since $\varrho_{(\pi,\alpha),\varphi}(\theta) \preceq \theta$ is satisfied by induction. The satisfaction relation of φ on (π, α) for an abstracted substitution $\vartheta = \varrho_{(\pi,\alpha),\varphi}(\theta)$ is naturally defined by $\mathcal{K}_{\mathcal{C}_\bot}, (\pi, \alpha) \models \vartheta\varphi$. The following lemma asserts that the satisfaction of a formula on an LKS satisfying FIP is preserved by the abstraction function $\varrho_{(\pi,\alpha),\varphi}$ of substitutions.

[3] A set is cofinite iff the complement of the set is finite.

Lemma 1. *Given an LKS $\mathcal{K}$ satisfying FIP over a parameter set $\mathcal{C}$, a quantified SE-LTL formula $\forall \overline{x}\ \varphi$, and a substitution $\theta \in [\overline{x} \to \mathcal{C}]$, for each path (π, α), $\mathcal{K}_{\mathcal{C}}, (\pi, \alpha) \models \theta\varphi$ iff $\mathcal{K}_{\mathcal{C}_\perp}, (\pi, \alpha) \models \varrho_{(\pi,\alpha),\varphi}(\theta)\varphi$.*

Proof. We show the following generalized version of the lemma by structural induction on φ: $\mathcal{K}_{\mathcal{C}}, (\pi, \alpha) \models \theta\varphi$ iff $\mathcal{K}_{\mathcal{C}_\perp}, (\pi, \alpha) \models \vartheta\varphi$ for any substitution $\vartheta \in [\overline{x} \to \mathcal{C}_\perp]$ such that $\varrho_{(\pi,\alpha),\varphi}(\theta) \preceq \vartheta \preceq \theta$. For a state proposition $p(\overline{x})$, $\mathcal{K}_{\mathcal{C}}, (\pi, \alpha) \not\models \theta p(\overline{x})$ iff $\theta \notin \mathcal{D}_{\pi(0),\alpha(0)}(p(\overline{x}))$ iff $\varrho_{(\pi,\alpha),p(\overline{x})}(\theta) = \perp_{\overline{x}}$, and for each substitution $\perp_{\overline{x}} \preceq \vartheta \prec \theta$, $\mathcal{K}_{\mathcal{C}_\perp}, (\pi, \alpha) \not\models \vartheta\varphi$. To prove the $\varphi_1 \mathbf{U} \varphi_2$ case, we need the following ordering properties of $\varrho_{(\pi,\alpha),\varphi}$, which are easy consequences from the definition: (i) $\varrho_{(\pi,\alpha)^i,\varphi_1}(\theta) \preceq \varrho_{(\pi,\alpha),\varphi_1\mathbf{U}\varphi_2}(\theta)\restriction_{dom(\theta_1)}$, and (ii) $\varrho_{(\pi,\alpha)^i,\varphi_2}(\theta) \preceq \varrho_{(\pi,\alpha),\varphi_1\mathbf{U}\varphi_2}(\theta)\restriction_{dom(\theta_2)}$, for each $i \geq 0$. Thus, for each $\varrho_{(\pi,\alpha),\varphi_1\mathbf{U}\varphi_2}(\theta) \preceq \vartheta \preceq \theta$, if $\overline{y} = vars(\varphi_1)$ and $\overline{z} = vars(\varphi_2)$, we have $\varrho_{(\pi,\alpha)^i,\varphi_1}(\theta \restriction_{\overline{y}}) \preceq \vartheta \restriction_{\overline{y}} \preceq \theta \restriction_{\overline{y}}$ and $\varrho_{(\pi,\alpha)^i,\varphi_2}(\theta \restriction_{\overline{z}}) \preceq \vartheta \restriction_{\overline{z}} \preceq \theta \restriction_{\overline{z}}$, $i \geq 0$. Hence, by induction hypothesis, we have $\mathcal{K}_{\mathcal{C}}, (\pi, \alpha)^i \models \theta\varphi_1$ iff $\mathcal{K}_{\mathcal{C}_\perp}, (\pi, \alpha)^i \models \vartheta\varphi_1$, and $\mathcal{K}_{\mathcal{C}}, (\pi, \alpha)^j \models \theta\varphi_2$ iff $\mathcal{K}_{\mathcal{C}_\perp}, (\pi, \alpha)^j \models \vartheta\varphi_2$ for each $i, j \geq 0$. Therefore, $\mathcal{K}_{\mathcal{C}}, (\pi, \alpha) \models \theta(\varphi_1 \mathbf{U} \varphi_2)$ iff $\mathcal{K}_{\mathcal{C}_\perp}, (\pi, \alpha) \models \vartheta(\varphi_1 \mathbf{U} \varphi_2)$. The other cases are similar. □

On the other hand, as a dual of $\varrho_{(\pi,\alpha),\varphi}$, the *concretization* function $I_{(\pi,\alpha),\varphi} : [vars(\varphi) \to \mathcal{C}_\perp] \to \mathcal{P}([vars(\varphi) \to \mathcal{C}])$ can be defined for a SE-LTL formula φ as follows, where $[\overline{x} \to \mathcal{C}]_{\succeq\vartheta}$ denotes the set $\{\theta \in [\overline{x} \to \mathcal{C}] \mid \vartheta \preceq \theta\}$ and the "glueing" $I_1 \odot I_2$ of two sets I_1 and I_2 of concrete substitutions is defined by $I_1 \odot I_2 = \{\theta \mid \theta\restriction_{dom(I_1)} \in I_1,\ \theta\restriction_{dom(I_2)} \in I_2\}$:

- $I_{(\pi,\alpha),p(\overline{x})}(\vartheta) = \textbf{if } \vartheta \in D \textbf{ then } \vartheta \textbf{ else } [\overline{x} \to \mathcal{C}]_{\succeq\vartheta} \setminus D \textbf{ fi},\ D = \mathcal{D}_{\pi(0),\alpha(0)}(p(\overline{x}))$
- $I_{(\pi,\alpha),\delta(\overline{x})}(\vartheta) = \textbf{if } \vartheta \in D \textbf{ then } \vartheta \textbf{ else } [\overline{x} \to \mathcal{C}]_{\succeq\vartheta} \setminus D \textbf{ fi},\ D = \mathcal{D}_{\pi(0),\alpha(0)}(\delta(\overline{x}))$
- $I_{(\pi,\alpha),\neg\varphi}(\vartheta) = I_{(\pi,\alpha),\varphi}(\vartheta)$
- $I_{(\pi,\alpha),\varphi_1\wedge\varphi_2}(\vartheta) = I_{(\pi,\alpha),\varphi_1}(\vartheta\restriction_{vars(\varphi_1)}) \odot I_{(\pi,\alpha),\varphi_2}(\vartheta\restriction_{vars(\varphi_2)})$
- $I_{(\pi,\alpha),\bigcirc\varphi}(\vartheta) = I_{(\pi,\alpha)^1,\varphi}(\vartheta)$
- $I_{(\pi,\alpha),\varphi_1\mathbf{U}\varphi_2}(\vartheta) = \bigcap_{i\geq 0} I_{(\pi,\alpha)^i,\varphi_1}(\vartheta)\ \odot\ \bigcap_{j\geq 0} I_{(\pi,\alpha)^j,\varphi_2}(\vartheta)$

It is easy to check that for each $\theta \in I_{(\pi,\alpha),\varphi}(\vartheta)$, $\vartheta \preceq \theta$ and $\varrho_{(\pi,\alpha),\varphi}(\vartheta) = \varrho_{(\pi,\alpha),\varphi}(\theta)$. The abstraction of a concrete substitution does always exist, but there may be *no* concretization for some abstracted substitution. For instance, given a path (π, α) such that $\mathcal{D}_{\pi(i),\alpha(i)}(p(x)) = \{i\}$ for each $i \geq 0$, if $\mathcal{C} = \mathbb{N}$, then $\varrho_{(\pi,\alpha),\Diamond p(x)}(\theta) = \theta$ for any $\theta \in [\{x\} \to \mathbb{N}]$, and $I_{(\pi,\alpha),\Diamond p(x)}(\perp_x) = \varnothing$. However, for a *finite* LKS that has only a finite set of states and a finite set of transitions, each abstracted substitution has a corresponding concrete substitution.

Lemma 2. *Given a* finite *LKS $\mathcal{K}$ satisfying FIP over parameters $\mathcal{C}$, a quantified SE-LTL formula $\forall \overline{x}\ \varphi$, and $\vartheta \in [\overline{x} \to \mathcal{C}_\perp]$, for each path (π, α), $I_{(\pi,\alpha),\varphi}(\vartheta) \neq \varnothing$.*

Proof. It suffices to show, by structural induction on φ, that for each $x \in vars(\theta)$, $I_{(\pi,\alpha),\phi}(\vartheta)\restriction_{\{x\}}$ is *cofinite* if $\vartheta(x) = \perp$, and the singleton $\{\vartheta(x)\}$ otherwise. The $\varphi_1 \wedge \varphi_2$ case comes from the fact that the intersection of two cofinite sets is cofinite. For $\varphi_1 \mathbf{U} \varphi_2$, it is enough to mention that: (i) the set of suffixes $\{(\pi, \alpha)^i \mid i \geq 0\}$ is finite when $\mathcal{K}$ is finite, and (ii) a finite intersection of cofinite sets is cofinite. The other cases are clear by definition and the induction hypothesis. □

3.3 Path-Realized Parameters for an LKS Satisfying FIP

For a *finite* LKS $\mathcal{K}$ satisfying FIP, we can determine the satisfaction of $\forall\overline{x}\ \varphi$ by considering a (possibly small) finite set of substitutions. Consider a set $\mathscr{R} \subseteq [\overline{x} \to \mathcal{C}_\perp]$ of substitutions with $\varrho_{(\pi,\alpha),\varphi}([\overline{x} \to \mathcal{C}]) \subseteq \mathscr{R}$. By definition, $\mathcal{K}, (\pi,\alpha) \models \forall\overline{x}\ \varphi$ iff $\mathcal{K}_\mathcal{C}, (\pi,\alpha) \models \theta\varphi$ for each $\theta \in [\overline{x} \to \mathcal{C}]$, and by Lemma 1, iff $\mathcal{K}_{\mathcal{C}_\perp}, (\pi,\alpha) \models \vartheta\varphi$ for each $\vartheta \in \varrho_{(\pi,\alpha),\varphi}([\overline{x} \to \mathcal{C}])$. If $\vartheta \in [\overline{x} \to \mathcal{C}_\perp] \setminus \varrho_{(\pi,\alpha),\varphi}([\overline{x} \to \mathcal{C}])$, by Lemma 2, there is a concrete substitution $\theta \in [\overline{x} \to \mathcal{C}]$ such that $\varrho_{(\pi,\alpha),\varphi}(\vartheta) = \varrho_{(\pi,\alpha),\varphi}(\theta)$, which implies $\mathcal{K}_{\mathcal{C}_\perp}, (\pi,\alpha) \models \vartheta\varphi$ iff $\mathcal{K}_\mathcal{C}, (\pi,\alpha) \models \theta\varphi$. Consequently, we have:

Theorem 1. *Given a* finite *LKS $\mathcal{K}$ satisfying FIP over parameters $\mathcal{C}$, a quantified SE-LTL formula $\forall\overline{x}\ \varphi$, and a path (π,α), if $\varrho_{(\pi,\alpha),\varphi}([\overline{x} \to \mathcal{C}]) \subseteq \mathscr{R} \subseteq [\overline{x} \to \mathcal{C}_\perp]$, then $\mathcal{K}, (\pi,\alpha) \models \forall\overline{x}\ \varphi$ iff for each $\vartheta \in \mathscr{R}$, $\mathcal{K}_{\mathcal{C}_\perp}, (\pi,\alpha) \models \vartheta\varphi$.*

If the operator $\oplus$ is extended to sets of substitutions by $I_1 \oplus I_2 = \{\theta_1 \oplus \theta_2 \mid \theta_1 \in I_1, \theta_2 \in I_2\}$, given a SE-LTL formula φ and a path (π,α), the *path-realized* set $\mathscr{R}_{(\pi,\alpha),\varphi} \subseteq [vars(\varphi) \to \mathcal{C}_\perp]$ of substitutions is defined as follows:

- $\mathscr{R}_{(\pi,\alpha),p(\overline{x})} = \mathcal{D}_{\pi(0),\alpha(0)}(p(\overline{x})) \cup \{\perp_{\overline{x}}\}$
- $\mathscr{R}_{(\pi,\alpha),\delta(\overline{x})} = \mathcal{D}_{\pi(0),\alpha(0)}(\delta(\overline{x})) \cup \{\perp_{\overline{x}}\}$
- $\mathscr{R}_{(\pi,\alpha),\neg\varphi} = \mathscr{R}_{(\pi,\alpha),\varphi}$
- $\mathscr{R}_{(\pi,\alpha),\varphi_1\wedge\varphi_2} = \mathscr{R}_{(\pi,\alpha),\varphi_1} \oplus \mathscr{R}_{(\pi,\alpha),\varphi_2}$
- $\mathscr{R}_{(\pi,\alpha),\bigcirc\varphi} = \mathscr{R}_{(\pi,\alpha)^1,\varphi}$
- $\mathscr{R}_{(\pi,\alpha),\varphi_1\mathbf{U}\varphi_2} = \bigcup_{i\geq 0} \mathscr{R}_{(\pi,\alpha)^i,\varphi_1} \ \oplus\ \bigcup_{j\geq 0} \mathscr{R}_{(\pi,\alpha)^j,\varphi_2}$

Since $\mathscr{R}_{(\pi,\alpha),\varphi}$ is the aggregation of all possible values of $\varrho_{(\pi,\alpha),\varphi}$, from Theorem 1, we have the following *localization lemma*:

Lemma 3. *Given a finite LKS $\mathcal{K}$ satisfying FIP over parameters $\mathcal{C}$, a quantified SE-LTL formula $\forall\overline{x}\ \varphi$, and a path (π,α), for each substitution $\theta \in [\overline{x} \to \mathcal{C}]$, there exists $\vartheta \in \mathscr{R}_{(\pi,\alpha),\varphi}$ such that $\mathcal{K}_\mathcal{C}, (\pi,\alpha) \models \theta\varphi$ iff $\mathcal{K}_{\mathcal{C}_\perp}, (\pi,\alpha) \models \vartheta\varphi$.*

If we consider a parameterized *fairness* formula $\forall\overline{x}\ \psi$,we can further reduce the set of substitutions necessary to determine the satisfaction of the formula. Since the satisfaction of a parameterized fairness formula ψ does not vary if we skip finitely many steps of a path, from the above lemma, we can consider only the set $\mathscr{R}^{inf}_{(\pi,\alpha),\psi}$ of *infinitely often* path-realized substitutions, whose elements belong to $\mathscr{R}_{(\pi,\alpha)^i,\psi}$ for infinitely many $i \in \mathbb{N}$. Note that $\mathscr{R}^{inf}_{(\pi,\alpha),\psi}$ is actually equal to $\mathscr{R}_{(\pi,\alpha)^N,\psi}$ for a sufficiently large $N \geq 0$ in which all substitutions with finite occurrences are skipped. Accordingly, by Theorem 1, we then have:

Theorem 2. *Given a* finite *LKS $\mathcal{K}$ satisfying FIP over parameters $\mathcal{C}$, a parameterized fairness formula $\forall\overline{x}\ \psi$, and a path (π,α), $\mathcal{K}, (\pi,\alpha) \models \forall\overline{x}\ \psi$ iff for each $\vartheta \in \mathscr{R}^{inf}_{(\pi,\alpha),\psi}$, $\mathcal{K}_{\mathcal{C}_\perp}, (\pi,\alpha) \models \vartheta\psi$.*

Note that if $\mathscr{R}^{inf}_{\psi} \subseteq [\overline{x} \to \mathcal{C}_\perp]$ is the union of $\mathscr{R}^{inf}_{(\pi,\alpha),\psi}$ for each (π,α) from a initial state of $\mathcal{K}$, by the above theorem, $\mathcal{K} \models \forall\overline{x}\ \psi$ iff for each $\vartheta \in \mathscr{R}^{inf}_{\psi}$, $\mathcal{K}_{\mathcal{C}_\perp} \models \vartheta\psi$.

4 Automata-Based Model Checking Algorithm

Given a finite LKS $\mathcal{K}$ satisfying FIP over parameters $\mathcal{C}$, the satisfiability of a quantified SE-LTL formula $\forall \overline{x}\ \varphi$ is now reduced to the satisfiability of $\vartheta\varphi$ on $\mathcal{K}_{\mathcal{C}_\perp}$ for each path-realized substitution ϑ. This reduction gives a simple algorithm to verify $\forall \overline{x}\ \varphi$ using the existing SE-LTL model checking algorithm as follows:

1. Traverse the state space of $\mathcal{K}$ to compute a path-realized set $\mathscr{R}_{(\pi,\alpha),\varphi}$ for each infinite path (π, α), witnessed by a cycle in the search graph.
2. For each substitution ϑ evaluated at Step 1, model check $\mathcal{K}_{\mathcal{C}_\perp} \models \vartheta\varphi$ using the existing algorithm. If all satisfied, then $\mathcal{K} \models \forall \overline{x}\ \varphi$. Otherwise, $\mathcal{K} \not\models \forall \overline{x}\ \varphi$.

This algorithm is not on-the-fly, since we have to traverse the entire state space first. However, for a parameterized fairness formulas, thanks to the fact that only infinitely often path-realized substitutions are necessary, we can give an on-the-fly model checking algorithm below based on a *strongly connected component* (SCC) analysis, seeing that each cycle is identified by a SCC.

4.1 Automata-Based Characterization

Given a set of parameterized strong/weak fairness formulas $\mathscr{F} \cup \mathscr{J}$ for a finite LKS $\mathcal{K}$ satisfying FIP over parameters $\mathcal{C}$, by Theorem 2, we can construct an equivalent set $\mathcal{G} = \hat{\mathscr{F}} \cup \hat{\mathscr{J}}$ of *propositional* generalized strong/weak fairness formulas by instantiating each parameterized formula $\forall \overline{x}\ \psi$ with the substitutions in $\mathscr{R}_\psi^{inf}$. Since generalized weak fairness formula $\Diamond\Box\Phi \to \Box\Diamond\Psi$ can be expressed by $\Box\Diamond\top \to \Box\Diamond(\neg\Phi \vee \Psi)$, we can regard $\mathcal{G}$ as a set of strong fairness formulas. Such strong fairness conditions can be incorporated into the acceptance conditions of a transition-based *Streett* automaton.

Definition 4. *A* Streett automaton $(Q, Q_0, P, \Delta, \mathcal{F})$ *is a 5-tuple with* Q *a set of states,* $Q_0 \subseteq Q$ *a set of initial states,* P *an alphabet of transition labels,* $\Delta \subseteq Q \times P \times Q$ *a transition relation, and* $\mathcal{F} \subseteq \mathcal{P}(\Delta \times \Delta)$ *an acceptance condition.*

A *run* of a Streett automaton $\mathcal{S}$ is an infinite sequence $q_0 \xrightarrow{l_0} q_1 \xrightarrow{l_1} q_2 \xrightarrow{l_2} \cdots$ of transitions starting from $q_0 \in Q_0$. A run σ is *accepted* by $\mathcal{S}$ iff for each pair $(G, H) \in \mathcal{F}$, whenever σ has transitions in G infinitely often, σ has transitions in H infinitely often. Given two Streett automata $\mathcal{S}_1$ and $\mathcal{S}_2$, their synchronous product $\mathcal{S}_1 \times \mathcal{S}_2$ is defined such that $|\mathcal{S}_1 \times \mathcal{S}_2| = O(|\mathcal{S}_1| \cdot |\mathcal{S}_2|)$ and $L(\mathcal{S}_1 \times \mathcal{S}_2) = L(\mathcal{S}_1) \cap L(\mathcal{S}_2)$ [11]. Note that a Büchi automaton $\mathcal{B} = (Q, Q_0, P, \Delta, F)$ can be translated into an equivalent Streett automaton $\mathcal{S}(\mathcal{B}) = (Q, Q_0, P, \Delta, \{(\Delta, F)\})$.

Given an LKS $\mathcal{K} = (S, S_0, AP, \mathcal{L}, ACT, T)$ and a set of generalized strong fairness formulas $\mathcal{G} = \{\Box\Diamond\Phi_i \to \Box\Diamond\Psi_i \mid i \in I\}$, we can define a fair Streett automaton $\mathcal{S}^{\mathcal{G}}(\mathcal{K}) = (S, S_0, \mathcal{P}(AP \uplus ACT), \Delta, \mathcal{F}^{\mathcal{G}})$ such that:[4]

$$\Delta = \{s \xrightarrow{\mathcal{L}(s) \uplus A} s' \mid s \xrightarrow{A} s' \in T\}$$
$$\mathcal{F}^{\mathcal{G}} = \{(\Delta^{\Phi_i}, \Delta^{\Psi_i}) \mid i \in I\}, \qquad \text{where } \Delta^{\Phi} = \{s \xrightarrow{B} s' \in \Delta \mid B \models \Phi\}$$

[4] $B \models \Phi$ is defined inductively as follows: $B \models p$ iff $p \in B$, $B \models \delta$ iff $\delta \in B$, $B \models \neg\Phi$ iff $B \not\models \Phi$, and $B \models \Phi_1 \wedge \Phi_2$ iff $B \models \Phi_1$ and $B \models \Phi_2$, where $p \in AP$ and $\delta \in ACT$.

Each path (π, α) of an LKS $\mathcal{K}$ is in one-to-one correspondence with a run $\pi(0) \xrightarrow{\mathcal{L}(\pi(0))\uplus\alpha(0)} \pi(1) \xrightarrow{\mathcal{L}(\pi(1))\uplus\alpha(1)} \cdots$ of the Streett automaton $\mathcal{S}^{\mathcal{G}}(\mathcal{K})$. Furthermore, (π, α) satisfies all fairness conditions of $\mathcal{G}$ iff the corresponding run of (π, α) is accepted by $\mathcal{S}^{\mathcal{G}}(\mathcal{K})$. Therefore, we can use a Streett automaton $\mathcal{S}^{\mathcal{G}}_{\neg\varphi}(\mathcal{K}) = \mathcal{S}^{\mathcal{G}}(\mathcal{K}) \times \mathcal{S}(\mathcal{B}_{\neg\varphi})$ to model check a SE-LTL formula in $\mathcal{K}$ under generalized strong/weak fairness conditions as follows:

Theorem 3. *Given an LKS $\mathcal{K}$, a SE-LTL formula φ, and a set of propositional generalized strong fairness formulas $\mathcal{G}$, there is a Streett automaton $\mathcal{S}^{\mathcal{G}}_{\neg\varphi}(\mathcal{K})$ with size $O(|\mathcal{K}| \cdot 2^{|\varphi|})$ such that $L(\mathcal{S}^{\mathcal{G}}_{\neg\varphi}(\mathcal{K})) = \varnothing$ iff $\mathcal{K} \models_{\mathcal{G}} \varphi$.*

Consequently, the model checking problem of SE-LTL formulas on a finite LKS $\mathcal{K}$ satisfying FIP under parameterized strong/weak fairness conditions $\mathscr{F} \cup \mathscr{J}$ is reduced to the emptiness checking problem of the Streett automaton whose acceptance condition is defined by the generalized strong/weak fairness conditions $\hat{\mathscr{F}} \cup \hat{\mathscr{J}}$ obtained by instantiating $\mathscr{F}$ and $\mathscr{J}$ for each $\vartheta \in \bigcup_{(\forall \overline{x}^k\ \psi)\in\mathscr{F}\cup\mathscr{J}} \mathscr{R}^{inf}_{\psi}$.

It is worth noting that a naive selection of such substitutions without parameter abstraction does not guarantee the equivalence between the parameterized fairness formula and the set of instantiated formulas. For example, in process algebra, the fairness formula $\forall x\, \Box\Diamond\neg enabled(x) \rightarrow \Box\Diamond execute(x)$ is always false, since for a process k not existing in a system, $\neg enabled(k)$ is true but $execute(k)$ is false. But all instantiated formulas only using the *existing* processes can be true if such processes are always enabled.

4.2 On-The-Fly Model Checking Algorithm

We present an *on-the-fly* automata-based algorithm for parameterized fairness, based on the emptiness checking algorithm for Streett automaton associated to the strong fairness conditions [11,19]. To check emptiness of a *finite* Streett automaton $(Q, Q_0, P, \Delta, \mathcal{F})$, the basic idea is to find a reachable SCC that satisfies all Streett acceptance conditions in $\mathcal{F}$ [12]. An acceptance condition $(g_i, h_i) \in \mathcal{F}$ is satisfied in a SCC $\mathfrak{S}$ iff whenever $\mathfrak{S}$ contains a transition $s_1 \xrightarrow{B} s_2$ such that $B \models g_i$, there exists some transition $s'_1 \xrightarrow{B'} s'_2 \in \mathfrak{S}$ such that $B' \models h_i$. If some (g_i, h_i) is not satisfied in $\mathfrak{S}$, then the *bad* transitions of $\mathfrak{S}$ are identified, satisfy $g_i \wedge \neg h_i$ and therefore prevent the satisfaction of (g_i, h_i).

The emptiness checking algorithm specified in Fig. 1 is to find a SCC with no bad transitions. The computeNextSCC(Q, q, Δ) function in Line 3 identifies each SCC $\mathfrak{S}$ in the graph (Q, Δ) containing q, which can be implemented by any *on-the-fly* algorithm to find a SCC; typically Tarjan's algorithm, or Couvreur's algorithm [9] for early finding of SCC. If $\mathfrak{S}$ satisfies all acceptance conditions (Line 4), then we can generate a counterexample given by a fair cycle from $\mathfrak{S}$ using breadth-first search [19]. Otherwise, if $\mathfrak{S}$ is a maximal strongly connected component (MSCC)[5] and contains bad transitions, then the whole $\mathfrak{S}$ is traversed again *except for* the bad transitions (Line 11), which leads to dividing $\mathfrak{S}$ into multiple smaller subcomponent with no such bad transitions.

[5] A MSCC is a SCC such that there is no other SCCs containing it.

findFairSCC(Q, Q_0, Δ)
2 **while** there is a reachable state $q \in Q$ from Q_0 that has not been visited **do**
3 $\mathfrak{S}$:= computeNextSCC(Q, q, Δ);
4 **if** fairnessSatisfied($\mathfrak{S}$) **then**
5 **return** $\mathfrak{S}$
6 **else if** $\mathfrak{S}$ is maximal and contains bad transitions **then**
7 $Q^{\mathfrak{S}}$:= the set of states in $\mathfrak{S}$;
8 $\Lambda^{\mathfrak{S}}$:= the set of bad transitions for unsatisfied acceptance conditions in $\mathfrak{S}$;
9 $Q_0^{\mathfrak{S}}$:= the set of states that occur in $\Lambda^{\mathfrak{S}}$;
10 mark each state in $Q^{\mathfrak{S}}$ as unvisited;
11 **return** findFairSCC($Q^{\mathfrak{S}}$, $\{q\} \cup Q_0^{\mathfrak{S}}$, $\Delta \setminus \Lambda^{\mathfrak{S}}$) **unless** $\bot$
12 **end if**
13 **end while**;
14 **return** $\bot$;

Fig. 1. Streett Emptiness Checking Algorithm for $\mathcal{S} = (Q, Q_0, P, \Delta, \mathcal{F})$

Given a Streett Automaton $\mathcal{S} = (Q, Q_0, P, \Delta, \mathcal{F})$, if there exists a nonempty SCC $\mathfrak{T} \in (Q, \Delta)$ satisfying $\mathcal{F}$ reachable from Q_0, then $\mathfrak{T}$ is a subcomponent of a MSCC $\mathfrak{S} = (Q^{\mathfrak{S}}, \Delta^{\mathfrak{S}})$ given by computeNextSCC(Q, q, Δ) with q reachable from Q_0. Further, since $\mathfrak{T}$ does not contain any bad transitions $\Lambda^{\mathfrak{S}}$ of $\mathfrak{S}$, we have $\mathfrak{T} \subseteq (Q^{\mathfrak{S}}, \Delta^{\mathfrak{S}} \setminus \Lambda^{\mathfrak{S}})$. Hence, whenever we meet findFairSCC($Q^{\mathfrak{S}}$, $\{q\} \cup Q_0^{\mathfrak{S}}$, $\Delta \setminus \Lambda^{\mathfrak{S}}$) in Line 11, $\mathfrak{T}$ is contained in $(Q^{\mathfrak{S}}, \Delta^{\mathfrak{S}} \setminus \Lambda^{\mathfrak{S}})$ and is reachable from $\{q\} \cup Q_0^{\mathfrak{S}}$. This yields the correctness of this algorithm as follows:

Theorem 4. *Assuming the correctness of an underlying SCC finding algorithm, given a Streett Automaton $\mathcal{S} = (Q, Q_0, P, \Delta, \mathcal{F})$, the findFairSCC($Q$, Q_0, Δ) finds a nonempty SCC satisfying $\mathcal{F}$ reachable from Q_0 if it exists.*

To make this algorithm on-the-fly under parameterized fairness conditions, we have to check fairnessSatisfied($\mathfrak{S}$) in Line 4 using only states and transitions in $\mathfrak{S}$. Given parameterized fairness formulas $\forall \overline{x}_i\ \psi_i$, $1 \leq i \leq n$, with ψ_i either $\Box\Diamond\Phi_i \rightarrow \Box\Diamond\Psi_i$, or $\Diamond\Box\Phi_i \rightarrow \Box\Diamond\Psi_i$, since Φ_i and Ψ_i have no temporal operators, when $\zeta(\overline{y})$ ranges over both state and event propositions, we have $\mathscr{R}_{(\pi,\alpha)^k,\Phi_i} = \bigoplus_{\zeta(\overline{y}) \in \Phi_i} (\mathcal{D}_{\pi(k),\alpha(k)}(\zeta(\overline{y})) \cup \{\bot_{\overline{y}}\})$, $k \geq 0$, and $\mathscr{R}_{(\pi,\alpha)^k,\Psi_i}$ similar. Thus, for any infinite path (π, α) whose infinite suffixes are included in $\mathfrak{S}$, the infinitely often path-realized set $\mathscr{R}^{inf}_{(\pi,\alpha),\psi_i}$ is a subset of the following set $\mathscr{R}_{\mathfrak{S},\psi_i}$:

$$\mathscr{R}_{\mathfrak{S},\psi_i} = \bigcup_{s \xrightarrow{A} s' \in \mathfrak{S}} \Big(\bigoplus_{\zeta(\overline{y}) \in \Phi_i} (\mathcal{D}_{s,A}(\zeta(\overline{y})) \cup \{\bot_{\overline{y}}\}) \ \oplus \bigoplus_{\zeta(\overline{z}) \in \Psi_i} (\mathcal{D}_{s,A}(\zeta(\overline{z})) \cup \{\bot_{\overline{z}}\}) \Big)$$

Thanks to the localization lemma, we only need to check fairness instances of ψ_i from $\mathscr{R}_{\mathfrak{S},\psi_i}$ to determine the satisfaction of $\forall \overline{x}_i\ \psi_i$ on such paths. That is, to decide whether acceptance conditions $\{(\Delta^{\theta\Phi_i}, \Delta^{\theta\Psi_i}) \mid \theta \in [\overline{x} \rightarrow \mathcal{C}]\}$ from $\forall \overline{x}_i\ \psi_i$ are all satisfied on $\mathfrak{S}$, it is enough to consider acceptance conditions $\{(\Delta^{\vartheta\Phi_i}, \Delta^{\vartheta\Psi_i}) \mid \vartheta \in \mathscr{R}_{\mathfrak{S},\psi_i}\}$, so that all parameterized acceptance conditions are *localized* to $\mathfrak{S}$ and fairnessSatisfied($\mathfrak{S}$) can be computed on-the-fly.

A Streett automaton emptiness check can be determined in time $O(|\mathcal{F}| \cdot (|Q| + |\Delta|))$ [11]. Thus, the time complexity of model checking a SE-LTL formula φ on $\mathcal{K}$ with parameterized fairness conditions is $O(f \cdot r \cdot |\mathcal{K}| \cdot 2^{|\varphi|})$, where f and r are, respectively, the numbers of parameterized fairness conditions and of infinitely often path-realized parameters in $\mathcal{K}$. That is, $f = |\mathscr{F} \cup \mathscr{J}|$, and $r = |R|$, where $R = \bigcup_{(\forall \overline{x}^k\ \psi) \in \mathscr{F} \cup \mathscr{J}} \mathscr{R}_{\psi}^{inf}$. Note that the space complexity is also exponential on $|\varphi|$, since in the worst case the whole state space can be a single SCC maintained by the underlying Streett emptiness checking algorithm.

5 Parameterized Fairness Case Studies

This section illustrates how our framework for parameterized fairness can be applied to a wide range of modeling applications, especially including nontrivial parameterized fairness assumptions such as (i) object fairness with dynamic object creation, and (ii) the fairness for the sliding window protocol, parametric not only on processes, but also on data in channels.

5.1 Evolving Dining Philosophers Problem

We illustrate *dynamic* parameterized fairness by means of the Evolving Dining Philosophers problem [17]. This problem is similar to the famous Dining Philosophers problem: there are N philosophers sitting at a circular table who are either *thinking*, *waiting*, or *eating*. A chopstick is placed in between each pair of adjacent philosophers. The thinking philosophers *wake* up to eat something. The waiting philosophers can *grab* a chopstick on their left or right, and eat when they have both. After eating, a philosopher places the chopsticks back on the table and *think*s. However, in the evolving version, a philosopher can join or leave the table, so that the number of philosophers can be dynamically changed. In this problem we cannot decide the total fairness conditions at the outset, since they apply to each philosopher.

Although there is no limit to the number of philosophers in the original problem, we give an unpredictable bound using the Collatz problem [10]. There is a global counter that symbolizes a philosophical problem, and philosophers keep thinking the problem by changing the number n to: (i) $3n + 1$ for n odd, or (ii) $n/2$ for n even. New philosophers can join the group if the global number is a multiple of the current number of philosophers. Only the last philosopher can leave the group. To keep consistency, whenever a philosopher joins or leaves the table, the related chopsticks should not be held by another philosopher.

Each philosopher is identified by a natural number $k \in \mathbb{N}$. The states of philosopher k are expressed by the parameterized state propositions $\mathit{thinking}(k)$, $\mathit{waiting}(k)$, and $\mathit{eating}(k)$. Similarly, the actions of the philosopher k are represented by the parameterized event propositions $\mathit{wake}(k)$, $\mathit{grab}(k)$, and $\mathit{think}(k)$. Initially, there are two philosophers 1 and 2 thinking. The corresponding parameterized LKS over parameters $\mathbb{N}$ can be generated from the initial state, for example, $\{\mathit{thinking}(1), \mathit{thinking}(2)\} \xrightarrow{\mathit{wake}(1)} \{\mathit{waiting}(1), \mathit{thinking}(2)\}$.

$\underline{\text{send}}_P$: { $a_P \leq i < s_P + l_P$ }
begin enqueue $I_P[i]$ to F_Q **end**

$\underline{\text{loss}}_P$: { $[i, w] \in F_P$ }
begin remove $[i, w]$ from F_P **end**

$\underline{\text{recv}}_P$: { $[i, w] := \text{dequeue}(F_P)$ }
begin
if $O_P[i] = \perp$ **then** $O_P[i] := \text{w}$;
$s_P := \min\{j \mid O_P[j] = \perp\}$;
$a_P := \max(a_P,\, i - l_Q + 1)$ **fi end**

Fig. 2. The Balanced Sliding-Window Protocol (For P)

In order to prove, e.g., the liveness property $\Diamond \mathit{eating}(1)$, we need the weak fairness condition of $\mathit{wake}(k)$ and the strong fairness condition of $\mathit{grab}(k)$ for each philosopher k, given by the following parameterized fairness formulas:

$$\forall x\ \Diamond\Box \mathit{enabled}.\mathit{wake}(x) \rightarrow \Box\Diamond \mathit{wake}(x) \qquad \forall x\ \Box\Diamond \mathit{enabled}.\mathit{grab}(x) \rightarrow \Box\Diamond \mathit{grab}(x)$$

The parameterized LKS generated over a set of parameters $\mathbb{N}$ is finite due to the Collatz bound, and satisfies FIP since the propositions in the fairness formulas can be true only for the existing philosophers. Hence, we can directly model check $\Diamond \mathit{eating}(1)$ under the above parameterized fairness conditions in our framework.

5.2 Balanced Sliding Window Protocol

In this example we show how a liveness property of a nontrivial system with an *unbounded* number of fairness assumptions can be verified under parameterized fairness. The balanced sliding window protocol is a symmetric protocol that allows information to be sent reliably in both directions. The verification task for this protocol is not simple, since the specification involves unbounded queue and dynamic fairness conditions.

The balanced sliding window protocol description is as follows [23]: there are two entirely symmetric processes P and Q connected to each other through a lossy channel. Packets exchanged by the processes are pairs $[i, w]$ with i an index number and w a data word. The acknowledgement is implicitly provided by sending and receiving messages. Process P contains an array I_P of packets to be sent, another array O_P of items to be received, and a FIFO queue F_P of packets in transit to be received. Process P also has three variables to describe a state of the process as follows: s_P the lowest index of packet not yet received from the other process, a_P the lowest index of packet sent but not yet acknowledged, and l_P a fixed bound allowing sending packets before being acknowledged.

Process P can *send* any packet $[i, w]$ in I_P to Q if no acknowledgement for it has been yet received but within bound, i.e., $a_P \leq i < s_P + l_P$. When *receiving* a packet, an already received packet is ignored. Otherwise, the packet is added to O_P, s_P is set to the smallest index that has not been received, and a_P is set to $max(a_P, i - l_Q + 1)$ to ignore messages in the future whose index is less then or equal to $i - l_Q$. Finally, the *loss* of a packet can happen at any time. The behavior of this protocol is summarized as the pseudo code in Fig. 2.

The liveness property we are interest in is that all messages are eventually delivered, given by the LTL formula $\Diamond \mathit{success}$ such that the state proposition

success holds if $I_P = O_Q$ and $I_Q = O_P$. Let the actions of a process p with packet $[i, w]$ be expressed by the parameterized event propositions $send(p, i, w)$ and $recv(p, i, w)$. The verification of $\Diamond success$ requires the weak fairness condition of $send(p, i, w)$ and the strong fairness condition of $recv(p, i, w)$ for each process p and packet $[i, w]$, specified by the following parameterized fairness formulas:

$$\forall(p, i, w)\ \ \Diamond\Box enabled.send(p, i, w) \rightarrow \Box\Diamond send(p, i, w)$$
$$\forall(p, i, w)\ \ \Box\Diamond enabled.recv(p, i, w)\ \rightarrow \Box\Diamond recv(p, i, w)$$

Since the state space of the original system is infinite due to the unbounded queue, we apply equational-abstraction [21] to collapse the set of states to a finite number by identifying repeated packets. At the level of the abstracted system all these fairness requirements are captured by the following *generalized* parameterized fairness conditions that only use state propositions, where the parameterized state proposition $inQueue(p, i, w)$ (resp., $inOutput(p, i, w)$) holds when the packet $[i, w]$ is in the queue F_p (resp., the output array O_p):

$$\begin{aligned}\forall(i, w)\ \ \Diamond\Box enabled.send(P, i, w)\ &\rightarrow \Box\Diamond inQueue(Q, i, w)\\ \forall(i, w)\ \ \Diamond\Box enabled.send(Q, i, w)\ &\rightarrow \Box\Diamond inQueue(P, i, w)\\ \forall(p, i, w)\ \ \Box\Diamond inQueue(p, i, w)\ \ \ &\rightarrow \Box\Diamond inOutput(p, i, w)\end{aligned}$$

Again, in the finite abstracted system, we can apply our framework to model check $\Diamond success$ under the above parameterized fairness conditions, since the system satisfies FIP, owing to the fact that each proposition in the fairness formula can be true only for the existing entities in the system.

6 Experimental Results

We have implemented our algorithm in the Maude system by extending the existing SE-LTL model checker [3]. Our tool accepts models with both unparameterized and parameterized fairness conditions. We have compared it with other explicit-state model checkers, such as Pat [22] and Spin [15], and then tested our algorithm on complex examples involving dynamic fairness conditions. Since Spin and Pat only support unparameterized fairness, the comparison with those tools uses a model with unparameterized fairness assumptions. The experiments in this section were conducted on an Intel Core 2 Duo 2.66 GhZ with 8GB RAM running Mac OS X 10.6. We set a timeout of 30 minutes for the experiments.

To evaluate our algorithm comparing it with other tools, we use the classical Dining Philosophers problem which requires both strong and weak fairness conditions to verify the liveness property $\Box\neg deadlock \rightarrow \Diamond eating(1)$, where *deadlock* is considered as an event proposition.[6] Table 1 shows the verification results for each tool, where "N" is the number of philosophers, and "Time" is the runtime in seconds. We can observe that in the weak-fairness case, our algorithm is comparable to Spin, and for the strong/weak fairness case, it shows similar performance with Pat. For Spin, we had to encode strong fairness conditions into the LTL formula since Spin only supports weak fairness.

[6] For the cases of Pat and Spin, we use a modified *deadlock-free* version.

Table 1. Dining Philosophers for the property $\Box\neg deadlock \rightarrow \Diamond eating(1)$

Fairness	N	Maude States	Maude Time	Pat States	Pat Time	Spin States	Spin Time
Weak Only (Counter Example)	6	913	< 0.1	1596	1.0	672	< 0.1
	7	2418	0.1	5718	5.1	2765	0.2
	8	11092	0.9	21148	33.5	9404	0.8
Strong/Weak (Valid)	6	5777	1.8	18101	3.9	> 30 minutes	
	7	24475	11.5	69426	16.1		
	8	103681	77.6	260998	79.0		

Table 2. Results for models with dynamic parameterized fairness

(a) Evolving Dining Philosophers

C. Nr.	States	Time	#Fairness
6	10532	3.6	10
18	86563	44.5	12
30	86387	47.5	12
42	13258	47.3	10
48	61751	31.1	12
54	697835	385.9	12

(b) Bounded Sliding Window Protocol

Size	Bound	States	Time	#Fairness
3	1	420	0.2	
3	2	1596	1.7	12
3	3	4095	5.7	
5	1	6900	5.5	
5	2	32256	42.6	20
5	3	123888	223.8	

Most interesting cases respecting parameterized fairness are models with dynamic fairness which cannot be easily predicted from the initial state, e.g., the examples in Sec. 5. Table 2a presents the model checking results for the evolving Dining Philosophers problem from the several initial Collatz numbers, where"#Fairness" is the total number of fairness instances generated during model checking. The results for the bounded sliding window protocol are provided in Table 2b, with different input array sizes and window bounds. In both cases, considerably large numbers of fairness constraints are automatically constructed, and verified within reasonable times.

7 Conclusions

We have presented a logical framework for parameterized fairness, which makes much easier expressing a wide range of fairness constraints that can be specified by universally quantified SE-LTL fairness formulas. We have also presented an on-the-fly algorithm for model checking SE-LTL properties under parameterized fairness, and have shown that it has reasonable performance when compared to other existing model checkers that support fairness. Furthermore, it answers the question of how to verify strong/weak fairness conditions for dynamic systems, in which the number of relevant parameter entities cannot be predicted. We have shown two case studies that require a dynamic, and unpredictable number of fairness conditions, which would be hard to handle by other tools.

Acknowledgments. This work has been partially supported by NSF Grants CNS 07-16638, CNS 08-34709, and CCF 09-05584. The authors would like to thank the referees for comments which helped to improve the paper.

References

1. Agha, G.: Actors: A Model of Concurrent Computation in Distributed Systems, Series in Artificial Intelligence, 11th edn. MIT Press, Cambridge (1986)
2. Bae, K., Meseguer, J.: The Maude LTLR model checker under parameterized fairness, manuscript (2011), `http://www.cs.uiuc.edu/homes/kbae4/fairness`
3. Bae, K., Meseguer, J.: The Linear Temporal Logic of Rewriting Maude Model Checker. In: Ölveczky, P.C. (ed.) WRLA 2010. LNCS, vol. 6381, pp. 208–225. Springer, Heidelberg (2010)
4. Bouajjani, A., Jonsson, B., Nilsson, M., Touili, T.: Regular model checking. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 403–418. Springer, Heidelberg (2000)
5. Chaki, S., Clarke, E.M., Ouaknine, J., Sharygina, N., Sinha, N.: State/Event-based software model checking. In: Boiten, E.A., Derrick, J., Smith, G.P. (eds.) IFM 2004. LNCS, vol. 2999, pp. 128–147. Springer, Heidelberg (2004)
6. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (2001)
7. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Bevilacqua, V., Talcott, C.: All About Maude - A High-Performance Logical Framework. LNCS, vol. 4350, pp. 31–37. Springer, Heidelberg (2007)
8. Cohen, A., Namjoshi, K.S., Sa'ar, Y.: A dash of fairness for compositional reasoning. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 543–557. Springer, Heidelberg (2010)
9. Couvreur, J., Duret-Lutz, A., Poitrenaud, D.: On-the-fly emptiness checks for generalized Büchi automata. Model Checking Software, 169–184 (2005)
10. Dams, D., Gerth, R., Grumberg, O.: Abstract interpretation of reactive systems. ACM Transactions on Programming Languages and Systems 19, 253–291 (1997)
11. Duret-Lutz, A., Poitrenaud, D., Couvreur, J.-M.: On-the-fly emptiness check of transition-based streett automata. In: Liu, Z., Ravn, A.P. (eds.) ATVA 2009. LNCS, vol. 5799, pp. 213–227. Springer, Heidelberg (2009)
12. Emerson, E.A., Lei, C.: Modalities for model checking: Branching time logic strikes back. Science of Computer Programming 8(3), 275–306 (1987)
13. Francez, N.: Fairness. Springer, Heidelberg (1986)
14. Henzinger, M., Telle, J.: Faster algorithms for the nonemptiness of Streett automata and for communication protocol pruning. In: Karlsson, R., Lingas, A. (eds.) SWAT 1996. LNCS, vol. 1097, Springer, Heidelberg (1996)
15. Holzmann, G.: The SPIN model checker: Primer and reference manual. Addison Wesley Publishing Company, Reading (2004)
16. Kesten, Y., Pnueli, A., Raviv, L., Shahar, E.: Model checking with strong fairness. Formal Methods in System Design 28(1), 57–84 (2006)
17. Kramer, J., Magee, J.: The evolving philosophers problem: Dynamic change management. IEEE Transactions on Software Engineering 16(11), 1293–1306 (2002)
18. Lamport, L.: Fairness and hyperfairness. Distributed Computing 13(4) (2000)
19. Latvala, T.: Model checking LTL properties of high-level petri nets with fairness constraints. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 242–262. Springer, Heidelberg (2001)

20. Meseguer, J.: Localized fairness: A rewriting semantics. In: RTA 2005. LNCS, vol. 3467, pp. 250–263. Springer, Heidelberg (2005)
21. Meseguer, J., Palomino, M., Martí-Oliet, N.: Equational abstractions. Theoretical Computer Science 403(2-3), 239–264 (2008)
22. Sun, J., Liu, Y., Dong, J., Pang, J.: PAT: Towards flexible verification under fairness. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 709–714. Springer, Heidelberg (2009)
23. Tel, G.: Introduction to distributed algorithms. Cambridge University Press, Cambridge (2000)
24. Vardi, M.Y.: Automata-theoretic model checking revisited. In: Cook, B., Podelski, A. (eds.) VMCAI 2007. LNCS, vol. 4349, pp. 137–150. Springer, Heidelberg (2007)