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 to verify LTL properties under parametric fairness, specified by *generalized* strong/weak fairness formulas. We also present an on-the-fly automata-based algorithm for model checking LTL formulas under universally quantified parameterized fairness assumptions. 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 a LTL property under fairness assumptions is to re-formulate the property such that fairness requirements become a conjunction of premises implying the original property [5]. Since this method is impractical for model checking properties under complex fairness assumptions, several specialized algorithms have been proposed, e.g., [11, 18, 15, 23].

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 [22], object/process fairness [12], actor fairness [1], etc. 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 or tool.

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 entities in the

system. We use a state/event-based version of LTL (SE-LTL) because fairness notions usually involve both states and events. For example, weak process fairness can be expressed by the universally quantified parameterized formula: $\forall x \in \mathit{Process}.\Diamond\Box \mathit{enabled}(x) \to \Box\Diamond \mathit{execute}(x)$, 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 conditions given by generalized strong (resp., weak) fairness formulas, specified by SE-LTL formulas of the form $\forall \overline{x}.\Box\Diamond\Phi \to \Box\Diamond\Psi$ (resp., $\forall \overline{x}.\Diamond\Box\Phi \to \Box\Diamond\Psi$). For parameterized fairness, the number of existing entities in the system over which the parametrization ranges can be unbounded[1] and may change during execution. For example, in process fairness with dynamic process creation, fairness is parametric over a number of processes unknown a priori. Thus, fairness of (infinitary) dynamic control [13] can be also easily expressed by our parameterized fairness with universal quantification.

Our framework is based on the observation that, if a state is fixed, only a *finite* number of entities or parameters are meaningful in such state for fairness purposes. For example, 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] A specific modeling language should be used to determine which are the *realized* parameters that are meaningful on given states of a system. We use *rewrite theories* [19] as a flexible modeling language, in which many concurrent systems, including object-based systems and process algebras, can be naturally described.

We have developed an on-the-fly model checking algorithm for SE-LTL properties that can directly handle universally quantified fairness formulas; its complexity is linear in the number of fairness conditions (see Sec. 4). We have implemented our algorithm within the Maude system, which is a verification framework for concurrent systems specified as rewrite theories. 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 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. In fact, fairness notions supported by usual modeling languages are parameterized, e.g., process fairness [12] 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. Similar

[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] For example, $\mathit{enabled}(p)$ is false for all states if a process p does not exist in the system. Therefore, $\Diamond\Box \mathit{enabled}(p) \to \Box\Diamond \mathit{execute}(p)$ is vacuously satisfied in the system.

to our work, localized fairness can be parameterized by *any* entities 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, 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 [10, 18] or a symbolic algorithm [15]. 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 [25]. There are several tools to support the specialized algorithms such as PAT [23] and Maria [18]. Our algorithm is related to the second approach to handle fairness, but it does not require pre-translation of parameterized fairness, and properly handles parametric generalized fairness. There are also other methods to support parameterized fairness not based on standard model checking methods, such as regular model checking [3] and compositional reasoning [7].

This paper is organized as follows. Section 2 presents the necessary background about fairness and the state/event-based logic. Section 3 introduces parameterized fairness and its properties, and Section 4 presents the automata-based model checking algorithm for parameterized fairness. Section 5 illustrates case studies of parameterized fairness with rewrite theory specifications. 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) [4] 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., $\Diamond\varphi \equiv \mathit{true}\mathbf{U}\varphi$ 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 [2, 4], 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 in our setting each transition of an LKS is labeled by a *set* A of atomic events, which enables us to describe a pattern of an event, not just an atomic 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 $i \geq 0$. A SE-LTL formula φ is satisfied by an LKS $\mathcal{K}$ with an initial state $s_0 \in S_0$, denoted by $\mathcal{K}, s_0 \models \varphi$, if and only if for each path (π, α) of $\mathcal{K}$ starting from s_0, the path satisfaction relation $\mathcal{K}, (\pi, \alpha) \models \varphi$ holds, which is defined inductively as follows:[3]

- $\mathcal{K}, (\pi, \alpha) \models p$ iff $p \in \mathcal{L}(s)$
- $\mathcal{K}, (\pi, \alpha) \models \delta$ 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 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 [17]. 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] $(\pi, \alpha)^i$ denotes the suffix of (π, α) beginning at position $i \in \mathbb{N}$, i.e., $(\pi, \alpha)^i = (\pi \circ s^i, \alpha \circ s^i)$ with s the successor function.

3 Parameterized Fairness as Quantified SE-LTL

Besides a temporal perspective about fairness, 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 in the system [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.

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 $AP_\mathcal{C} = \{p(\overline{a}^n) \mid \overline{a}^n \in \mathcal{C}^n, p \in \Pi, n \in \mathbb{N}\}$ and event propositions $ACT_\mathcal{C} = \{\delta(\overline{b}^m) \mid \overline{b}^m \in \mathcal{C}^m, \delta \in \Omega, m \in \mathbb{N}\}$.*

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 $\forall \overline{x}^k\ \varphi$, where φ is a propositional SE-LTL formula over state propositions $AP_{\mathcal{C}\cup\mathcal{V}} = \{p(\overline{a}^n) \mid \overline{a}^n \in (\mathcal{C} \cup \mathcal{V})^n, p \in \Pi, n \in \mathbb{N}\}$ and event propositions $ACT_{\mathcal{C}\cup\mathcal{V}} = \{\delta(\overline{b}^m) \mid \overline{b}^m \in (\mathcal{C} \cup \mathcal{V})^m, \delta \in \Omega, m \in \mathbb{N}\}$, with $\mathcal{V}$ an infinite set of variables disjoint from $\mathcal{C}$, and $\overline{x}^k = \mathit{vars}(\varphi)$ the set of variables, of size k, occurring in φ. The *satisfaction* of such formulas in a path (π, α) of a parameterized labeled Kripke structure $\mathcal{K} = (S, S_0, \Pi, \mathcal{L}, \Omega, T)$ is now defined in the obvious way:

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

where $\theta\varphi$ is the propositional SE-LTL formula obtained by applying the simultaneous substitution θ to the variables $\overline{x}^k$ in φ. Note that $\mathcal{K}, s_0 \models \forall \overline{x}^k\ \varphi$ iff $\mathcal{K}, (\pi, \alpha) \models \forall \overline{x}^k\ \varphi$ for each path (π, α) starting from the initial state $s_0 \in S$.

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 $[\overline{x}^k \to \mathcal{C}]$ denotes the set of all substitutions $\theta : \overline{x}^k \to \mathcal{C}$, given a path (π, α) of $\mathcal{K}$ and a quantified formula $\forall \overline{x}^k\ \varphi$, the *definable* set $\mathcal{D}_{(\pi,\alpha)}(\varphi)$ is the set of substitutions that make φ satisfied:

$$\mathcal{D}_{(\pi,\alpha)}(\varphi) = \{\theta \in [\mathit{vars}(\varphi) \to \mathcal{C}] \mid \mathcal{K}_\mathcal{C}, (\pi, \alpha) \models \theta\varphi\}.$$

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

Definition 3. *A parameterized LKS* $\mathcal{K} = (S, S_0, \Pi, \mathcal{L}, \Omega, T)$ *over a parameter set* $\mathcal{C}$ *is* tractable *if for each path* (π, α) *of* $\mathcal{K}$*, the sets* $\mathcal{D}_{(\pi,\alpha)}(p(\overline{x}^n))$ *for each* $p \in \Pi$*,* $n \in \mathbb{N}$*, and* $\mathcal{D}_{(\pi,\alpha)}(\delta(\overline{x}^m))$ *for each* $\delta \in \Omega$*,* $m \in \mathbb{N}$*, are finite.*

A parameterized strong (resp., weak) fairness formula from Φ to Ψ is a universally quantified SE-LTL formula $\forall \overline{x}^k\ \Box\Diamond\Phi \to \Box\Diamond\Psi$ (resp, $\forall \overline{x}^k\ \Diamond\Box\Phi \to \Box\Diamond\Psi$), where Φ and Ψ do not contain any temporal operators. The intuitive meaning of a parameterized strong/weak fairness formula is that for each group of entities $(a_1, \ldots, a_k)$ of a system, if certain actions or conditions are infinitely often provided (Φ), then they are infinitely often taken (Ψ). In reality, such entities in each state are finite, so that the system is often tractable. Consider sets of strong (resp., weak) parameterized fairness formulas $\mathscr{F} = \{\forall \overline{x_i}^{k_i}\ \Box\Diamond\Phi_i \to \Box\Diamond\Psi_i \mid i \in I\}$ (resp., $\mathscr{J} = \{\forall \overline{x_j}^{k_j}\ \Diamond\Box\Phi_j \to \Box\Diamond\Psi_j \mid j \in J\}$), where I and J are index sets. A path (π, α) of $\mathcal{K}$ is *fair under* parameterized fairness conditions $\mathscr{F} \cup \mathscr{J}$ iff $\mathcal{K}, (\pi, \alpha) \models \forall \overline{x_i}^{k_i}.\Box\Diamond\Phi_i \to \Box\Diamond\Psi_i$ for each $i \in I$ and $\mathcal{K}, (\pi, \alpha) \models \forall \overline{x_j}^{k_j}.\Box\Diamond\Phi_j \to \Box\Diamond\Psi_j$ for each $j \in J$. 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 $\mathcal{K}, (\pi, \alpha) \models \varphi$ holds for each fair path (π, α) under $\mathscr{F} \cup \mathscr{J}$ starting from any initial state of $\mathcal{K}$.

3.1 Parameter Abstraction

For a tractable LKS $\mathcal{K}$ over a parameter set $\mathcal{C}$ we can define abstraction of substitutions $\theta : \overline{x}^k \to \mathcal{C}$ with respect to definable sets. The key idea is to collapse the cofinite[4] complement set of each proposition-definable set into the abstracted substitution $\bot_{\overline{x}^n} : \overline{x}^n \to \{\bot\}$ with a fresh new constant $\bot$, which intuitively denotes a parameter that does *never* appear in the finite definable set. For example, for a state proposition $p(\overline{x}^n)$, each substitution $\theta \notin \mathcal{D}_{(\pi,\alpha)}(p(\overline{x}^n))$ is abstracted to $\bot_{\overline{x}^n} : \overline{x}^n \to \{\bot\}$. 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 $\mathcal{K}_{\mathcal{C}_\bot}, (\pi, \alpha) \not\models \bot_{\overline{x}^n} p(\overline{x}^n)$ holds, as $\bot_{\overline{x}^n} p(\overline{x}^n) \in \mathcal{L}(\pi(0))$ is impossible.

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

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

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:

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

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 $\theta_1 \oplus \theta_2$:

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

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

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

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

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 a tractable LKS is preserved by the abstraction function $\varrho_{(\pi,\alpha),\varphi}$ of substitutions.

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

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 $\mathcal{K}_{\mathcal{C}_\bot}, (\pi, \alpha) \models \vartheta\varphi$ for *any path* (π, α) and any substitution $\vartheta \in [\overline{x}^k \to \mathcal{C}_\bot]$ such that $\varrho_{(\pi,\alpha),\varphi}(\theta) \preceq \vartheta \preceq \theta$. For a state proposition $p(\overline{x}^n)$, $\mathcal{K}_\mathcal{C}, (\pi, \alpha) \not\models \theta p(\overline{x}^n)$ iff $\theta \notin \mathcal{D}_{(\pi,\alpha)}(p(\overline{x}^n))$ iff $\varrho_{(\pi,\alpha),p(\overline{x}^n)}(\theta) = \bot_{\overline{x}^n}$, and for each substitution $\bot_{\overline{x}^n} \preceq \vartheta \preceq \theta$, $\mathcal{K}_{\mathcal{C}_\bot}, (\pi, \alpha) \not\models \vartheta\varphi$. To prove the $\varphi_1\mathbf{U}\varphi_2$ case, we need the following 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 $i \geq 0$. As a result, for each $\varrho_{(\pi,\alpha),\varphi_1\mathbf{U}\varphi_2}(\theta) \preceq \vartheta \preceq \theta$, if $V_1 = vars(\varphi_1)$ and $V_2 = vars(\varphi_2)$, we have $\varrho_{(\pi,\alpha)^i,\varphi_1}(\theta{\restriction}_{V_1}) \preceq \vartheta{\restriction}_{V_1} \preceq \theta{\restriction}_{V_1}$ and $\varrho_{(\pi,\alpha)^i,\varphi_2}(\theta{\restriction}_{V_2}) \preceq \vartheta{\restriction}_{V_2} \preceq \theta{\restriction}_{V_2}$, $i \geq 0$. Hence, by induction hypothesis, we have $\mathcal{K}_\mathcal{C}, (\pi, \alpha)^i \models \theta\varphi_1$ iff $\mathcal{K}_{\mathcal{C}_\bot}, (\pi, \alpha)^i \models \vartheta\varphi_1$, and $\mathcal{K}_\mathcal{C}, (\pi, \alpha)^j \models \theta\varphi_2$ iff $\mathcal{K}_{\mathcal{C}_\bot}, (\pi, \alpha)^j \models \vartheta\varphi_2$ for each $i, j \geq 0$. Thus, $\mathcal{K}_\mathcal{C}, (\pi, \alpha) \models \theta(\varphi_1\mathbf{U}\varphi_2)$ iff $\mathcal{K}_{\mathcal{C}_\bot}, (\pi, \alpha) \models \vartheta(\varphi_1\mathbf{U}\varphi_2)$. The other cases are similar. □

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

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

It is easy to check that for each $\theta \in I_{(\pi,\alpha),\varphi}(\vartheta)$, $\vartheta \preceq \theta$ and $\varrho_{(\pi,\alpha),\varphi}(\vartheta) = \varrho_{(\pi,\alpha),\varphi}(\theta)$. There may be *no* concretization for some abstracted substitution,[5]

[5] Consider a path (π, α) such that $\mathcal{D}_{(\pi,\alpha)^i}(p(x)) = \{i\}$ for each $i \geq 0$, where $\mathcal{C} = \mathbb{N}$. Then, $\varrho_{(\pi,\alpha),\Diamond p(x)}(\theta) = \theta$ for a substitution $\theta \in [\{x\} \to \mathbb{N}]$, and $I_{(\pi,\alpha),\Diamond p(x)}(\bot_x) = \varnothing$.

whereas the abstraction of a concrete substitution does always exist. 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 *tractable LKS $\mathcal{K}$ over a parameter set $\mathcal{C}$, a quantified SE-LTL formula $\forall \overline{x}^k\ \varphi$, and $\vartheta \in [\overline{x}^k \to \mathcal{C}_\bot]$, 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 $x \in \mathit{vars}(\theta)$, $I_{(\pi,\alpha),\phi}(\vartheta)\restriction_{\{x\}}$ is *cofinite* if $\vartheta(x) = \bot$, 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 a formula $\varphi_1 \mathbf{U} \varphi_2$, 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 obvious by definition and the induction hypothesis. □

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

Theorem 1. *Given a* finite *tractable LKS $\mathcal{K}$ over a parameter set $\mathcal{C}$ and a quantified SE-LTL formula $\forall \overline{x}^k\ \varphi$, for each path (π, α) and a set of substitutions $\mathscr{R} \subseteq [\overline{x}^k \to \mathcal{C}_\bot]$ such that $\mathscr{R} \supseteq \varrho_{(\pi,\alpha),\varphi}([\overline{x}^k \to \mathcal{C}])$,*

$$\mathcal{K}, (\pi, \alpha) \models \forall \overline{x}^k\ \varphi \quad \Leftrightarrow \quad \forall (\theta \in \mathscr{R}).\ \mathcal{K}_{\mathcal{C}_\bot}, (\pi, \alpha) \models \theta\varphi.$$

3.2 Parameter Abstraction for Parameterized Fairness

If we consider a parameterized fairness formula $\forall \overline{x}^k\ \psi$ from Φ to Ψ, with ψ either $\Box\Diamond\Phi \to \Box\Diamond\Psi$, or $\Diamond\Box\Phi \to \Box\Diamond\Psi$, a set $\mathscr{R}$ with $\varrho_{(\pi,\alpha),\psi}([\overline{x}^k \to \mathcal{C}]) \subseteq \mathscr{R} \subseteq [\overline{x}^k \to \mathcal{C}_\bot]$ can be computed easily, since the abstraction with respect to Φ and Ψ, not having temporal operators, depends only on the *first step* of (π, α), namely, a pair $\langle \pi(0), \alpha(0) \rangle$. For that reason, we can have the alternative definitions of definable sets for propositions:

$$\begin{aligned} \mathcal{D}_{\langle s,A \rangle}(p(\overline{x}^n)) &= \{\theta \in [\overline{x}^n \to \mathcal{C}] \mid \theta p(\overline{x}^n) \in \mathcal{L}(s)\} \\ \mathcal{D}_{\langle s,A \rangle}(\delta(\overline{x}^m)) &= \{\theta \in [\overline{x}^m \to \mathcal{C}] \mid \theta\delta(\overline{x}^m) \in A\} \end{aligned}$$

Let $\mathfrak{p}(s, A)$ denote the set of all paths (π, α) such that $\pi(0) = s$ and $\alpha(0) = A$. For any path $(\pi, \alpha) \in \mathfrak{p}(s, A)$, $\mathcal{D}_{\langle s,A \rangle}(p(\overline{x}^n))$ (resp., $\mathcal{D}_{\langle s,A \rangle}(\delta(\overline{x}^m))$) is equivalent to $\mathcal{D}_{(\pi,\alpha)}(p(\overline{x}^n))$ (resp., $\mathcal{D}_{(\pi,\alpha)}(\delta(\overline{x}^m))$). 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 boolean formula Φ, the *step-realized* set $\mathscr{R}_{\langle s,A \rangle}(\Phi) \subseteq [\mathit{vars}(\Phi) \to \mathcal{C}]$ is defined as follows:

- $\mathscr{R}_{\langle s,A \rangle}(p(\overline{x}^n)) = \mathcal{D}_{\langle s,A \rangle}(p(\overline{x}^n)) \cup \{\bot^{\overline{x}^n}\}$

- $\mathscr{R}_{\langle s,A\rangle}(\delta(\overline{x}^m)) = \mathcal{D}_{\langle s,A\rangle}(\delta(\overline{x}^m)) \cup \{\bot^{\overline{x}^m}\}$
- $\mathscr{R}_{\langle s,A\rangle}(\neg\varphi) = \mathscr{R}_{\langle s,A\rangle}(\varphi)$
- $\mathscr{R}_{\langle s,A\rangle}(\varphi_1 \wedge \varphi_2) = \mathscr{R}_{\langle s,A\rangle}(\varphi_1) \oplus \mathscr{R}_{\langle s,A\rangle}(\varphi_2)$

A step-realized set $\mathscr{R}_{\langle s,A\rangle}(\Phi)$ *covers* any (abstracted) substitution $\theta \in [vars(\Phi) \to \mathcal{C}_\bot]$, in the sense that $\varrho_{(\pi,\alpha),\Phi}(\theta) \in \mathscr{R}_{\langle s,A\rangle}(\Phi)$ for each path $(\pi,\alpha) \in \mathfrak{p}(s,A)$.

A *path-realized* set $\mathscr{R}_{(\pi,\alpha),\psi}$ of a parameterized fairness formula ψ from Φ to Ψ is then defined by $\bigcup_{i\geq 0}\mathscr{R}_{(\pi,\alpha),\psi}(i)$, where $\mathscr{R}_{(\pi,\alpha),\psi}(i) = \mathscr{R}_{\langle\pi(i),\alpha(i)\rangle}(\Phi) \oplus \mathscr{R}_{\langle\pi(i),\alpha(i)\rangle}(\Psi)$. The inclusion relation $\varrho_{(\pi,\alpha),\psi}([\overline{x}^k \to \mathcal{C}]) \subseteq \mathscr{R}_{(\pi,\alpha),\psi}$ can be shown because it is actually the aggregation of all possible values of $\varrho_{(\pi,\alpha),\psi}$. Therefore, $\varrho_{(\pi,\alpha),\psi}(\theta) \in \mathscr{R}_{(\pi,\alpha)}(\psi)$ for any substitution $\theta \in [vars(\Phi) \to \mathcal{C}_\bot]$, and from Lemma 1, we have the following *localization lemma*:

Lemma 3. *Given a tractable LKS $\mathcal{K}$ over a parameter set $\mathcal{C}$ and a parameterized fairness formula $\forall\overline{x}^k\ \psi$, for each path (π,α) and substitution $\theta \in [\overline{x}^k \to \mathcal{C}]$, there exists $\vartheta \in \mathscr{R}_{(\pi,\alpha),\psi}$ such that $\mathcal{K}_\mathcal{C}, (\pi,\alpha) \models \theta\varphi$ iff $\mathcal{K}_{\mathcal{C}_\bot}, (\pi,\alpha) \models \vartheta\varphi$.*

Also, since the satisfaction of a parameterized fairness formula ψ does not vary if we skip finitely many steps, 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 $\mathscr{R}_{(\pi,\alpha),\psi}(i)$ 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. Let $\mathscr{R}^{inf}_\psi \subseteq [\overline{x}^k \to \mathcal{C}]$ be the union of $\mathscr{R}^{inf}_{(\pi,\alpha),\psi}$ for each (π,α) from a initial state of $\mathcal{K}$. Accordingly, by Theorem 1, we then have:

Proposition 1. *Given a* finite *tractable LKS $\mathcal{K}$ over a parameter set $\mathcal{C}$, a parameterized fairness formula $\forall\overline{x}^k\ \psi$, and an initial state s_0 of $\mathcal{K}$, there is a finite set $\mathscr{R}^{inf}_\psi$ such that $\mathcal{K}, s_0 \models \forall\overline{x}^k\ \psi$ iff for each $\vartheta \in \mathscr{R}^{inf}_\psi$, $\mathcal{K}_{\mathcal{C}_\bot}, s_0 \models \vartheta\psi$.*

Therefore, given sets of parameterized strong/weak fairness formulas $\mathscr{F}$ and $\mathscr{J}$, we can construct an equivalent set of generalized strong/weak fairness formulas $\hat{\mathscr{F}}$ and $\hat{\mathscr{J}}$ for each of the infinitely often path-realized parameters of $\mathcal{K}$.

4 Automata-based Model Checking Algorithm

In the previous section we have shown that parameterized fairness can be reduced to the equivalent finite conjunction of path-realized fairness instances. This section presents an automata-based characterization of parameterized strong/weak fairness in a state/event-based logic, and an on-the-fly model checking algorithm. In the automata-theoretic approach, strong fairness conditions can be incorporated into the acceptance conditions of a *Streett* automaton.

Definition 4. *A* transition-based Streett automaton $(Q, Q_0, P, \Delta, \mathcal{F})$ *is a 5-tuple with Q a finite 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 $L(\mathcal{S}_1 \times \mathcal{S}_2) = L(\mathcal{S}_1) \cap L(\mathcal{S}_2)$ [10]. 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[6] $\mathcal{G} = \{\Box\Diamond\Phi_i \to \Box\Diamond\Psi_i \mid i \in I\}$, 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:[7]

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

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 2. *Given an LKS $\mathcal{K}$, a SE-LTL formula φ, and a set of 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 for each initial state s_0, $\mathcal{K}, s_0 \models_{\mathcal{G}} \varphi$*

Consequently, the model checking problem of SE-LTL formulas on a finite tractable LKS $\mathcal{K}$ 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 $\hat{\mathscr{F}}$ and $\hat{\mathscr{J}}$ obtained by instantiating $\mathscr{F}$ and $\mathscr{J}$ for each $\theta \in \bigcup_{(\forall \overline{x}^k\ \psi) \in \mathscr{F} \cup \mathscr{J}} \mathscr{R}^{inf}_{\psi}$.

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 [10, 18]. The basic idea to check emptiness of a Streett automaton is to find a reachable *strongly connected component* (SCC) that satisfies all Streett acceptance conditions [11]. An acceptance condition (g_i, h_i) is satisfied in a SCC $\mathfrak{S}$ iff there exists a transition $s_1 \xrightarrow{B} s_2$ in $\mathfrak{S}$ such that $B \models g_i$ implies the existence of some transition $s'_1 \xrightarrow{B'} s'_2 \in \mathfrak{S}$ such that $B' \models h_i$. Given acceptance conditions $(g_1, h_1), (g_2, h_2), \ldots, (g_k, h_k)$, the emptiness checking algorithm is as follows:

1. Identify each SCC $\mathfrak{S}$ of the automaton, typically using Tarjan's algorithm, or Couvreur's algorithm [8] for early finding of SCC.
2. If $\mathfrak{S}$ satisfies all Streett acceptance conditions, then we can generate a fair cycle from $\mathfrak{S}$, e.g., by breadth-first search [18], and return a counterexample.

[6] Generalized weak fairness $\Diamond\Box\Phi \to \Box\Diamond\Psi$ can be expressed by $\Box\Diamond\top \to \Box\Diamond(\neg\Phi \vee \Psi)$.

[7] $B \models \Phi$ 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 $\delta \in ACT$.

3. If $\mathfrak{S}$ is a maximal strongly connected component (MSCC)[8] but some (g_i, h_i) is not satisfied in $\mathfrak{S}$, then the *bad* transitions of $\mathfrak{S}$ are identified, which satisfy $g_i \wedge \neg h_i$ and therefore prevent the satisfaction of (g_i, h_i).
4. If there are no bad transitions in $\mathfrak{S}$, then go back to Step 1 to check the next SCC, since some acceptance condition cannot be satisfied on $\mathfrak{S}$. Return *true* if no SCC remains.
5. Otherwise, traverse $\mathfrak{S}$ again *except for* bad transitions, which leads to dividing $\mathfrak{S}$ into multiple smaller subcomponent with no bad transitions. Go back to Step 1 to check each subcomponent recursively one by one.

However, the above algorithm cannot model check parameterized fairness on-the-fly, since we have to traverse the entire state space first to determine all instantiated fairness conditions for parameterized fairness.

The key to our on-the-fly algorithm for parameterized fairness conditions $\forall \overline{x}^{k_1}\ \psi_1, \ldots, \forall \overline{x}^{k_n}\ \psi_n$ is to keep track of path-realized substitutions for each ψ_i, $1 \leq i \leq n$, in each SCC $\mathfrak{S}$. Let $\mathscr{R}_{\mathfrak{S},\psi_i}$ denote the union of all step-realized sets of ψ_i for each transition in $\mathfrak{S}$. Since $\mathfrak{S}$ involves all paths of an automaton whose infinite suffixes are included in $\mathfrak{S}$, each infinitely often path-realized substitution of such paths is contained in $\mathscr{R}_{\mathfrak{S},\psi_i}$. Thanks to the localization lemma, we only need to check fairness instances of ψ_i from $\mathscr{R}_{\mathfrak{S},\psi_i}$ in order to determine the satisfaction of $\forall \overline{x}^{k_i}\ \psi_i$ on the paths, so that fairness instances are localized to $\mathfrak{S}$. That is, to decide if parameterized fairness conditions $\forall \overline{x}^{k_1}\ \psi_1, \ldots, \forall \overline{x}^{k_n}\ \psi_n$ are satisfied on $\mathfrak{S}$, it is enough to consider the set of fairness instances $\bigcup_{1 \leq i \leq n} \{\theta\psi_i \mid \theta \in \mathscr{R}_{\mathfrak{S},\psi_i}\}$, which can be easily computed on-the-fly from the given SCC $\mathfrak{S}$.

A Streett automaton emptiness check can be determined in time $O(|\mathcal{F}| \cdot (|Q| + |\Delta|))$ [10]. 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 $R = \bigcup_{(\forall \overline{x}^k\ \psi) \in \mathscr{F} \cup \mathscr{J}} \mathscr{R}^{inf}_{\psi}$.

5 Parameterized Fairness Case Studies

This section shows how our framework for parameterized fairness can be applied to a wide range of modeling applications, especially including nontrivial parameterized fairness assumptions, using rewrite theories. A *rewrite theory* is a formal specification of a concurrent system [19], by which many concurrent systems such as actors, process algebras, Petri nets, and the semantics of concurrent programming languages can be naturally described.

A rewrite theory is a triple $\mathcal{R} = (\Sigma, E, R)$ such that: (i) (Σ, E) is a theory in *membership equational logic* [6] with Σ a signature[9] and E a set of *conditional* equations and memberships, and (ii) R is a set of (possibly conditional) *rewrite rules* specifying the system's concurrent transitions between states, written l :

[8] A MSCC is a SCC such that there is no other SCCs containing it.
[9] i.e., Σ is a set of declarations of *sorts*, *subsorts*, *constants*, and *function symbols*.

$q \to r$, where l is a *label*, and q and r are Σ-terms. The set of concurrent states is specified as an algebraic data type $T_{\Sigma/E}$, i.e., each state is an E-equivalence class $[t]_E$ of ground terms. Each rule $l : q \to r$ specifies a *one-step transition* from each state $t[\theta q]$ containing a substitution instance θq of q to the state $t[\theta r]$ where θq has been replaced by the corresponding instance θr, denoted by $t[l(\theta)] : [t[\theta q]]_E \to_{\mathcal{R}} [t[\theta r]]_E$, where $t[l(\theta)]$ is called a *one-step proof term*.

Given a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ with an initial state $[t]_E$, the associated (tractable) LKS can be generated to model check SE-LTL properties [2]. State propositions, possibly parametrized, should be declared as operators of sort `Prop`, and their semantics should be given by (possibly conditional) equations of the form $t_{\texttt{State}}(\overline{x}^k) \texttt{ |= } p(\overline{x}^n) = \texttt{true}$, which defines that in each state $\theta t_{\texttt{State}}(\overline{x}^k)$ instantiated from $t_{\texttt{State}}(\overline{x}^k)$ with a substitution θ, the instance proposition $\theta p(\overline{x}^n)$ holds iff $\theta(t_{\texttt{State}}(\overline{x}^k) \texttt{ |= } p(\overline{x}^n))$ evaluates to `true`. Event propositions, also possibly parameterized, can be declared as operators of sort `Action` by the similar setting, using one-step proof terms as actions. For each equation $t_{\texttt{State}}(\overline{x}^k) \texttt{ |= } p(\overline{x}^n) = \texttt{true}$ of a state proposition $p(\overline{x}^n)$, if $\overline{x}^n \subseteq \overline{x}^k$, then the equation can have only a *finite* number of matching instances with respect to a fixed ground state term, and therefore $p(\overline{x}^n)$ becomes tractable. Similarly, a parameterized event proposition $e(\overline{x}^m)$ also gets to be tractable if $\overline{x}^m \subseteq \overline{x}^k$ for each corresponding equation of the form $t_{\texttt{ProofTerm}}(\overline{x}^k) \texttt{ |= } e(\overline{x}^m) = \texttt{true}$.

Evolving Dining Philosophers Problem. We illustrate a rewrite theory specification with *dynamic* parameterized fairness by means of the Evolving Dining Philosophers problem [16]. This problem is similar to the famous Dining Philosophers problem, but 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 depend on each philosopher. Although there is no limit to the number of philosophers in the original problem, we can give an unpredictable bound using the Collatz problem [9]. 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 only if the global number is a multiple of the current number of philosophers. We assume that 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 represented by a term `ph(I, S, C)`, where `I` is the philosopher's id, `S` is a status, and `C` is the number of chopsticks held. Similarly, a chopstick is represented by a term `stk(I)`. The state is a triple `< P, N, CF >` with `P` a global counter, `N` the number of philosophers, and `CF` a *set* of philosophers and chopsticks. The behavior of philosophers is then described by rewrite rules in the Maude language, specifically:

```
 rl [wake]: ph(I, think, 0) => ph(I, hungry, 0) .
crl [grab]: < P, N, ph(I, hungry, C) ; stk(J) ; CF >
     => < P, N, ph(I, hungry, C + 1) ; CF >  if J == left(I) or J == right(I, N) .
```

```
crl [solve]: <P, N, ph(I, think, 0) ; CF> => <Q, N, ph(I, think, 0) ; CF>
     if P > 1 /\ Q := collatz(P) .
crl [leave]: <P, N, CF ; ph(N, think, 0) ; stk(N)> => <P, N - 1, CF>  if N > 2 .
```

A philosopher is eating whenever it holds 2 chopsticks, which is expressed by the state proposition `eating(I)`. To prove the liveness property $\diamond$`eating`(1), we need weak fairness for the `wake` rule and strong fairness for the `grab` rule for each philosopher, given by the following parameterized fairness formulas: $\forall$`I`. $\diamond\Box$`enabled(wake(I))` $\rightarrow \Box\diamond$`wake(I)` and $\forall$`I`. $\Box\diamond$`enabled(grab(I))` $\rightarrow \Box\diamond$`grab(I)`.

We can model check $\diamond$`eating`(1) under the above parametrized fairness conditions in our framework, since the state-space is finite and the propositions in the above fairness formulas are tractable. That is, a variable `I` of each proposition is included in either the state term or the one-step proof term for each corresponding satisfaction equation, in particular:

```
eq < P, N, ph(I,think,0) ; CF > |= enabled(wake(I)) = true .
eq {'wake : 'I \ N ; SUBST} |= wake(N) = true .
```

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 simply 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 the balanced sliding window protocol is not simple, since the specification involves unbounded queue data structures and dynamic fairness conditions.

The balanced sliding window protocol description is as follows [24]: 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. 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 four 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, l_p a fixed bound allowing sending packets before being acknowledged, l_q a bound of the other process. A state of Process p is defined by a tuple $[p : s_p, a_p, l_p, l_q, I_p, O_p, F_p]$. Then, the behaviors of this protocol are specified by the following rewriting rules:

```
crl [snd]: [P : SP, AP, LP, LQ, IP, OP, FP] & [Q : SQ, AQ, LQ, LP, IQ, OQ, FQ]
        => [P : SP, AP, LP, LQ, IP, OP, FP] & [Q : SQ, AQ, LQ, LP, IQ, OQ, FQ [J, W]]
    if SP + LP > AP  /\  ([J, W] ; MAP) := IP < AP : SP + LP - 1 > .
rl [rec]: [P : S, A, LP, LQ, I, O, [J, W] FP]
    => if $hasMapping(O, J) then [P : S, A, LP, LQ, I, O, FP]
     else [P : 1st-undef(O;[J, W]), max(A, J - LQ + 1), LP, LQ, I, O;[J, W], FP] fi .
rl [los]: [P : S, A, LP, LQ, I, O, L [J, W] G] => [P : S, A, LP, LQ, I, O, L G] .
```

The liveness property we are interest in is that all messages are eventually delivered, given by the LTL formula $\diamond$*success*. Since the number of states are

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. However, we need weak fairness conditions for the sending of a packet for each process, and strong fairness conditions for the receiving of each packet by the process. 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: $\forall(\mathtt{P},\mathtt{I},\mathtt{W}).\Box\Diamond\mathtt{in\text{-}fifo}(\mathtt{P},\mathtt{I},\mathtt{W}) \rightarrow \Box\Diamond\mathtt{in\text{-}rec}(\mathtt{P},\mathtt{I},\mathtt{W})$ and $\forall(\mathtt{P},\mathtt{I},\mathtt{W}).\Diamond\Box\mathtt{enabled}(\mathtt{snd}(\mathtt{P},\mathtt{I},\mathtt{W})) \rightarrow \Box\Diamond\mathtt{in\text{-}fifo}(\mathtt{other}(\mathtt{P}),\mathtt{I},\mathtt{W})$.

Again, we can apply our framework to model check $\Diamond \mathit{success}$ under the above parameterized fairness conditions, since the propositions in the fairness formulas are tractable, owing to the fact that each variable in the propositions is included in either the state or the one-step proof term for the corresponding satisfaction equation, such as:

```
eq [P : SP, AP, LP, LQ, IP, OP, L [I, W] G] & PROC |= in-fifo(P, I, W) = true .
eq [P : SP, AP, LP, LQ, OP, M ; [I, Q] ; N, FP] & PROC |= in-rec(P, I, Q) = true .
```

6 Experimental Results

We have implemented our algorithm in the Maude system by extending the existing state/event-based LTL model checker [2]. We have compared it with other explicit-state model checkers, such as PAT [23] and SPIN [14]. We then have tested our algorithm on complex examples involving dynamic fairness conditions. 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 with the liveness property $\Box\neg \mathit{deadlock} \rightarrow \Diamond \mathit{eating}(1)$, where *deadlock* is considered as an event proposition.[10] In order to estimate how the parameter computation affects the performance, we have tested two different algorithms: MaudeP is the model checking algorithm for parameterized fairness, and MaudeF is one for *ground* fairness where MaudeP is implemented on top of it. 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, even though the abstracted parameter computation is involved. Note that for Spin we had to encode strong fairness conditions into the corresponding LTL since Spin only supports weak fairness.

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. In this case, however, we do not include the comparison with the other tools, since the fairness specification in such tools is difficult by reason of dynamic controls and complex data structures. Table 2a presents the

[10] 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	MaudeP States	MaudeP Time	MaudeF States	MaudeF Time	Pat States	Pat Time	Spin States	Spin Time
Weak Only (Counter Example)	6	913	< 0.1	913	< 0.1	1596	1.0	672	< 0.1
	7	2418	0.2	2418	0.1	5718	5.1	2765	0.2
	8	11092	1.2	11092	0.9	21148	33.5	9404	0.8
Strong/Weak (Valid)	6	5777	5.0	5777	1.8	18101	3.9		
	7	24475	22.7	24475	11.5	69426	16.1	> 30 minutes	
	8	103681	98.0	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	

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

The general fairness framework presented here is based on parameter abstraction. It can be used to specify and verify a large class of fairness constrains that can be expressed by parameterized fairness. We have presented an automata-theoretic on-the-fly algorithm to model check SE-LTL properties under parameterized fairness conditions, 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 (generalized) 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.

References

1. Agha, G.: Actors: A Model of Concurrent Computation in Distributed Systems, Series in Artificial Intelligence. MIT Press 11(12), 12 (1986)
2. Bae, K., Meseguer, J.: The Linear Temporal Logic of Rewriting Maude Model Checker. In: WRLA. LNCS, vol. 6381, pp. 208–225. Springer (2010)
3. Bouajjani, A., Jonsson, B., Nilsson, M., Touili, T.: Regular model checking. In: Computer Aided Verification. pp. 403–418. Springer (2000)
4. Chaki, S., Clarke, E.M., Ouaknine, J., Sharygina, N., Sinha, N.: State/event-based software model checking. In: In Integrated Formal Methods. Springer (2004)
5. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press (2001)
6. Clavel, M., Durán, F., Eker, S., Meseguer, J., Lincoln, P., Martí-Oliet, N., Talcott, C.: All About Maude – A High-Performance Logical Framework, LNCS, vol. 4350. Springer (2007)
7. Cohen, A., Namjoshi, K., Saar, Y.: A dash of fairness for compositional reasoning. In: Computer Aided Verification. pp. 543–557. Springer (2010)
8. Couvreur, J., Duret-Lutz, A., Poitrenaud, D.: On-the-fly emptiness checks for generalized Büchi automata. Model Checking Software pp. 169–184 (2005)
9. Dams, D., Gerth, R., Grumberg, O.: Abstract interpretation of reactive systems. ACM Transactions on Programming Languages and Systems 19, 253–291 (1997)
10. Duret-Lutz, A., Poitrenaud, D., Couvreur, J.M.: On-the-fly emptiness check of transition-based Streett automata. In: ATVA. LNCS, vol. 5799. Springer (2009)
11. Emerson, E.A., Lei, C.: Modalities for model checking: Branching time logic strikes back. Science of Computer Programming 8(3), 275–306 (1987)
12. Francez, N.: Fairness. Springer (1986)
13. Hoenicke, J., Olderog, E., Podelski, A.: Fairness for Dynamic Control. Tools and Algorithms for the Construction and Analysis of Systems pp. 251–265 (2010)
14. Holzmann, G.: The SPIN model checker: Primer and reference manual. Addison Wesley Publishing Company (2004)
15. Kesten, Y., Pnueli, A., Raviv, L., Shahar, E.: Model checking with strong fairness. Formal Methods in System Design 28(1), 57–84 (2006)
16. Kramer, J., Magee, J.: The evolving philosophers problem: Dynamic change management. Software Engineering, IEEE Transactions on 16(11), 1293–1306 (2002)
17. Lamport, L.: Fairness and hyperfairness. Distributed Computing 13(4) (2000)
18. Latvala, T.: Model checking LTL properties of high-level Petri nets with fairness constraints. In: ICATPN. LNCS, vol. 2075, pp. 242–262. Springer (2001)
19. Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science 96(1), 73–155 (1992)
20. Meseguer, J.: Localized fairness: A rewriting semantics. In: RTA. LNCS, vol. 3467, pp. 250–263. Springer (2005)
21. Meseguer, J., Palomino, M., Martí-Oliet, N.: Equational abstractions. Theoretical Computer Science 403(2-3), 239–264 (2008)
22. Queille, J., Sifakis, J.: Fairness and related properties in transition systems — a temporal logic to deal with fairness. Acta Informatica 19(3), 195–220 (1983)
23. Sun, J., Liu, Y., Dong, J., Pang, J.: PAT: Towards flexible verification under fairness. In: Computer Aided Verification. pp. 709–714. Springer (2009)
24. Tel, G.: Introduction to distributed algorithms. Cambridge Univ. Press (2000)
25. Vardi, M.: Automata-theoretic model checking revisited. In: Verification, Model Checking, and Abstract Interpretation. pp. 137–150. Springer (2007)