Verifying Universal Properties of Parameterized Networks*

Kai Baukus[1] Yassine Lakhnech[2†] Karsten Stahl[1]

[1] Institute of Computer Science and Applied Mathematics
University of Kiel
Preusserstr. 1–9, D-24105 Kiel, Germany
{kba, kst}@informatik.uni-kiel.de

[2] VERIMAG, Centre Equation
2 Av. de Vignate, 38610 Giéres, France
lakhnech@imag.fr

Abstract

In this paper, we present a method for the verification of so-called *universal* properties of fair parameterized networks of similar processes, that is, properties of the form $\forall p_1 \ldots p_n : \psi$, where ψ is a quantifier-free LTL formula. To prove an universal property of a parameterized network, we first model the infinite family of networks by a single fair WS1S transition system, that is, a transition system whose variables are set (2nd-order) variables and whose transitions are described in WS1S. Then, we abstract the WS1S system into a finite state system that can be model-checked. We present a generic abstraction relation for verifying universal properties as well as an algorithm for computing an abstract system. However, the abstract system may contain infinite computations that have no corresponding fair computations at the concrete level, and hence, in case the property of interest is a progress property, verification may fail because of this. Therefore, we present methods that allow to synthesize fairness conditions from the parameterized network and discuss under which conditions and how to lift fairness conditions of this network to fairness conditions on the abstract system. We implemented our methods in a tool, called PAX, and applied it to several examples.

*This work has been partially supported by the Esprit-LTR project Vires.
†Contact Author.

1 Introduction

Problem statement and contributions In this paper, we present a method for the verification of so-called *universal* properties of fair parameterized networks of similar processes. In other words, we present a method for tackling the following problem:

Given a parameterized network $P_1 \parallel \cdots \parallel P_n$*, fairness conditions, and a quantifier-free linear-time temporal property* $\psi(p_1 \ldots, p_k)$*, we want to prove* $P_1 \parallel \cdots \parallel P_n \models \forall p_1 \ldots, p_k \leq n : \psi(p_1 \ldots, p_k)$*, for every* $n \in \omega$*, i.e., every fair computation of* $P_1 \parallel \cdots \parallel P_n$ *satisfies* $\psi(p_1 \ldots, p_k)$*.*

Our method uses the verification by abstraction approach [CC77, CGL94, LGS$^+$95, DGG94] and consists of the following steps:

1. Transforming the infinite family of fair networks $P_1 \parallel \cdots \parallel P_n$ to a single fair transition system $\mathcal{S}$ whose variables range over finite sub-sets of ω and whose transitions are expressed in WS1S, the weak second-order logic of one-successor [Büc60]. We call such systems fair WS1S transition systems. The transformation is such that there is a one-to-one onto mapping (a bijection) between the fair computations of the parameterized network and the fair computations of the WS1S transition system.

2. Constructing an abstraction relation that maps the states of the WS1S transition system $\mathcal{S}$ to abstract states which are valuations of boolean variables. We present a generic abstraction relation for verifying universal properties.

3. Automatically constructing a finite abstract system $\mathcal{S}_A$ that is an abstraction of $\mathcal{S}$ which implies that every computation of $\mathcal{S}$ can be mapped to a computation of $\mathcal{S}_A$. Moreover, we construct an abstract formula ψ_A such that if $\mathcal{S}_A$ satisfies ψ_A then, using the preservation results of [CGL94, LGS$^+$95, DGG94], we can deduce $P_1 \parallel \cdots \parallel P_n \models \forall p_1 \ldots, p_k \leq n : \psi(p_1 \ldots, p_k)$, for every $n \in \omega$.

4. Since the abstract system is finite, we can use model-checking to verify that it satisfies ψ_A. However, verifying progress properties using abstractions often fails because of infinite computations in the abstract system that do not correspond to fair infinite ones in the concrete one. To mitigate this problem, we augment the abstract system with *safe* fairness conditions, that is, conditions that only remove infinite computations that do not correspond to concrete ones. We present two techniques for synthesizing fairness conditions from the concrete system that can be safely added to the abstract one:

   (a) An algorithm that given a WS1S formula characterizing a ranking function computes pairs of sets of transitions expressing strong fairness conditions that are *guaranteed* to hold for the parameterized network, and hence, abstractions of them can be safely added at the abstract level.

   (b) A method that allows the generation of fairness conditions at the abstract level from the fairness conditions of the parameterized network. In particular, we discuss which kind of weak/strong fairness can be lifted from the concrete to the abstract level.

We implemented our method in a tool, we call PAX [1], that uses the decision procedures of

[1] http://www.informatik.uni-kiel.de/~kba/PAX

MONA [HJJ+96] to check the satisfiability of WS1S formulae. We then applied our tool and method to several examples including Dijkstra's and Szymanski's mutual exclusion algorithms.

Relevance and related work There has been recently much interest in the automatic and semi-automatic verification of parameterized networks. The methods presented in [GS92, EN96] show that for classes of ring networks of arbitrary size and client-server systems, there exists k such that the verification of the parameterized network can be reduced to the verification of networks of size up to k. Alternative methods presented in [KM89, WL89, BCG89, SG89, HLR92, LHR97] are based on induction on the number of processes. These methods require finding a network invariant that abstracts any arbitrary number of processes with respect to a pre-order that preserves the property to be verified. While this method has been originally presented for linear networks, it has been generalized in [CGJ95] to networks generated by context-free grammars. In [CGJ95], abstract transition systems were used to specify the invariant. An abstract transition system consists of abstract states specified by regular expressions and transitions between abstract states. The idea of representing sets of states of parameterized networks by regular languages is applied in [KMM+97], where additionally finite-state transducers are used to compute predecessors. These ideas are applied to linear networks as well as to processes arranged in a tree architecture and semi-automatic symbolic backward analysis methods for solving the reachability problem are given. The work in [ABJN99, JN00, PS00] extends the ideas in [KMM+97] by considering the effect of applying infinitely often a transition that satisfies certain restrictions. These restrictions allow to characterize the effect of the repeated application of the transition by finite-state transducers. The work in [PS00] also presents an interesting algorithm for checking liveness properties of parameterized systems. In [BBLS00], we presented a method based on abstraction for verifying invariance and a restricted class of liveness properties of parameterized networks that is applicable to a larger class than in [ABJN99]. The method we present here allows us to deal with a larger class of progress properties.

2 Preliminaries

In this section we briefly recall the definition of weak second order theory of one successor (WS1S for short) [Büc60, Tho90] and introduce the logic we use to describe the class of parameterized systems we are interested in.

Terms of WS1S are built up from the constant 0 and 1st-order variables by applying the successor function $\mathrm{suc}(t)$ ("$t+1$"). *Atomic formulae* are of the form b, $t = t'$, $t < t'$, $t \in X$, where b is a boolean variable, t and t' are terms, and X is a set variable (2nd-order variable). WS1S-formulae are built up from atomic formulae by applying the boolean connectives as well as quantification over both 1st-order and 2nd-order variables. *First-order monadic formulae* are WS1S-formulae in which no 2nd-order variables occur.

WS1S-formulae are interpreted in models that assign finite sub-sets of ω to 2nd-order variables and elements of ω to 1st-order variables. The interpretation is defined in the usual way.

Given a WS1S formula f, we denote by $[\![f]\!]$ the set of models of f. The set of free variables in f is denoted by $\mathit{free}(f)$. We say that f is (1st-order) *closed*, if it does not contain (1st-order) free variables. In addition to the usual abbreviations, given a 2nd-order variable P, we write

$\forall_P i : f$ instead of $\forall i : i \in P \rightarrow f$ and $\exists_P i : f$ instead of $\exists i : i \in P \wedge f$.

Finally, we recall that by Büchi [Büc60] and Elgot [Elg61] the satisfiability problem for WS1S is decidable. Indeed, the set of all models of a WS1S-formula is representable by a finite automaton (see, e.g., [Tho90]).

Let n be a variable. To define parameterized systems, we introduce the set $\mathrm{AF}(n)$ of formulae f defined by:

$$\begin{aligned} t &::= 0 \mid \mathrm{suc}(t) \mid i \\ f &::= t < t \mid t = t \mid b[x] \mid \neg f \mid f \wedge f \mid \forall_n x : f \mid \exists_n x : f \ , \end{aligned}$$

where x is a *position variable*, b is a *boolean array variable*, and i is 1st-order variable. Let $m \in \omega$. We denote by Σ_m the set of evaluations s such that $s(n) = m$, $s(x) \in \{0, \cdots, m-1\}$ and $s(b) : \{0, \cdots, m-1\} \rightarrow \{\mathrm{true}, \mathrm{false}\}$. Then, formulae in $\mathrm{AF}(n)$ are interpreted over evaluations $s \in \bigcup_{m \in \omega} \Sigma_m$ in the usual way. In the sequel, we also assume the usual notion of free variables, closed formulae, etc. for $AF(n)$.

3 Parameterized Networks and WS1S Transition Systems

We now introduce a class of parameterized systems for which we develop an abstraction-based verification technique.

Definition 3.1 (Fair Monadic Transition Systems)
A fair monadic transition system $S(i, n)$, *parameterized by n and i, where n and i are variables ranging over natural numbers, is described by a tuple $(V, \Theta, \mathcal{T}, \mathcal{J}, \mathcal{C})$ consisting of:*

- $V = \{b_1, \ldots, b_k\}$ *where each b_j, $1 \leq j \leq k$, is a boolean array of length n. For each $j \in [1, \ldots, k]$ and $i \in [0, \ldots, n-1]$, $b_j[i]$ models a boolean local variable of process P_i.*
- Θ *is a formula in* $\mathrm{AF}(n)$ *with free$(\Theta) \subseteq V \cup \{i\}$ and which describes the set of initial states.*
- $\mathcal{T}$ *is a finite set of transitions where each $\tau \in \mathcal{T}$ is given by a formula $\rho_\tau \in \mathrm{AF}(n)$ such that free$(\rho_\tau) \subseteq V \cup V' \cup \{i\}$ and V' contains a primed copy for each variable in V.*
- $\mathcal{J}$ *is a finite set of weak fair*[2] *(parameterized) transitions. In a computation a weak fair transition has to be eventually taken when continuously enabled.*
- $\mathcal{C}$ *is a finite set of strong fair*[3] *(parameterized) transitions. A strong fair transition has to be taken infinitely often whenever it is infinitely often enabled.* □

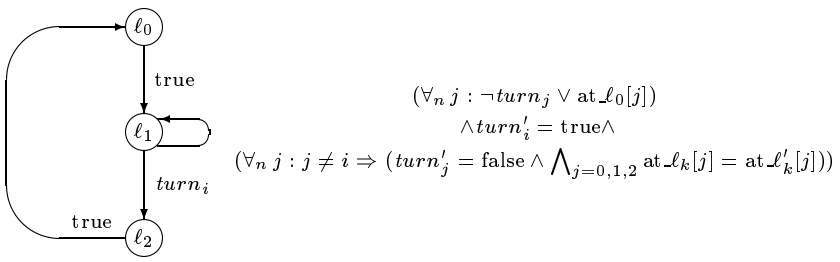
We denote by $\mathcal{P}_m$ the parallel composition $\|_{l=0}^{m-1} S(l, m)$, where $S(l, m)$ is obtained from $S(i, n)$ by substituting m for n and l for i and where $\|$ denotes the interleaving-based parallel composition (asynchronous product). Notice, that if we identify the boolean array variables b_j by the m boolean variables $b_j[1], \cdots, b_j[m]$, the formulae describing the initial states as well as the transitions of $\|_{l=0}^{m-1} S(l, m)$ are first-order WS1S formulae whose free variables are in $\mathcal{V}$, respectively, $\mathcal{V} \cup \mathcal{V}'$. $\mathcal{P}_m$ is a transition system in the usual sense, i.e., it does not contain the

[2] Also called just.
[3] Also called compassionate.

parameters n and i. Hence, we assume the definition of a computation of $\mathcal{P}_m$ as known and we denote by $[\![\mathcal{P}_m]\!]$ the set of its fair computations, that is, the computations that fulfill the fairness conditions. Note, that for a parameterized transition $\tau(i)$ the requirements have to be satisfied for all $i < m$. Then, a *monadic parameterized system* $\mathcal{P}$ (MPS for short) built from $S(i,n)$ is the set $\{\mathcal{P}_m \mid m \geq 1\}$.

Example 3.1 As a running example we use a simple mutual exclusion algorithm. We present more sophisticated examples in Section 6.

The example consists of a parameterized network of processes where each process is described as follows:



The transition from ℓ_0 to ℓ_1 is weak fair whereas the loop from ℓ_1 to ℓ_1 is strong fair. Initially, all processes are at ℓ_0. Location ℓ_2 represents the critical section.

It is easy to see how each process P_i can be described in our setting using a boolean array variable at_ℓ_m for each control point $\ell_m[i]$. For instance, the self-loop in ℓ_1 can be described by the following formula in AF(n):

$$\text{at_}\ell_1[i] \wedge (\forall_n\, j : \neg turn[j] \vee \text{at_}\ell_0[j]) \wedge \text{at_}\ell'_1[i] \wedge turn'[i] = \text{true} \wedge \bigwedge_{m=0,2} \neg\text{at_}\ell'_m[i] \wedge$$
$$(\forall_n\, j : j \neq i \Rightarrow (turn'[j] = \text{false} \wedge \bigwedge_{j=0,1,2} \text{at_}\ell_k[j] = \text{at_}\ell'_k[j])).$$

□

We will verify that the algorithm satisfies the mutual exclusion property as well as the universal property that each process p reaches its critical section infinitely often, i.e., $\forall_n\, p : \Box\Diamond\text{at_}\ell_2[p]$.

WS1S Transition Systems We introduce WS1S transition systems which are transition systems with variables ranging over finite sub-sets of ω and show how they can be used to represent infinite families of monadic transition systems. In order to simulate the behavior of fair parameterized systems we also have to add the notion of fairness to WS1S transition systems.

Definition 3.2 (Fair WS1S Transition Systems)
A fair WS1S transition system $\mathcal{S} = (\mathcal{V}, \Theta, \mathcal{T}, \mathcal{J}, \mathcal{C})$ *is given by the following components:*

- $\mathcal{V} = \{X_1, \ldots, X_k\}$*: A finite set of second order variables* X_i *ranging over finite sets of natural numbers.*
- Θ*: A WS1S formula with free*$(\Theta) \subseteq \mathcal{V}$ *describing the initial condition of the system.*
- $\mathcal{T}$*: A finite set of transitions where each* $\tau \in \mathcal{T}$ *is represented as a WS1S formula* $\rho_\tau(\mathcal{V}, \mathcal{V}')$*, i.e.,* free$(\rho_\tau) \subseteq \mathcal{V} \cup \mathcal{V}'$.

- *$\mathcal{J}$: A set of pairs of second order variables expressing a weak fairness condition. Each pair $(X_m, X_{m'})$ requires, for each $i \in \omega$, the weak fairness condition that i cannot be continuously in X_m without being eventually in $X_{m'}$, that is, $\forall i \in \omega : (\Diamond\Box(i \in X_m) \rightarrow \Box\Diamond(i \in X_{m'}))$.*
- *$\mathcal{C}$: A set of pairs $(X_m, X_{m'})$ of second order variables expressing the strong fairness condition $\forall i \in \omega : (\Box\Diamond(i \in X_m) \rightarrow \Box\Diamond(i \in X_{m'}))$.* □

A state s of $\mathcal{S}$ is a mapping from the variables in $\mathcal{V}$ into finite sub-sets of ω. A *computation* of $\mathcal{S}$ is a sequence $(s_i)_{i\in\omega}$ of states such that $\Theta[s_0(\mathcal{V})/\mathcal{V}]$ and $\bigvee_{\tau\in\mathcal{T}} \tau[s_i(\mathcal{V}), s_{i+1}(\mathcal{V})/\mathcal{V}, \mathcal{V}']$ are valid formulae. A computation $(s_i)_{i\in\omega}$ satisfies a weak fairness condition $(X_m, X_{m'}) \in \mathcal{J}$ iff the following condition holds for every $x \in \omega$:

if $\exists i \in \omega.\forall j \geq i : x \in s_j(X_m)$, then there exist infinitely many i's such that $x \in s_i(X_{m'})$.

The computation satisfies the strong fairness condition $(X_m, X_{m'}) \in \mathcal{C}$ iff the following condition holds for every $x \in \omega$:

if there exist infinitely many i's such that $x \in s_i(X_m)$, then there exist infinitely many i's such that $x \in s_i(X_{m'})$.

Then, a *fair computation* of $\mathcal{S}$ is a computation that satisfies all fairness conditions in $\mathcal{J}$ and $\mathcal{C}$. Henceforth, we denote the set of fair computations of $\mathcal{S}$ by $[\![\mathcal{S}]\!]$.

Our idea of using WS1S systems to model parameterized networks is inspired by an idea often applied to show that Peano arithmetic is expressive enough for encoding strongest post-conditions of While-programs (e.g. [dB80]). The state of a network $P_0 \parallel \cdots \parallel P_{n-1}$ is a vector $(s_0, \cdots, s_n)$ of length n of local states where each local state s_i interprets the local variables of process P_i. Thus, if each process P_i contains two local boolean variables $b_1[i]$ and $b_2[i]$, Fig. 1 describes a possible state of the network. Now, our idea is to describe the same state from a horizontal point of view using for each boolean array variable b_m a set variable B_m which will contain a process index i iff the interpretation of the local variable $b_m[i]$ in the state is true.

	Vertical View						
	P_1	...	P_i	...	P_{n-1}		
b_1	0	...	0	...	0	$\emptyset$	Horizontal View
b_2	0	...	1	...	1	$[i, n-1]$	

Figure 1: A configuration

Regular expressions have been used in [KMM+97, ABJN99, JN00] to represent sets of configurations of parameterized networks. In these works, the alphabet of the regular expressions consists of one symbol for each local state and a configuration is represented by a word over this alphabet. Thus, if each process contains k variables, the alphabet contains 2^k different symbols and the usual translation of the regular expressions to WS1S (cf. [Tho90]) results into formulae with 2^k free second-order variables. Our encoding only uses k second-order variables.

Relating parameterized and WS1S transition systems We define a translation that maps an MPS to a WS1S system with the same set of (fair) computations. In particular, we have to give translations to WS1S for the initial state condition, the translations, and the fairness conditions.

We fix a 2nd-order variable P that is used to model the set of indices of the processes up to n. The translation from MPS to WS1S systems uses a function tr from AF(n) into WS1S. This function replaces in an AF(n)-formula all occurrences of atomic sub-formulae of the form $b[i]$ by $i \in B$, all n by $\max(P) + 1$[4], and λ_n by λ_P where λ is one of the quantifiers $\forall$ or $\exists$. Consider an MPS system $\mathcal{P}$ built from $S(i,n)$ where $S(i,n) = (V, \Theta, \mathcal{T}, \mathcal{J}, \mathcal{C})$. Define a WS1S system $(\widetilde{V}, \widetilde{\Theta}, \widetilde{\mathcal{T}}, \widetilde{\mathcal{J}}, \widetilde{\mathcal{C}})$ by constructing the variable set, the initial condition, the transitions, and the fairness conditions as follows:

- For each boolean array b_k in V, $\widetilde{V}$ contains the variable B_k. For each transition $\tau \in \mathcal{T}$, introduce two second order variables E_τ, T_τ. Additionally, $\widetilde{V}$ contains the set variable P.
- Let $\widetilde{\Theta}$ be

$$\exists n : P = \{0, \ldots, n-1\} \wedge \bigcup_{l=1}^{k} B_l \subseteq P \wedge \bigcup_{\tau \in \mathcal{T}} T_\tau = \emptyset$$
$$\wedge \bigwedge_{\tau \in \mathcal{T}} E_\tau = \{i \in P \mid \exists V' : \rho_\tau(V, V')\} \wedge (\forall_P i : tr(\Theta)) \ .$$

- Let $\widetilde{\mathcal{T}}$ be the set containing for each $\tau \in \mathcal{T}$ a transition represented as the WS1S formula

$$\exists_P i : \quad tr(\rho_\tau) \wedge P = P' \wedge \bigcup_{l=1}^{k} B'_l \subseteq P' \wedge \bigcup_{\tau \neq \tau' \in \mathcal{T}} T'_{\tau'} = \emptyset \wedge T'_\tau = \{i\}$$
$$\wedge \bigwedge_{\tau \in \mathcal{T}} E'_\tau = \{i \in P \mid \exists V'' : \rho_\tau(V', V'')\} \ .$$

- Let $\widetilde{\mathcal{J}} = \{(E_\tau, T_\tau) \mid \tau \in \mathcal{J}\}$.
- Let $\widetilde{\mathcal{C}} = \{(E_\tau, T_\tau) \mid \tau \in \mathcal{C}\}$.

We denote the transformation of an MPS system $\mathcal{P}$ by $Tr(\mathcal{P})$.

Example 3.2 Recall our running example 3.1. The translation into a WS1S system introduces three set variables $\mathrm{At_}\ell_0, \mathrm{At_}\ell_1, \mathrm{At_}\ell_2$ corresponding to the control locations, the set variable $Turn$ corresponding to $turn$ and two additional set variables E_τ and T_τ for each transition to express the fairness conditions. Let $\widetilde{V}$ denote this set of variables. If we denote by τ_0 the self-loop in ℓ_1, then $\widetilde{\mathcal{C}} = \{(E_{\tau_0}, T_{\tau_0})\}$. The liveness property we will check later can then be expressed by $\forall_P p : \Box\Diamond(p \in \mathrm{At_}\ell_2)$. For the sake of illustration, we show the translation of τ_0:

$\exists_P i : i \in \mathrm{At_}\ell_1 \wedge (\forall_P j : j \notin Turn \vee j \in \mathrm{At_}\ell_0)$
$\wedge i \in \mathrm{At_}\ell'_1 \wedge i \in Turn' \wedge (\forall_P j : j \neq i \Rightarrow (j \notin Turn' \wedge \bigwedge_{j=0,1,2}(j \in \mathrm{At_}\ell_k \Leftrightarrow j \in \mathrm{At_}\ell'_k))) \wedge$
$P = P' \wedge \bigcup_{B \in \widetilde{V}} B' \subseteq P' \wedge \bigcup_{\tau_0 \neq \tau'} T'_{\tau'} = \emptyset \wedge T'_{\tau_0} = \{i\} \wedge \bigwedge_{\tau \in \mathcal{T}} E'_\tau = \{i \in P \mid \exists V'' : \rho_\tau(V', V'')\}$.

□

The relationship between an MPS $\mathcal{P}$ and its translation is stated in the following theorem.

[4]It is not difficult to check that $\max(P)$ can be expressed in WS1S.

Proposition 3.1 (Relating Boolean and WS1S Systems) *Let $\mathcal{P}$ be an MPS built from $S(i,n) = (V, \Theta, \mathcal{T}, \mathcal{J}, \mathcal{C})$ and $m \in \omega$. Then, $\mathcal{P}_m$ and $Tr^*(\mathcal{P})$ are bisimilar, where $Tr^*(\mathcal{P})$ is obtained from $Tr(\mathcal{P})$ by taking as initial condition $\widetilde{\Theta} \equiv P = \{0, \ldots, m-1\} \wedge \bigcup_{l=1}^{k} B_l \subseteq P \wedge \forall_P i : tr(\Theta)$.*

Moreover, each fair computation in $\mathcal{P}_m$ corresponds to a fair computation in $Tr^*(\mathcal{P})$ and conversely, each computation of $Tr^*(\mathcal{P})$ corresponds to a computation in $\mathcal{P}_m$ for some $m \in \omega$. □

As a final remark, notice that we can translate a slightly more general class of parameterized networks into WS1S transition systems than MPSs. Indeed, we can allow a finite number of global variables g that range over natural numbers and allow terms $g + c$, where c is constant. Notice, however, that, for instance, the term $2 \cdot g$ cannot be described in WS1S (see [Tho90]).

4 Abstracting WS1S Systems

In Section 3, we have shown how parameterized networks of fair monadic transition systems can be translated into fair WS1S systems. This translation allows us to consider a single, though infinite-state, transition system instead of an infinite family of systems.

In the following, we present a method to construct a finite abstract system of a given WS1S systems. Then, we show in Section 5 how the obtained abstract system can be enriched with fairness conditions such that interesting progress properties of the WS1S system can be verified. Let us first define what we mean by universal temporal properties.

Let Π be a countable set of process indices p that range over natural numbers and let Ξ be a countable set of variables X that range over finite sets of natural numbers. The set LTL of linear-time temporal properties over Π and Ξ is defined as follows:

$$\varphi ::= i \in \omega \mid p \in X \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \mathcal{U} \varphi, \text{ where } i \text{ is a constant in } \omega.$$

As usual, we use that temporal modalities $\Box$ (always) and $\Diamond$ (eventually) which can be introduced as abbreviations.

Formulae in LTL are interpreted over infinite sequences of structures of the form $(\mathcal{I}, \mathcal{I}')$, where $\mathcal{I}$ maps each variable $X \in \Xi$ to a finite sub-set of ω and $\mathcal{I}'$ maps each variable in Π to an element of ω. The definition of the interpretation of LTL is not given here as it is standard.

Let φ be an LTL formula with $\{X_1, \ldots, X_k\}$ as free set-variables and $\{p_1, \ldots, p_n\}$ as free 1st-order variables. Moreover, let $\mathcal{S}$ be a WS1S transition system with $\{X_1, \ldots, X_k\}$ as variables. A computation $(s_i)_{i \in \omega}$ satisfies φ iff for every injective mapping $\mathcal{I}'$ from $\{p_1, \ldots, p_n\}$ into ω, the sequence $(s_i, \mathcal{I}')_{i \in \omega}$ satisfies φ. In other words, the computation $(s_i)_{i \in \omega}$ satisfies φ, if it satisfies all the temporal formulae obtained by instantiating the variables $p_1, \ldots, p_n$. We say that $\mathcal{S}$ satisfies φ, denoted by $\mathcal{S} \models \varphi$, if every fair computation of $\mathcal{S}$ satisfies φ.

A temporal property is called *universal*, if it can be described by a formula in LTL. For instance, mutual-exclusion of Szymanski's algorithm can be described by the formula $\Box\neg(p_1 \in \text{At_}\ell_7 \wedge p_2 \in \text{At_}\ell_7)$, which is an universal temporal property. However, the communal liveness property stating whenever some process is in $\text{At_}\ell_2$, eventually some process (not necessarily the same) reaches $\text{At_}\ell_7$ is not an universal temporal property. On the other hand, the stronger liveness property stating that every process in $\text{At_}\ell_2$ eventually reaches $\text{At_}\ell_7$ is an universal property as it can be described by the property $\Box(p \in \text{At_}\ell_2 \Rightarrow \Diamond p \in \text{At_}\ell_7)$.

The problem we are interested in is given a WS1S system $\mathcal{S}$ and given an universal temporal formula φ to show $\mathcal{S} \models \varphi$.

Abstractions and fair abstractions Given a deadlock-free[5] transition system $\mathcal{S} = (\mathcal{V}, \Theta, \mathcal{T})$ and a total abstraction relation $\alpha \subseteq \Sigma \times \Sigma_A$, we say that $\mathcal{S}_A = (\mathcal{V}_A, \Theta_A, \mathcal{T}_A)$ is an *abstraction* of $\mathcal{S}$ w.r.t. α, denoted by $\mathcal{S} \sqsubseteq_\alpha \mathcal{S}_A$, if the following conditions are satisfied: (1) $s_0 \models \Theta$ implies $\alpha(s_0) \models \Theta_A$ and (2) $\tau \circ \alpha^{-1} \subseteq \alpha^{-1} \circ \tau_A$.

In case Σ_A is finite, we call α finite abstraction relation. Let φ, φ_A be LTL formulae and let $[\![\varphi]\!]$ (resp. $[\![\varphi_A]\!]$) denote the set of models of φ (resp. φ_A). Then, from $\mathcal{S} \sqsubseteq_\alpha \mathcal{S}_A$, $\alpha^{-1}([\![\varphi_A]\!]) \subseteq [\![\varphi]\!]$, and $\mathcal{S}_A \models \varphi_A$ we can conclude $\mathcal{S} \models \varphi$ (here we identify α^{-1} and its pointwise lifting to sequences). In case $\mathcal{S}$ is a fair TS with $\mathcal{F}$ as fairness formula and if $\mathcal{F}_A$ is the fairness formula of $\mathcal{S}_A$, then by requiring $\alpha^{-1}([\![\neg\mathcal{F}_A]\!]) \subseteq [\![\neg\mathcal{F}]\!]$, we have the same preservation result as above. We indicate this type of abstraction by $\mathcal{S} \sqsubseteq^F_\alpha \mathcal{S}_A$.

Next, we explain the main steps of our approach for verifying universal temporal properties before presenting each step in more detail.

Approach Let $\mathcal{S} = (\mathcal{V}, \Theta, \mathcal{T}, \mathcal{J}, \mathcal{C})$ be a WS1S system obtained by translating an MPS $\mathcal{P}$ and let ψ be an universal temporal formula with $\{X_1, \ldots, X_k\} \cup \{p_1, \ldots, p_m\}$ as free variables. To simplify the presentation assume that $m = 1$ and write p instead of p_1. Moreover, we denote by $\psi(i)$ the formula obtained from ψ by replacing p by the constant $i \in \omega$.

For each $i \in \omega$, we construct a finite abstraction relation α_i which maps states of $\mathcal{S}$ to abstract states. The abstract state space defined by α_i is such that it contains for each subformula $i \in X$ of $\psi(i)$ a boolean variable b^i_X and for each X_j an abstract variable b_j. Then, α_i relates a concrete state s to an abstract state s^A iff $s^A(b^i_X) \Leftrightarrow i \in s(X)$ and $s^A(b_j) \Leftrightarrow s(X_j) \neq \emptyset$. Henceforth, let $\widehat{\alpha_i}$ be a predicate defining α_i. Clearly, the abstract state spaces defined by α_i and α_j are the same modulo renaming of the variables b^i_X.

Then, for each $i \in \omega$, one can *effectively* construct a finite abstract system S^i_A and an LTL-formula ψ^i_A such that $S^i_A \models \psi^i_A$ implies $\mathcal{S} \models \psi(i)$. We show that one can effectively construct the set $\{S^i_A \mid i \in \omega\}$ of abstract systems. However, although this set is finite, it is computationally costly to construct. Therefore, we present an algorithm for constructing a *single* finite abstract system S_A which is itself an abstraction of each S^i_A and, as we show, is an abstraction of $\mathcal{S}$. Moreover, we show how to construct an LTL-formula ψ_A such that $S_A \models \psi_A$ implies $\mathcal{S} \models \psi$. Figure 2 summarizes this discussion.

Abstraction relation α_i The set $\mathcal{V}^i_A$ of abstract variables consists of boolean variables. For each set X in the WS1S system $\mathcal{S}$ we have an abstract boolean variable $b^i_X \in \mathcal{V}^i_A$ corresponding to $i \in X$. Thus, in particular we have the variable $b^i_{E_\tau}$ and $b^i_{T_\tau}$, for each $\tau \in \mathcal{T}$. Additionally, for each strong fairness condition $(E_\tau, T_\tau) \in \mathcal{C}$ we introduce boolean variables e_τ, t_τ such that α_i implies:

$$\begin{aligned} e_\tau &\equiv \exists_P\, j : j \in E_\tau \\ t_\tau &\equiv \exists_P\, j : j \in T_\tau \ . \end{aligned}$$

[5]Throughout this paper we only consider deadlock free transition systems which can be achieved by adding an idle transition.

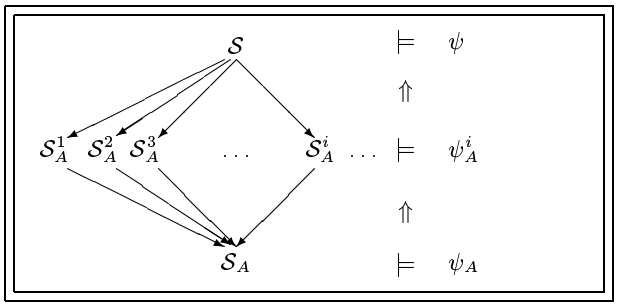


Figure 2: Abstraction structure

For all other global state properties φ that may influence the progress of a certain process p another variable is added for which the abstraction is given by φ. This includes an adequate abstraction of the used natural numbers to express their influence on the behavior of the system.

Henceforth, we also use $\widehat{\alpha_i}(\mathcal{V}', \mathcal{V}_A'^i)$ to denote the predicate obtained from $\widehat{\alpha}_i$ by substituting the unprimed variables with their primed versions.

Construction of $\mathcal{S}_A^i$ It is easy to see that the abstract systems $\mathcal{S}_A^i$ whose transitions are defined by

$$\xi(\alpha_i, \tau) \equiv \exists \mathcal{V}, \mathcal{V}' : \widehat{\alpha}_i(\mathcal{V}, \mathcal{V}_A^i) \wedge \rho_\tau(\mathcal{V}, \mathcal{V}') \wedge \widehat{\alpha}_i(\mathcal{V}', \mathcal{V}_A'^i)$$

and whose initial states are defined by

$$\xi(\alpha_i) \equiv \exists \mathcal{V} : \widehat{\alpha}_i(\mathcal{V}, \mathcal{V}_A^i)$$

satisfies $\mathcal{S} \sqsubseteq_{\alpha_i} \mathcal{S}_A^i$. Now, since $\widehat{\alpha_i}$ and ρ_τ are WS1S formulae, by Büchi and Elgot's result, $\mathcal{S}_A^i$ can be constructed effectively.

We now show that the set $\{S_A^i \mid i \in \omega\}$ can be effectively constructed. For $i \in \omega$, let $\Lambda(p)$ be the following WS1S formula:

$$\bigvee_{\tau \in \mathcal{T}} \exists \mathcal{V}_A^p, \mathcal{V}_A'^p : [(\xi(\alpha_i, \tau)[p/i] \not\Leftrightarrow \xi(\alpha_p, \tau)) \vee \xi(\alpha_i)[p/i] \not\Leftrightarrow \xi(\alpha_p)]$$

where p is a free 1st-order variable and $[p/i]$ denotes the substitutions of the abstract variables in $\mathcal{V}_A^i \cup \mathcal{V}_A'^i$ by their counterparts in $\mathcal{V}_A^p \cup \mathcal{V}_A'^p$. Let $\mathcal{C}(i)$ denote the set of all $k \in \omega$ that satisfy $\Lambda(i)$. Thus, $\mathcal{C}(i)$ consists of the numbers $k \in \omega$ such that S_A^i and S_A^k are different (after substituting the abstract variables of S_A^i by their counterparts in S_A^k). By Büchi and Elgot's result, $\mathcal{C}(i)$ can be effectively constructed. Then, the following algorithm computes $\{S_A^i \mid i \in \omega\}$:

$C := \{1\}$;
While $C \neq \emptyset$ Do
    $x :=$ some element in C; $R := R \cup \{S_A^x\}$; $C := C \cap \mathcal{C}(x)$
oD; return(R)

This algorithm converges, since there are only finitely many different abstract transition systems and each execution of the body of the loop adds a new abstract transition system to R.

Construction of $\mathcal{S}_A$ It is clear that in many cases it is not feasible to compute $\{S_A^i \mid i \in \omega\}$. Therefore, we show how one can construct a system that abstracts each of the elements of this set, and hence, by transitivity of $\sqsubseteq$ abstracts $\mathcal{S}$. The set $\mathcal{V}_A$ of abstract variables of $\mathcal{S}_A$ contains for each abstract variable $b^i \in \mathcal{V}_A^i$ a variable b.

A first solution could consist in defining the transitions of $\mathcal{S}_A$ by the following WS1S formula:

$$\exists \mathcal{V}, \mathcal{V}' : (\exists_P\, p : \widehat{\alpha}_p(\mathcal{V}, \mathcal{V}_A)) \wedge \rho_\tau(\mathcal{V}, \mathcal{V}') \wedge (\exists_P\, p : \widehat{\alpha}_p(\mathcal{V}', \mathcal{V}'_A)). \tag{1}$$

This would lead, however, to a too coarse (non-deterministic) abstract system that would satisfy only a few interesting properties. Instead, a less non-deterministic abstract system can be obtained by the following definition of abstract transitions:

$$\exists_P\, p : \exists \mathcal{V}, \mathcal{V}' : \widehat{\alpha}_p(\mathcal{V}, \mathcal{V}_A) \wedge \rho_\tau(\mathcal{V}, \mathcal{V}') \wedge \widehat{\alpha}_p(\mathcal{V}', \mathcal{V}'_A). \tag{2}$$

Thus, in contrast to (1), in formula (2) we make sure that the choice of p in the concretizations of the source and target states of an abstract transitions is the same. We can then show the following:

Proposition 4.1 *$\mathcal{S}_A$ is an abstraction of $\mathcal{S}$, that is, $\mathcal{S} \sqsubseteq_\alpha \mathcal{S}_A$ with $\alpha = \bigcup_{i \in \omega} \alpha_i$.*

Notice that the formulae above are WS1S formulae, and hence, by Büchi and Elgot's result, the sets of numbers satisfying these formulae can be characterized by finite automata. We use MONA [HJJ+96] to construct these automata.

5 Fair Abstractions

It is well known that an obstacle to the verification of liveness properties using abstraction, is that often the abstract system contains cycles that do not correspond to fair computations of the concrete system. A way to overcome this difficulty is to enrich the abstract system with fairness conditions or more generally ranking functions over well-founded sets that eliminate undesirable computations. This problem is partially tackled in [BBLS00], where we present a marking algorithm that given a reachability state graph of an abstraction of a WS1S system enriches the graph with strong fairness conditions while preserving the property that to each concrete computation corresponds an abstract *fair* one. The enriched graph is used to prove liveness properties of the WS1S systems, and consequently, of the parameterized network.

Here, we first generalize this marking algorithm to deal with more sophisticated ranking functions. Then, we discuss under which requirements the fairness conditions of the parameterized system can be lifted to the finite abstract one. In particular, we show that by requiring some conditions on the abstraction relation, it is sound to lift strong fairness. Weak fairness can only be lifted for a distinguished process. Finally, we illustrate by examples that the combination of our generalized marking algorithm with our results on lifting fairness, allows us to *automatically* verify interesting universal liveness properties of non-trivial algorithms such as Dijkstra's and Szymanski's mutual exclusion algorithms.

Throughout this section, we fix a WS1S system $\mathcal{S} = (\mathcal{V}, \Theta, \mathcal{T}, \mathcal{J}, \mathcal{C})$ obtained by translating an MPS $\mathcal{P}$ and an abstraction relation α constructed as explained in Section 4. Then, let $\mathcal{S}_A = (\mathcal{V}_A, \Theta_A, \mathcal{T}_A)$ be the finite abstract system (without fairness) obtained by the method introduced in Section 4. We show how to add fairness conditions to $\mathcal{S}_A$ leading to a fair abstract system $\mathcal{S}_A^F = (\mathcal{V}_A, \Theta_A, \mathcal{T}_A, \mathcal{J}_A, \mathcal{C}_A)$ such that $\mathcal{S} \sqsubseteq_\alpha^F \mathcal{S}_A^F$.

Marking algorithm We use WS1S formulae to express ranking functions. Let $\chi(i, X_1, \cdots, X_k)$ be a predicate with i as free 1st-order variable and $X_1, \cdots, X_k \in \mathcal{V}$ as free 2nd-order variables. Given a state s of $\mathcal{S}$, i.e., a valuation of the variables in $\mathcal{V}$, the ranking value $\zeta(s)$ associated to s by ζ is the cardinality of $\{i \in \omega \mid \chi(i, s(X_1), \ldots, s(X_k))\}$.

The marking algorithm we present labels each abstract transition of the abstract system with one of the symbols $\{+_\chi, -_\chi, =_\chi\}$. Intuitively, an abstract transition τ_A is labeled by $-_\chi$, if it is guaranteed that the concrete transition τ associated with τ_A decreases the ranking value, i.e., $(s, s') \in \tau$ implies $\zeta(s) > \zeta(s')$. Similarly for $+_\chi$ and $=_\chi$.

Algorithm

Input: WS1S system $\mathcal{S} = (\mathcal{V}, \Theta, \mathcal{T})$, abstract system $\mathcal{S}_A = (\mathcal{V}_A, \Theta_A, \mathcal{T}_A)$, set of predicates $\chi(i, X_1, \cdots, X_k)$

Output: Labeling of $\mathcal{T}_A$

Description: For each $\chi(i, X_1, \cdots, X_k)$, for each edge $\tau_A \in \mathcal{T}_A$, let τ be the concrete transition in $\mathcal{T}$ corresponding to τ_A. Moreover, let $\Delta(\chi, \tau, \prec)$, with $\prec \in \{\subset, =\}$, denote the WS1S formula:

$$\widehat{\alpha}(\mathcal{V}, \mathcal{V}_A) \wedge \widehat{\alpha}(\mathcal{V}', \mathcal{V}'_A) \wedge \rho_\tau(\mathcal{V}, \mathcal{V}') \Rightarrow \{i \mid \chi'(i)\} \prec \{i \mid \chi(i)\} \ .$$

Then, mark τ_A with $-_\chi$, if $\Delta(X, \tau, \subset)$ is valid, mark τ_A with $=_\chi$, if $\Delta(X, \tau, =)$ is valid; otherwise mark τ_A with $+_\chi$.

Now, for a set formula χ we denote with $\mathcal{T}^+_\chi$ the set of edges labeled with $+_\chi$. Then, we add for each such χ and each transition τ_A labeled with $-_\chi$ the fairness condition $(\tau_A, \mathcal{T}^+_\chi)$ which states that τ_A can only be taken infinitely often when one of the transitions in $\mathcal{T}^+_\chi$ are taken infinitely often.

The presented algorithm is indeed a generalization of the one in [BBLS00] because one can choose $i \in X$ as $\chi(i, X)$ for each $X \in \mathcal{V}$. To express that the maximum (resp. minimum) of a certain set X decreases one chooses $\chi(i, X) \equiv \exists k \in X : i \leq k$ (resp. $\chi(i, X) \equiv X \neq \emptyset \wedge \forall k \in X : i \leq k$. When dealing with MPS translations one has the set variable P representing the processes of the original system. All the elements of set variables are meant to be process indices. Then, also an unbounded increasing of the maximum (resp. minimum) of a certain set is impossible and therefore of interest. So, one can choose $\chi(i, X) \equiv \forall k \in X : i \geq k$ (resp. $\chi(i, X) \equiv \exists k \in X : i \geq k$).

Lifting fairness Recall that by definition of α (see Section 4), we introduce the abstract variables $e_\tau \equiv \exists_P i : i \in E_\tau$ and $t_\tau \equiv \exists_P i : i \in T_\tau$. We now argue that it is safe to augment $\mathcal{S}_A$ with the strong fairness $\mathcal{C}_A = \{(e_\tau, t_\tau) \mid (E_\tau, T_\tau) \in \mathcal{C}\}$, i.e., if e_τ is infinitely often true, then also t_τ is infinitely often true. Consider a computation where e_τ is infinitely often true, that is, $\exists_P i : i \in E_\tau$ is infinitely often true. Now, each instance of the parameterized system only contains a bounded number of processes, hence, by König's lemma, there must exists some i such that $i \in E_\tau$ infinitely often in this computation. Therefore, by the strong fairness condition of the concrete system, we must have $i \in T_\tau$ infinitely often, and hence, the computation satisfies $\Box\Diamond(\exists_P i : i \in T_\tau)$. Consequently, we have the following:

Lemma 5.1 *Under the assumptions above we have* $\mathcal{S} \sqsubseteq^F_\alpha \mathcal{S}_A$. □

The reasoning above does not hold for weak fairness. Indeed, $\Diamond\Box e_\tau$ may hold for a computation without the existence of an i with $\Diamond\Box(i \in E_\tau)$.

Recall also that as explained in Section 4, we introduce for each transition of the distinguished process p abstract variables b_{E_τ} and b_{T_τ} expressing whether the transition is enabled, respectively, taken. We can show that it is safe to augment the abstract system with strong and weak fairness conditions on the transitions of p.

Lemma 5.2 *For the concrete WS1S system $\mathcal{S}$ and the abstract system $\mathcal{S}_A$ we have:*

$$\mathcal{S} \sqsubseteq^F_\alpha \mathcal{S}^F_A \ ,$$

where $\alpha = \bigcup_{i\in\omega} \alpha_i$ for a generic abstraction function α_i and $\mathcal{S}^F_A = (\mathcal{V}_A, \Theta_A, \mathcal{T}_A, \mathcal{J}_A, \mathcal{C}_A)$ with strong fairness requirements $\mathcal{C}_A = \{(b_{E_\tau}, b_{T_\tau}) \mid (E_\tau, T_\tau) \in \mathcal{C}\}$ and weak fairness requirements $\mathcal{J}_A = \{(b_{E_\tau}, b_{T_\tau}) \mid (E_\tau, T_\tau) \in \mathcal{J}\}$. □

Proof: It suffices to prove

$$\mathcal{S} \sqsubseteq^F_{\alpha_i} \mathcal{S}^{i,F}_A \ ,$$

for $i \in \omega$, where we have the corresponding fairness requirements $\mathcal{C}^i_A = \{(b^i_{E_\tau}, b^i_{T_\tau}) \mid (E_\tau, T_\tau) \in \mathcal{C}\}$ and $\mathcal{J}^i_A = \{(b^i_{E_\tau}, b^i_{T_\tau}) \mid (E_\tau, T_\tau) \in \mathcal{J}\}$. For these abstract systems $b^i_{E_\tau}$ expresses directly the enabledness of a transition for a concrete i. Therefore, both weak and strong fairness may be applied also on the abstract level. □

To illustrate the presented methods we consider, for the sake of conciseness, a simplified version of Dijkstra's mutual exclusion algorithm [Dij65] as a monadic transition system. The abstract system corresponding to the original Dijkstra's algorithm as well as other examples can be found at www.informatik.uni-kiel.de/∼kba/pax.

Example 5.1 Let us reconsider our running example 3.1. Recall that we want to verify that the algorithm satisfies the mutual exclusion property as well as the universal property that each process p reaches its critical section infinitely often, i.e., $\forall_n\, p : \Box\Diamond \mathrm{at}_\ell_2[p]$.

According to the method presented in Section 4 we construct the abstract system $\mathcal{S}_A$ from the WS1S translation. For the mutual exclusion property we take as abstract variable $inv \equiv \mathrm{At}_\ell_2 \subseteq \{turn\} \wedge \forall_P\, i, j : (i \in Turn \wedge j \in Turn) \Rightarrow i = j$. Our tool PAX constructs the abstract system and provides translations to several input languages for model-checkers, e.g., Spin and SMV. Also, the abstract state space can be explored to prove that inv is indeed an invariant of the abstract system and, hence, mutual exclusion holds for the original system.

Next, using the marking algorithm, we augment $\mathcal{S}_A$ with the strong fairness requirements $\{(t_{01}, t_{20}), (t_{12}, t_{01}), (t_{20}, t_{12})\}$ to obtain a fair abstract system $\mathcal{S}^F_A$. Moreover, with Lemma 5.1 we can lift the strong fairness (e_{11}, t_{11}). Lemma 5.2 allows us to augment $\mathcal{S}^F_A$ with another strong fairness condition $(b_{E_{11}}, b_{T_{11}})$ for the distinguished process as well as with the weak fairness $(b_{E_{01}}, b_{T_{01}})$.

All the fairness conditions can be expressed as LTL formulae. We used Spin to prove that $\Box\Diamond b_{\mathrm{At}_\ell_2}$ holds in $\mathcal{S}^F_A$ which means that, in the original system, each process reaches its critical section infinitely often. Note, that the fairness conditions are mandatory to verify the progress property.

6 Examples

In this section we present some more complicated examples to prove applicability of our approach to a broader class of algorithms. First we prove individual accessibility for Szymanski's mutual exclusion algorithm. To our knowledge this is the first time this property is automatically proven for this algorithm. In this example, we only have to lift a few fairness conditions. In contrast, the second example strongly relies on a lot of fairness conditions that has to be added to the abstract system to be able to prove individual accessibility there.

6.1 Szymanski's Mutual Exclusion Algorithm

We first give a pseudo code characterization which defines an MPS. The translation to a WS1S system is straightforward and can be seen in detail in [BBLS00].

The code for one process of Szymanski's mutual exclusion algorithm is given as:

ℓ_1: **await** $\forall_n\, j : \mathrm{at_}\ell_1[j] \vee \mathrm{at_}\ell_2[j] \vee \mathrm{at_}\ell_4[j]$
ℓ_2: **skip**
ℓ_3: **if** $\exists_n\, j : \mathrm{at_}\ell_2[j] \vee \mathrm{at_}\ell_5[j] \vee \mathrm{at_}\ell_6[j] \vee \mathrm{at_}\ell_7[j]$
then goto ℓ_4
else goto ℓ_5
ℓ_4: **await** $\exists_n\, j : \mathrm{at_}\ell_5[j] \vee \mathrm{at_}\ell_6[j] \vee \mathrm{at_}\ell_7[j]$
ℓ_5: **await** $\forall_n\, j : \mathrm{at_}\ell_1[j] \vee \mathrm{at_}\ell_2[j] \vee \mathrm{at_}\ell_5[j] \vee \mathrm{at_}\ell_6[j] \vee \mathrm{at_}\ell_7[j]$
ℓ_6: **await** $\forall_n\, j : j < i : \mathrm{at_}\ell_1[j] \vee \mathrm{at_}\ell_2[j] \vee \mathrm{at_}\ell_4[j]$
ℓ_7: goto ℓ_1

The control flow is modeled by a boolean variable $\mathrm{at_}\ell_k$ for each location ℓ_k. The initial condition states that each process starts in ℓ_1.

Considering the transition from ℓ_1 to ℓ_2 to be strong fair and all others to be weak fair we would like to prove that for each process p, $\Box\Diamond \mathrm{at_}\ell_7[p]$ holds.

The skip transition results from the translation of the algorithm into our formalism. The transition should not be omitted, since otherwise the mutual exclusion is trivially fulfilled.

According to our method we take as generic abstraction α_i:

$$
\begin{aligned}
in_1 &\equiv i \in \mathrm{At_}\ell_1 \\
en_{12} &\equiv i \in \mathrm{At_}\ell_1 \wedge \forall_P\, p : p \in \mathrm{At_}\ell_1 \cup \mathrm{At_}\ell_2 \cup \mathrm{At_}\ell_4 \\
tk_{12} &\equiv i \in T_{t_{12}} \\
in_2 &\equiv i \in \mathrm{At_}\ell_2 \\
in_3 &\equiv i \in \mathrm{At_}\ell_3 \\
en_{34} &\equiv i \in \mathrm{At_}\ell_3 \wedge \exists_P\, p : p \in \mathrm{At_}\ell_2 \cup \mathrm{At_}\ell_5 \cup \mathrm{At_}\ell_6 \cup \mathrm{At_}\ell_7 \\
en_{35} &\equiv i \in \mathrm{At_}\ell_3 \wedge \neg\, \exists_P\, p : p \in \mathrm{At_}\ell_2 \cup \mathrm{At_}\ell_5 \cup \mathrm{At_}\ell_6 \cup \mathrm{At_}\ell_7 \\
in_4 &\equiv i \in \mathrm{At_}\ell_4 \\
en_{45} &\equiv i \in \mathrm{At_}\ell_4 \wedge \exists_P\, p : p \in \mathrm{At_}\ell_5 \cup \mathrm{At_}\ell_6 \cup \mathrm{At_}\ell_7 \\
in_5 &\equiv i \in \mathrm{At_}\ell_5 \\
en_{56} &\equiv i \in \mathrm{At_}\ell_5 \wedge \forall_P\, p : p \in \mathrm{At_}\ell_1 \cup \mathrm{At_}\ell_2 \cup \mathrm{At_}\ell_5 \cup \mathrm{At_}\ell_6 \cup \mathrm{At_}\ell_7 \\
in_6 &\equiv i \in \mathrm{At_}\ell_6
\end{aligned}
$$

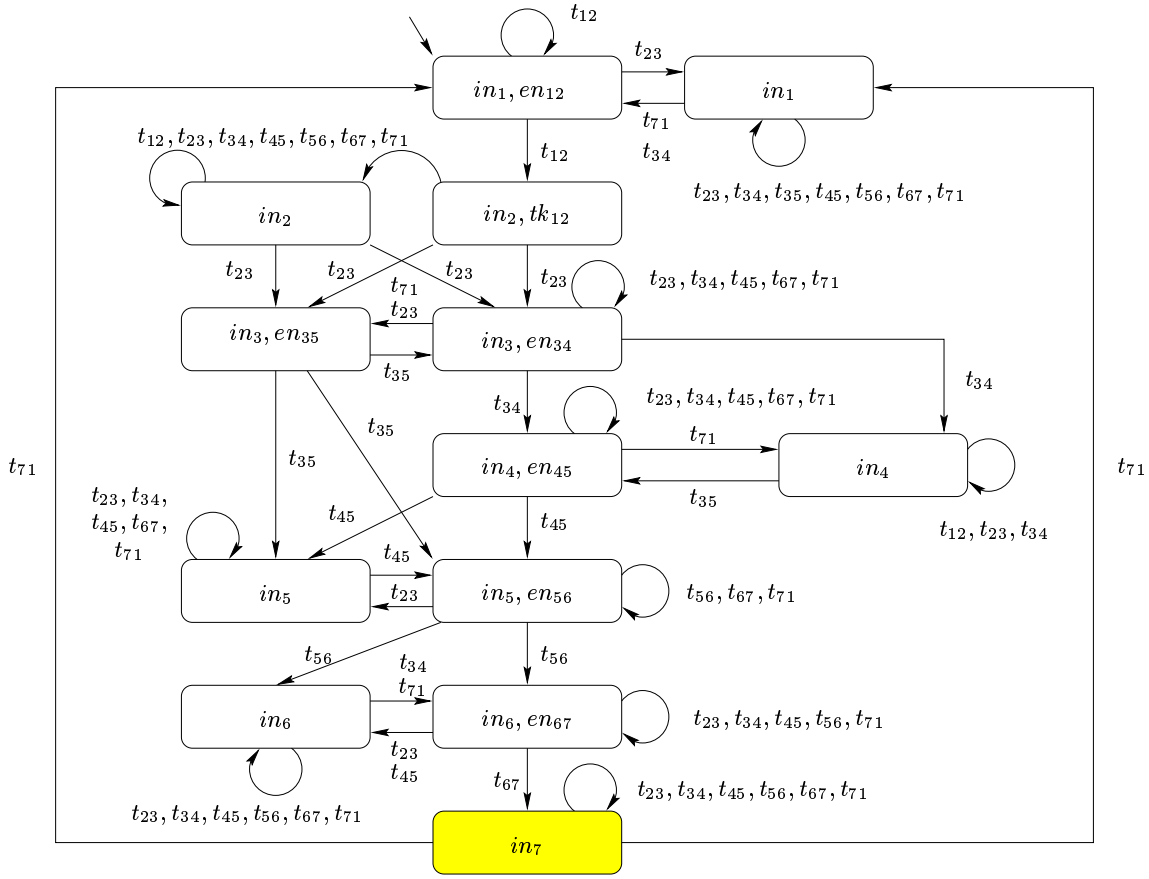


Figure 3: Abstract state graph

$$
\begin{aligned}
en_{67} &\equiv i \in \mathrm{At}_\ell_6 \wedge \forall_P\, p < i : p \in \mathrm{At}_\ell_1 \cup \mathrm{At}_\ell_2 \cup \mathrm{At}_\ell_4 \\
in_7 &\equiv i \in \mathrm{At}_\ell_7
\end{aligned}
$$

All the *taken* variables $tk_{ll'}$ but the one tk_{12} for the distinguished process are left out for readability. Since en_{23} resp. en_{71} coincide with in_2 resp. in_7, they are also omitted. All the fairness due to the marking algorithm and the weak fairness for the distinguished process can be applied on the level of edges in the abstract state graph presented in Figure 3. The states are labeled with those abstract variables which are true in that state.

With the fairness conditions due to the marking algorithm, the lifted strong fairness (en_{12}, tk_{12}), and weak fairness on all other transitions, it can be seen that the property $\Box\Diamond\ in_7$ holds in the abstract state graph. Therefore, as shown in Sections 4 and 5, we have $\forall p \in \omega : \Box\Diamond\ (p \in \mathrm{At}_\ell_7)$. This corresponds to $\Box\Diamond \mathrm{at}_\ell_7[p]$ for an arbitrary p in the original MPS.

6.2 Dijkstra's Mutual Exclusion Algorithm

Consider the following version of a mutual exclusion algorithm inspired by Dijkstra's algorithm, where each process $S(i, n)$ is given in Figure 4. In this description, we have one single global

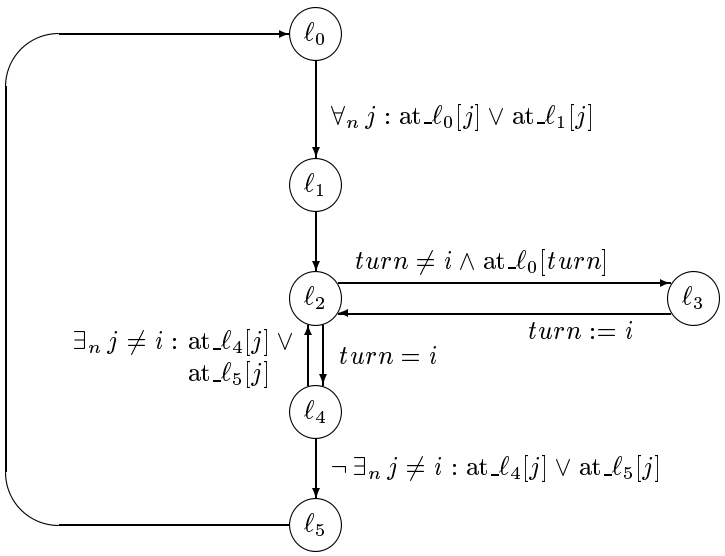


Figure 4: Dijkstra's mutual exclusion algorithm

variable *turn* that ranges over natural numbers. Although, we could have replaced by local boolean variables, we have chosen not to do so, to remain as close as possible to Dijkstra's original description. When the algorithm is translated into WS1S this variable is translated identically.

Note, that the control flow indeed can be modeled as boolean variables. In the translation this introduces the five set variables $Pc_0, \ldots, Pc_5$. As one and only non-boolean variable the *turn* variable ranging over the domain of involved process indices may be read and written by all processes.

Initially, all processes start at ℓ_0, and the *turn* variable has an arbitrary value.

In fact, the algorithm is not the original algorithm from Dijkstra since Dijkstra's algorithm does not satisfy the individual accessibility property, which we like to prove on our algorithm. To achieve individual accessibility we block processes in ℓ_0 if the 'waiting room' ℓ_2, ℓ_3, ℓ_4 is non-empty. This causes an interesting behavior in the waiting room.

For the generic abstraction α_p we also monitor the behavior of the current *turn* process, i.e., the process having the turn:

$$\begin{aligned}
\phi_i &\equiv turn \in Pc_i \quad \text{for } 0 \leq i \leq 5 \\
\psi_i &\equiv p \in Pc_i \quad \text{for } 0 \leq i \leq 5 \\
\xi_1 &\equiv Pc_5 \neq \emptyset \\
\xi_2 &\equiv \exists_P\, k : Pc_5 \subseteq \{k\} \\
\gamma &\equiv p = turn \\
\delta &\equiv Pc_4 \setminus \{turn\} \neq \emptyset
\end{aligned}$$

We leave out here the abstract variables monitoring whether p or *turn* takes a transition, but they are added to the variables given.

Using NuSMV we can prove the following LTL formula to be correct:

$$
\begin{aligned}
(\Diamond\Box\xi_1 \rightarrow \Box\Diamond tk_{50}) \quad & \wedge && (3)\\
(\Diamond\Box\delta \rightarrow \Box\Diamond(tk^o_{42} \vee tk^o_{45})) \quad & \wedge && (4)\\
(\Diamond\Box\psi_1 \rightarrow \Box\Diamond\psi_2) \quad & \wedge && (5)\\
(\Box\Diamond tk_{12} \rightarrow \Box\Diamond tk_{50}) \quad & \wedge && (6)\\
(\Box\Diamond tk_{45} \rightarrow \Box\Diamond tk_{12}) \quad & \wedge && (7)\\
(\Box\Diamond tk_{01} \rightarrow \Box\Diamond tk_{50}) \quad & \wedge && (8)\\
(\Box\Diamond tk_{32} \rightarrow \Box\Diamond tk_{23}) \quad & \wedge && (9)\\
(\Box\Diamond tk_{42} \rightarrow \Box\Diamond tk_{24}) \quad & \wedge && (10)\\
(\Box\Diamond tk^o_{42} \rightarrow \Box\Diamond tk_{32}) \quad & \wedge && (11)\\
(\Box\Diamond tk^o_{23} \rightarrow \Box\Diamond(tk_{32} \vee tk_{42} \vee tk_{12})) \quad & && (12)\\
\rightarrow \quad & \Box(\psi_1 \rightarrow \Diamond\psi_5) && (13)
\end{aligned}
$$

Fairness condition (3) is due to the requirement that the critical section has to be left eventually. The weak fairness condition on the concrete level is equivalent to a strong fairness condition, since the transition can only be disabled by taking it.

Condition (4) expresses the strong fairness for processes unequal *turn* for taking a transition leaving ℓ_4. It is the combination of two strong fairness conditions (for transition ℓ_4 to ℓ_2 resp. ℓ_5).

(5) is a weak fairness condition for process p lifted from the concrete level. Conditions (6) to (12) are generated by the marking algorithm. For example, (7) is the result of the algorithm for the set $Pc_2 \cup Pc_3 \cup Pc_4$, and (11) is the result for the set $(Pc_4 \setminus \{turn\}) \cup (Pc_2 \cap \{turn\})$.

Checking the mutual exclusion property, i.e., the formula $\Box\xi_2$, takes 3.6 sec, while checking the liveness property takes 1100 sec on a Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 440MHz) with 1GB of memory.

7 Conclusion

We presented a method for the verification of universal properties of parameterized networks. Our method is based on the transformation of an infinite family of systems into a single WS1S transition system and applying abstraction techniques on this system. To be able to prove liveness properties we presented a method to add fairness requirements to the abstract system. We have applied this method, which has been implemented in our tool PAX, to a number of parameterized protocols. To compute the finite abstract system we use MONA to decide the WS1S formulae. The following table shows the time and memory needed for the construction of some examples. We used a Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 440MHz) with 1024 MB of memory.

We are currently applying our approach to the group membership algorithm used in the Timed Triggered Architecture [KG94]. Our results on this experiment will be soon available on PAX's web page [6].

[6] http://www.informatik.uni-kiel.de/~kba/PAX

Example	Time	Memory
Szymanski	3 sec	20 MB
Dijkstra	1 min 30 sec	260 MB
Simple D.	1 min 7 sec	214 MB

Table 1: PAX construction of abstract transition systems

References

[ABJN99] P.A. Abdulla, A. Bouajjani, B. Jonsson, and M. Nilsson. Handling Global Conditions in Parameterized System Verification. In N. Halbwachs and D. Peled, editors, *CAV '99*, volume 1633 of *LNCS*, pages 134–145. Springer, 1999.

[BBLS00] K. Baukus, S. Bensalem, Y. Lakhnech, and K. Stahl. Abstracting WS1S Systems to Verify Parameterized Networks. To appear in the proceedings of TACAS 2000, 2000.

[BCG89] M.C. Browne, E.M. Clarke, and O. Grumberg. Reasoning about networks with many identical finite state processes. *Information and Computation*, 1989.

[Büc60] J.R. Büchi. Weak Second-Order Arithmetic and Finite Automata. *Z. Math. Logik Grundl. Math.*, 6:66–92, 1960.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th ACM symp. of Prog. Lang.*, pages 238–252. ACM Press, 1977.

[CGJ95] E. Clarke, O. Grumberg, and S. Jha. Verifying Parameterized Networks using Abstraction and Regular Languages. In I. Lee and S. Smolka, editors, *CONCUR '95: Concurrency Theory*, LNCS. Springer, 1995.

[CGL94] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5), 1994.

[dB80] J.W. de Bakker. *Mathematical Theory of Program Cortrectness*. Prentice-Hall, NJ, 1980.

[DGG94] D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems: Abstractions preserving ACTL*, ECTL* and CTL*. In E.-R. Olderog, editor, *Proceedings of PROCOMET '94*. North-Holland, 1994.

[Dij65] E.W. Dijkstra. Solution of a problem in concurrent programming control. *Communications of the ACM*, 8(9):569, September 1965.

[Elg61] C.C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98:21–52, 1961.

[EN96] E. A. Emerson and K. S. Namjoshi. Automatic verification of parameterized synchronous systems. In *8th Conference on Computer Aided Verification*, LNCS 1102, pages 87–98, 1996.

[GS92] S.M. German and A.P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.

[HJJ+96] J.G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, B. Paige, T. Rauhe, and A. Sandholm. Mona: Monadic Second-Order Logic in Practice. In *TACAS '95*, volume 1019 of *LNCS*. Springer, 1996.

[HLR92] N. Halbwachs, F. Lagnier, and C. Ratel. An experience in proving regular networks of processes by modular model checking. *Acta Informatica*, 22(6/7), 1992.

[JN00] B. Jonsson and M. Nilsson. Transitive closures of regular relations for verifying infinite-state systems. In S. Graf, editor, *TACAS'00*, volume To appear. Lecture Notes in Computer Science, 2000.

[KG94] H. Kopetz and G. Grünsteidl. Ttp-a protocol for fault-tolerant real-time systems. *IEEE Computer*, 1994.

[KM89] R.P. Kurshan and K. McMillan. A structural induction theorem for processes. In *ACM Symp. on Principles of Distributed Computing, Canada*, pages 239–247, Edmonton, Alberta, 1989.

[KMM+97] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic Model Checking with Rich Assertional Languages. In O. Grumberg, editor, *Proceedings of CAV '97*, volume 1256 of *LNCS*, pages 424–435. Springer, 1997.

[LGS+95] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6(1), 1995.

[LHR97] D. Lesens, N. Halbwachs, and P. Raymond. Automatic verification of parameterized linear networks of processes. In *POPL '97*, Paris, 1997.

[PS00] A. Pnueli and E. Shahar. Liveness and acceleration in parameterized verification. In E.A. Emerson and A.P. Sistla, editors, *CAV'00*, volume To appear. Lecture Notes in Computer Science, 2000.

[SG89] Z. Stadler and O. Grumberg. Network grammars, communication behaviours and automatic verification. In *Proc. Workshop on Automatic Verification Methods for Finite State Systems*, Lecture Notes in Computer Science, pages 151–165, Grenoble, France, 1989. Springer Verlag.

[Tho90] W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B: Formal Methods and Semantics*, pages 134–191. Elsevier Science Publishers B. V., 1990.

[WL89] P. Wolper and V. Lovinfosse. Verifying properties of large sets of processes with network invariants (extended abstract). In Sifakis, editor, *Workshop on Computer Aided Verification*, LNCS 407, pages 68–80, 1989.