On Efficient Data Structures for the Verification of Parameterized Synchronous Systems

Giorgio Delzanno

Dipartimento di Informatica e Scienze dell'Informazione - Università di Genova
via Dodecaneso 35, 16146 Italy
e-mail: delzanno@disi.unige.it

Abstract. We propose a fully-automatic method for checking *safety properties* of parameterized synchronous systems based on a *backward reachability* procedure working over *real arithmetics*. We consider here concurrent systems consisting of many identical (finite-state) processes and one monitor where processes may react *non-deterministically* to the messages sent by the monitor. This type of non-determinism allows us to model abstractions of situations in which processes are re-allocated according to individual properties. The resulting class of systems extend previously proposed models [GS92,EN98,EFM99]. We represent concisely collections of global states *counting* the number of processes in a given state during a run of the global system, i.e., we reason modulo *symmetries*. We use a special class of *linear arithmetic constraints* to represent collections of *global* system states. We define a *decision procedure* for checking safety properties for parameterized systems using *efficient* constraints operations defined over *real arithmetics*. The procedure can be implemented using existing *constraint-based symbolic model checkers* or tools for *program analysis* defined over *real-arithmetics* (e.g. [CH78,DP99,Hal93,HHW97]). In our experiments we have applied DMC [DP99] and HyTech [HHW97] in order to verify safety properties for abstractions of control systems and cache coherence policies taken from the literature.

1 Introduction

The verification of concurrent systems with an arbitrary number of processes is an important and challenging research goal. Concurrent systems composed by many similar finite-state processes arise in several contexts like hardware designs, communication protocols, and cache coherence policies. When applied to the verification of instances with a large number of processes, traditional finite-state verification techniques suffer from the state-explosion problem. Alternative techniques are necessary in order to reason on (possibly infinite) families of processes. *Network invariants* [KM89,WL89,LHR97], *bisimulation relations* [BCG89], *theorem proving* [LD90], *regular languages* [ABJN99,PD95], *network grammars* [CGJ97], *abstract interpretation* [LS97], *search procedures* [GS92,EN98], *well-structured systems* [AJ98] are examples of techniques used to attack different instances of the problem.

Though in general the verification problem for parameterized concurrent systems is *undecidable* [AK86], decision procedures have been discovered for the verification of subclasses where systems consist of many *identical*, *finite-state* processes. Specifically, German and Sisla [GS92] have defined a procedure to verify temporal properties for families of CCS-like processes and one monitor with point-to-point communication. In [EN98], Emerson and Namjoshi have extended the model with a global operation of *synchronization* achieved through broadcast messages. In this setting, all processes react *deterministically* to the broadcast and move to a special state. In [EFM99], Esparza, Finkel and Mayr have shown that verification of a special class of safety properties is decidable for the systems of [EN98]. This result is obtained for the model checking problem called *control state reachability* [AJ99] that consists of deciding whether a state from a given *upward-closed* set of (unsafe) states is reachable from one initial states. The verification of the MESI cache coherence protocol [Han93] shows the practical interest of the previous models [EN98].

In this paper we propose a new fully automatic method based on *constraint-based symbolic model checking* (see e.g. [ACJT96,BGP97,Bul00,DP99]) for the verification of *safety properties* of a class of parameterized synchronous systems that extends the models proposed in [GS92,EN98,EFM99]. Specifically, we consider here concurrent systems with many identical (finite-state) processes and one monitor where processes may react *non-deterministically* to the 'broadcast' messages sent by the monitor. This type of non-determinism allows us to model 'abstractions' of situations in which processes are re-allocated according to individual properties. In the paper we show that the *control state reachability* problem [AJ99] is decidable for the above mentioned class of parameterized synchronous system. This result is obtained reformulating the model checking problem for parameterized systems as a *control state reachability* problem for a special class of *infinite-state* systems with data variables of integer type, we call *re-allocation* machines. We use these machines in order to represent sets of global states modulo *symmetries*. Specifically, we use a set of *counters* (one for each process state) to keep track of the number of processes in a given state during a run of the original concurrent system. Re-allocation machines are an extension of Vector Addition Systems and Broadcast Protocols (the models underlying the systems of [GS92,EN98,EFM99]) where we add reset and *non-deterministic* transfer operations in the style of P/T nets with marking-dependent arcs [Cia94]. As we show in the paper, the control state reachability problem can be solved using the abstract backward reachability algorithm of [ACJT96,AJ99,FS99]. Furthermore, following [DEP99], in order to turn the abstract algorithm into a practical method, we propose to use a special class of *linear arithmetic constraints* we called *additive constraints* as symbolic representation of sets of states for re-allocation machines. *Variable elimination*, *satisfiability* and *entailment tests* are the operations we need to handle this symbolic representation. This way, we compact further the representation of the state-space of the original parameterized systems. In fact, note that we use constraints to implicitly represent sets of states of a re-allocation machines, and that each state of a re-allocation machine represents a set of global states of the original parameterized system.

Differently from previous approaches where constraints are implicitly interpreted over natural numbers [ACJT96,AJ99,DEP99], in order to obtain an *efficient* symbolic model procedure for the abstract algorithm of [ACJT96] we apply the following *relaxation* to the constraint operations. Everytime we need to solve an *integer* constraint problem we relax the condition that the solutions must be integral and we solve the corresponding problem over the *reals*. It is important to note that we apply this 'abstraction' *during* the analysis and not at the semantic-level of our machines (i.e. we do not change the semantic domain of our systems; instead we make the operations that handle constraints more efficient). The relaxation integer-reals is a well-established technique used in integer programming and program analysis (see e.g [BF99,Sch98,CH78,Hal93,HPR94]).

We show that the resulting symbolic procedure is an exact implementation of the original algorithm of [ACJT96] in the following sense. Whenever the unsafe states are represented as additive constraints, our reachability procedure (where the constraint operations are executed over the *reals*) always terminates and gives accurate answers to the control reachability problem for re-allocation machines. In other words, we show that the symbolic formulation of the algorithm of [ACJT96] is *robust* for this class of verification problems under the relaxation *integer-reals* of the constraint operations. In order to attack the 'symbolic' state-space explosion problem, we propose to use *acceleration* rules that can be used to dynamically compact the state-space, i.e., to generate new additive constraints from existing ones during the symbolic reachability analysis. Our results show that existing symbolic model checkers (e.g. [DP99,Hal93,HPR94,HHW97]) and tools for program analysis working on *real arithmetics* (e.g. [CH78]) can be applied with the guarantee of *termination* and *accuracy in the results* to check safety properties of parameterized systems like the one we propose and the ones in [GS92,EN98]. In practical experiments, we are currently using two of such tools, namely, DMC [DP99] (based on a constraint solver over reals) and HyTech [HHW97] (based on the efficient *polyhedra* library of Halbwachs [Hal93]), to verify safety properties of examples taken from the literature (e.g. cache-coherence protocols and control systems). The experimental results are avalailable on the web at the address '`http://www.disi.unige/person/DelzannoG/parameterized.html`'.

Plan of the paper. In Section 2 we first define the model of parameterized system we are interested. Then, we introduce the notion of *re-allocation* machine and show how to reduce parameterized reachability in the original systems to reachability in re-allocation machines. In Section 3 we define the data-structures and the operations we use to implement a constraint-based model checking procedure. In Section 4 we briefly discuss some practical experiments. Finally, in Section 5 and 6 we discuss related works and address some future directions of this research.

2 One Monitor, Many Processes

Following [GS92,PD95,EN98], we consider a concurrent system composed by many identical *finite-state* processes and by one *monitor*. In this paper we limit ourselves to specify the *communication layer* of a system. Without loss of generality, we assume that processes can communicate only with the monitor. This assumption will simplify the technical aspects of our presentation. A parameterized system $\mathcal{S}$ consists of the following components. The set Σ_P denotes the messages that processes send to the monitor. The indexed set $\Sigma_i = \Sigma_P \times \{i\}$ denotes the messages σ_i sent by process i. All processes have the same set of states Q_P. The set Σ_M denotes the messages that the monitor sends to the processes. The monitor is represented as a *deterministic* finite-state machine $\mathcal{M} = \langle Q_M, \Sigma_M \cup (\Sigma_P \times \mathbb{N}), \delta_M \rangle$, where $\delta_M : Q_M \times (\Sigma_M \cup (\Sigma_P \times \mathbb{N})) \rightarrow Q_M$ (the monitor receives messages from any process). Process i is represented as a *non-deterministic* finite state machine $\mathcal{P}_i = \langle Q_P, \Sigma_M \cup \Sigma_i, \delta_i \rangle$, where $\delta_i : Q_P \times (\Sigma_M \cup \Sigma_i) \times Q_P$. To enforce the condition that all processes behave in the same way, we require that $\delta_i(q, \sigma_m) = \delta_j(q, \sigma_m)$ and that $\delta_i(q, \sigma_i) = \delta_j(q, \sigma_j)$ for any message $\sigma_m \in \Sigma_M$, $\sigma \in \Sigma_P$, $q \in Q_P$ and any process i, j. For simplicity, we limit the *non-determinitism* of a process to the *reactions* to a monitor message $\sigma_m \in \Sigma_M$ as follows: upon reception of σ_m, the target state q' for process i may be picked up non-deterministically from a set $\Pi(\sigma_m)$ that does not depend on the current state q. More precisely, given $\sigma_m \in \Sigma_M$, let us fix the set $\Pi(\sigma_m) \subseteq Q_P$. If $q \in \Pi(\sigma_m)$ then $\delta_i(q, \sigma_m) = \{q\}$; otherwise, if $q \notin \Pi(\sigma_m)$, then either $\delta_i(q, \sigma_m) = \{q\}$ or $\delta_i(q, \sigma_m) = \Pi(\sigma_m)$. We also allow *deterministic reactions*. In this case $\delta_i(q, \sigma_m)$ is a singleton for all q (with no further restrictions).

Remark 1. By hypothesis, we cannot distinguish one process from another. However, we can use non-deterministic reactions to model *abstractions* of situations in which processes are re-allocated according to individual properties (e.g. priorities). We will clarify this point in Example 1.

We describe a system with n processes as a *global*, finite-state machine $\mathcal{M}_G = \langle Q_G, \Sigma_G, \delta_G \rangle$, where $Q_G = Q_M \times Q^n$, $\Sigma_G = \sigma_M \cup (\Sigma \times \{1, \ldots, n\})$, and $\delta_G : Q_G \times \Sigma_G \times Q_G$. The global transition relation is defined as follows. If a *process sends a message to the monitor*, i.e., $\sigma_i \in \Sigma_i$, $\langle q_i, \sigma_i, q'_i \rangle \in \delta_i$, and $\langle q, \sigma_i, q' \rangle \in \delta_M$, then $\langle \langle q, q_1, \ldots, q_n \rangle, \sigma_i, \langle q', q'_1, \ldots, q'_n \rangle \rangle \in \delta_G$ and $q'_j = q_j$ for every $j \neq i$. If the *monitor sends a message* to all processes, i.e., $\sigma \in \Sigma_M$, $\langle q, \sigma_M, q' \rangle \in \delta_M$ and $\langle q_i, \sigma_M, q'_i \rangle \in \delta_i$ for $i : 1, \ldots, n$, then $\langle \langle q, q_1, \ldots q_n \rangle, \sigma, \langle q', q'_1, \ldots, q'_n \rangle \rangle \in \delta_G$. An element of Q_G is called a *global state* (denoted as $G, G_1, \ldots$ etc). A *run* of $\mathcal{M}_G$ is a sequence of global states $G_1, G_2, \ldots, G_i, \ldots$ where $\langle G_i, \sigma, G_{i+1} \rangle \in \delta_G$ for some $\sigma \in \Sigma_G$. We write $G_1 \rightarrow^* G_2$ to denote that there exists a run starting from G_1 and reaching G_2.

Example 1. In Fig. 1, we show an abstraction of a multiprocess system with a *load balancing* monitor. Here $Q = \{req, use, high, low\}$, $\Sigma_P = \{req, release\}$, $Q_M = \{idle, busy\}$, and $\Sigma_M = \{swap\_in, swap\_out\}$. In the initial configuration of the system the processes are in state *req*, whereas the monitor is in state *idle*. When the monitor broadcasts the message *swap_out* (and moves to *busy*) all processes in the CPU are suspended. Two different priorities are assigned to the suspended processes. This is simulated through the non-deterministic reaction to *swap-out* with target states $\Pi(swap\_out) = \{high, low\}$. When the CPU is released by the monitor (through the broadcast *release*), it is assigned to processes with *high* priority. Processes with *low*-priority go back to the *request* state. This behaviour is simulated through the *deterministic* reaction to the broadcast *swap_in*.

Remark 2. Our definition of parameterized systems extends the model of German and Sisla [GS92] and the Broadcast Protocols of Emerson and Namjoshi [EN98] with a global *non-deterministic* synchronization operation.

In this paper we are interested in automatic verification methods for *safety* properties, i.e., to check that during a run of the system it is not possible to reach *inconsistent* states. In Example 1, when the monitor is busy no process can be in state *use*. Thus, all global state where *use* and *busy* occur simultaneously

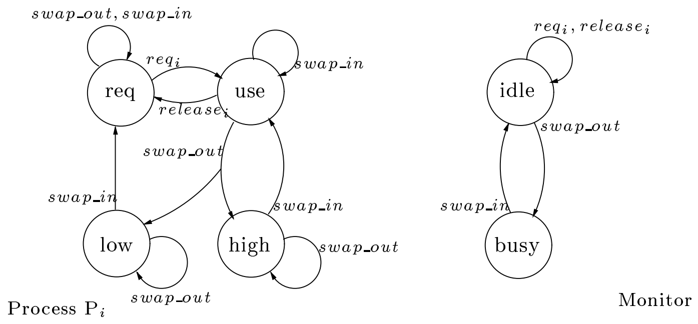

Fig. 1. Load balancing monitor.

must be considered as unsafe states. Actually, given a parameterized system we would like to check this type of safety properties independently from the parameter n=number of processes. This problem can be reduced to the following *parameterized* reachability problem.

Definition 1 (Parameterized Reachability). *Given a parameterized system $\mathcal{S}$ with n processes, let $I(n)$ and $U(n)$ be the set of initial and unsafe global states of the corresponding global machine $\mathcal{M}_G$. The parameterized reachability problem is defined as:* $\exists n \geq 1$ s.t. $G_1 \rightarrow^* G_2$ for $G_1 \in I(n)$, $G_2 \in U(n)$?

If the reachability problem is satisfied for a given m, then the safety property (whose complement is represented as $U(m)$) is violated for an instance of the system with m processes.

From Parameterized Systems to Re-allocation Machines. We show next that a parameterized system $\mathcal{S}$ can be described in terms of extensions of Vector Addition Systems similar to P/T nets with transfer and reset arcs [Cia94]. We will call these integer systems *re-allocation machines.* Intuitively, the idea is to use tuples of positive integers to count the number of processes in each state $q \in Q_G$ at any time during a run of the global machine $\mathcal{M}_G$ associated to $\mathcal{S}$. This way, we exploit the *symmetries* in the global states and reduce the state-space we need to visit. In Example 1, for $n = 3$ we use the tuple $\langle c_{req}, c_{use}, c_{high}, c_{low}, c_{idle}, c_{busy}\rangle$ with $c_{req} = 2$, $c_{use} = 1$, $c_{idle} = 1$ and all other components $= 0$, to represent all global states where one process is in state *use.*

Notation. In the following we will use $\mathbf{c}$ *to denote a vector of constants* $\langle c_1, \ldots, c_K\rangle$ *and* $\mathbf{x}$ *to denote the vector of variables* $\langle x_1, \ldots, x_K\rangle$. $\mathbf{x} \geq \mathbf{c}$ *denotes the constraint* $x_1 \geq c_1 \wedge \ldots \wedge x_K \geq c_K$ *(similarly for* $=, \leq$ *etc). Given a* $K \times K$*-matrix* A *with elements* $\{a_{i,j}\}_{i,j:1,\ldots,K}$, $A \cdot \mathbf{x} \geq \mathbf{c}$ *denotes the constraint* $a_{1,1}x_1 + \ldots + a_{1,n}x_n \geq c_1 \wedge \ldots \wedge a_{n,1}x_1 + \ldots + a_{n,n}x_n \geq c_n$.

Definition 2 (Re-allocation machine). *A re-allocation machine* $\mathcal{R}$ *is a tuple* $\langle K, C_0, \tau\rangle$, *where* $K \geq 1$ *is a natural number,* $C_0 \subseteq \mathcal{P}(\mathbb{N}^K)$ *is the initial configuration,* $\tau \subseteq \mathbb{N}^K \times \mathbb{N}^K$ *is a transition relation defined via a* set of guarded commands[1] $G(\mathbf{x}, \mathbf{x}') \rightarrow T(\mathbf{x}, \mathbf{x}')$ *with the following features. A* guard *is defined as*

$$G(\mathbf{x}, \mathbf{x}') \quad \equiv \quad D_1 \cdot \mathbf{x}' \geq D_2 \cdot \mathbf{x} \;\wedge\; \mathbf{x} \geq \mathbf{d},$$

where D_1, D_2 *are* $K \times K$ diagonal *matrices with* 0 *and* 1 *coefficients*[2], *and* $\mathbf{d}$ *is a vector of positive integers. A* transformation *is defined as*

$$T(\mathbf{x}, \mathbf{x}') \quad \equiv \quad A \cdot \mathbf{x}' = B \cdot \mathbf{x} + \mathbf{c},$$

where A *and* B *are* $K \times K$*-matrices with unite vectors as columns, and* $\mathbf{c}$ *is a vector of* K *integers. We also assume that for each guard of the form* $x'_i \geq x_i$ *in* $D_1 \cdot \mathbf{x}' \geq D_2$ *there exists an assignment of the form* $x'_i + \ldots = x_i + \ldots$ *in the corresponding row of the transition equation* $A \cdot \mathbf{x}' = B \cdot \mathbf{x} + \mathbf{c}$.

A run of $\mathcal{R}$ is a (possibly infinite) sequence of states $\mathbf{c}_1, \ldots, \mathbf{c}_i, \ldots$ where, at step i, $G_r(\mathbf{c}_i, \mathbf{c}_{i+1}) \wedge T_r(\mathbf{c}_i, \mathbf{c}_{i+1})$ evaluates to *true* for some rule r. The predecessor operator $pre : \mathcal{P}(\mathbb{N}^K) \rightsquigarrow \mathcal{P}(\mathbb{N}^K)$ associated to $\mathcal{R}$ is defined as

$$pre(S) = \{\mathbf{c} \mid \mathbf{c}\ \tau\ \mathbf{c}',\ \mathbf{c}' \in S\}.$$

[2] That is, constraints of the form $x'_{i,1} \geq x_{i,1} \wedge \ldots \wedge x'_{i,m} \geq x_{i,m}$ where $\{x_{i_1}, \ldots, x_{i,m}\} \subseteq \{x_1, \ldots, x_K\}$.

We use $pre_{|r}$ to indicate the restriction of pre to a given transition rule r in $\mathcal{R}$.

Remark 3. Re-allocation machines are extensions of Vector Addition Systems and Broadcast Protocols. In Vector Addition Systems $G(\mathbf{x}) \equiv \mathbf{x} \geq \mathbf{c}$ and $T(\mathbf{x}, \mathbf{x}') \equiv \mathbf{x}' = I \cdot \mathbf{x} + \mathbf{d}$, whereas in Broadcast Protocols $G(\mathbf{x}) \equiv \mathbf{x} \geq \mathbf{c}$ and $T(\mathbf{x}, \mathbf{x}') \equiv \mathbf{x}' = A \cdot \mathbf{x} + \mathbf{d}$, and A is a matrix with unit vectors as colunmns

The re-allocation machine $\mathcal{R}_\mathcal{S}$ associated to a parameterized system $\mathcal{S}$ with n processes is defined as follows. Let $\mathcal{M}_G$ be the global machine $\langle Q_G, \Sigma_G, \delta_G \rangle$ with $Q_G = \{q_1, \ldots, q_K\}$. $\mathcal{R}_\mathcal{S}$-states are tuples $\langle c_1, \ldots, c_K \rangle$ where $c_i \in \mathbb{N}$ keeps track of how many processes are in state q_i during a run of $\mathcal{M}_G$. An $\mathcal{R}_\mathcal{S}$-transition is defined over the tuple of variables $\langle x_1, \ldots, x_K \rangle$ (x_i is the counter for q_i) ranging over positive integers. We define them by cases depending on the type of messages. (Notes: since all processes are indentical, we consider an arbitrary process i with transition relation δ_i. Furthermore, we will omit all equalities of the form $x' = x$.)
A process sends a message to the monitor, i.e., $\langle q_p, \sigma_i, q_l \rangle \in \delta_i$ and $\langle q_r, \sigma_i, q_s \rangle \in \delta_M$ for some $q_p, q_l \in Q_P$, $q_r, q_s \in Q_M$ and $\sigma_i \in \Sigma_i$. In this case, we simply keep track of the local *synchronization* between the process and the monitor by decrementing and incrementing the counters for the corresponding starting and target states:

$$\begin{array}{lcl} G(\mathbf{x}, \mathbf{x}') & \equiv & x_p \geq 1 \ \wedge \ x_r \geq 1, \\ T(\mathbf{x}, \mathbf{x}') & \equiv & x'_p = x_p - 1 \ \wedge \ x'_l = x_l + 1 \ \wedge \ x'_r = x_r - 1 \ \wedge \ x'_s = x_s + 1. \end{array}$$

The monitor sends a broadcast, i.e., $\langle q_r, \sigma_m, q_s \rangle \in \delta_M$ for some $q_r, q_s \in Q_M$ and $\sigma_m \in \Sigma_M$. It is important to note that in the semantics of $\mathcal{S}$ we have assumed that *all processes* react simultaneously to the monitor message. We have two subcases.
(1) If the reaction of process i to σ_m is *deterministic*, then we simply re-allocate the counters of all processes as follows. (Note: if $\langle q_j, \sigma_m, q_p \rangle \in \delta_i$, then all processes in q_j must move to q_p, namely $x'_j = 0, x_p = x_j + \ldots$ etc.)

$$\begin{array}{lcl} G(\mathbf{x}, \mathbf{x}') & \equiv & x_r \geq 1, \\ T(\mathbf{x}, \mathbf{x}') & \equiv & x'_r = x_r - 1 \ \wedge \ x'_s = x_s + 1 \ \wedge \bigwedge_{q_p \in Q_P} \ x'_p = \Sigma_{q_k \in \{q_j \ \mid \ \langle q_j, \sigma_m, q_p \rangle \in \delta_i\}} \ x_k \end{array}$$

where $\Sigma_{q_i \in \emptyset} \ x_i = 0$.
(2) If the reaction is *non-deterministic*, the set of states $\Pi(\sigma_m) \subseteq Q_P$ in which all processes move when they receive σ_m is fixed a priori (by definition of $\mathcal{S}$). Before we give the $\mathcal{R}_\mathcal{S}$ rule, we define the set $\Omega(\sigma_m) = \{q_p \notin \Pi(\sigma_m) \mid \delta_i(q_p, \sigma_m) = \Pi(\sigma_m)\}$. The corresponding $\mathcal{R}$-transition is as follows. (Note: if $\delta_i(q_j, \sigma_m) = \{q_p, q_l, \ldots\}$ then all processes in q_j will move to one of $q_p, q_l, \ldots$, i.e., $x'_p + x'_l + \ldots = x_j + \ldots$).

$$\begin{array}{lcl} G(\mathbf{x}, \mathbf{x}') & \equiv & x_r \geq 1 \ \wedge \ \bigwedge_{q_j \in \Pi(\sigma_m)} \ x'_j \geq x_j, \\ T(\mathbf{x}, \mathbf{x}') & \equiv & x'_r = x_r - 1 \ \wedge \ x'_s = x_s + 1 \ \wedge \ \Sigma_{q_p \in \Pi(\sigma_M)} \ x'_p = \Sigma_{q_j \in \Omega(\sigma_m)} \ x_j \ \wedge \ \bigwedge_{q_j \in \Omega(\sigma_m)} \ x'_j = 0. \end{array}$$

The constraint in $T(\mathbf{x}, \mathbf{x}')$ in point (a) and (b) perform the necessary re-allocations of tokens. A sum in the left hand side of $=$ denotes a non-deterministic re-allocation. The guards $x'_p \geq x_p$ in (b) ensure that at least all processes with state $q_p \in \Pi(\sigma_M)$ will not change state.

Remark 4. The basic model we defined in Section 2 can be enriched easily with *internal actions* ($\langle q_r, \sigma_i, q_s \rangle \in \delta_i$ modeled as $x_r \geq 1 \wedge x'_r = x_r - 1 \wedge x'_s = x_s + 1$) and *inter-process communication* (as in the case of process-monitor communication).

Example 2. The re-allocation machine associated to the example 1 is defined in Fig. 2 (where x_s is the number of processes in state s). The guards $x'_{high} \geq x_{high}$ and $x'_{low} \geq x_{low}$ in the *swap_out* rule ensure that only the processes in x_{use} are redistributed between x_{high} and x_{low} (i.e. there is no migration of processes from state *low* to *high* or vice versa from state *high* to state *low*).

(1) $x_{req} \geq 1,\ x_{idle} \geq 1 \ \longrightarrow\ x'_{req} = x_{req} - 1,\ x'_{use} = x_{use} + 1.$

(2) $x_{use} \geq 1,\ x_{idle} \geq 1 \ \longrightarrow\ x'_{use} = x_{use} - 1,\ x'_{req} = x_{req} + 1.$

(3) $x_{idle} \geq 1,\ x'_{high} \geq x_{high},\ x'_{low} \geq x_{low} \longrightarrow$
$x'_{idle} = x_{idle} - 1,\ x'_{busy} = x_{busy} + 1,\ x'_{high} + x'_{low} = x_{high} + x_{low} + x_{use},\ x'_{use} = 0.$

(4) $x_{busy} \geq 1 \ \longrightarrow$
$x'_{busy} = x_{busy} - 1,\ x'_{idle} = x_{idle} + 1,\ x'_{high} = 0,\ x'_{low} = 0,\ x'_{use} = x_{use} + x_{high},\ x'_{req} = x_{req} + x_{low}.$

Fig. 2. Re-allocation machine for Ex. 1: (1)=req_i, (2)=$release_i$, (3)=*swap_out*, (4)=*swap_in*.

2.1 From Parameterized Reachability in $\mathcal{S}$ to Reachability in $\mathcal{R}_{\mathcal{S}}$.

Given a global state G of a parameterized system $\mathcal{S}$, let us define the mapping $\# : Q_G \rightarrow \mathbb{N}^k$ such that $\#G$ is the tuple $\langle c_1, \ldots, c_K \rangle$ where c_i=number of processes of G in state q_i. The following property relates the runs in $\mathcal{S}$ with the runs of the corresponding re-allocation machine $\mathcal{R}_{\mathcal{S}}$.

Proposition 1. $\mathcal{S}$ and $\mathcal{R}_{\mathcal{S}}$ are bisimulation equivalent.

Proof. The relation $R \subseteq Q_G \times N^K$ defined as $R(G, \mathbf{c})$ if and only if $\mathbf{c} = \#G$ is a bisimulation, i.e., for each G' s.t. $G \rightarrow G'$ there exists $\mathbf{c}'$ s.t. $\mathbf{c} \rightarrow \mathbf{c}'$ and $R(G', \mathbf{c}')$, and for each $\mathbf{c}'$ s.t. $\mathbf{c} \rightarrow \mathbf{c}'$, and there exists G' s.t. $G \rightarrow G'$ and $R(G', \mathbf{c}')$. □

Corollary 1. If there exist two global states G, G' such that $G \rightarrow^* G'$ in $\mathcal{S}$, then $\#G \rightarrow^* \#G'$ in $\mathcal{R}_{\mathcal{S}}$. Viceversa, given two $\mathcal{R}_{\mathcal{S}}$-state $\mathbf{c}, \mathbf{c}'$, if $\mathbf{c} \rightarrow \mathbf{c}'$ in $\mathcal{R}_{\mathcal{S}}$ then there exist two global states G, G' such that $\#G = \mathbf{c}$ and $\#G' = \mathbf{c}'$ and $G \rightarrow^* G'$ in $\mathcal{S}$.

It is important to note that $\mathcal{R}_{\mathcal{S}}$ gives us a description of a parameterized system $\mathcal{S}$ *independently* from the parameter n=number of processes (states in $\mathcal{R}_{\mathcal{S}}$ are tuples of length K=number of states in $\mathcal{S}$). Thus, Corollary 1 allows us to reduce the parameterized reachability problem for $\mathcal{S}$ to a reachability problem for $\mathcal{R}_{\mathcal{S}}$. We proceed as follows. We first express the parameterized initial and unsafe global states $S_o(n)$ and $S_f(n)$ via the mapping #. We obtain two set of tuples $C_0(n)$ and $C_f(n)$ that depend on n. Now, we define $C_0 = \bigcup_{n \geq 1} C_0(n)$ and $C_f = \bigcup_{n \geq 1} C_f(n)$, and try to solve the reachability problem: exist $\mathbf{c} \in C_0$, $\mathbf{c}' \in C_f$ s.t. $\mathbf{c} \rightarrow^* \mathbf{c}'$? in $\mathcal{R}_{\mathcal{S}}$.

Example 3. For a fixed n, the initial state $S_o(n)$ of the system of Example 1 is the global states with n processes in state *req* and one (the monitor) in state *idle*. Thus, $C_0(n)$ is the set of tuples with $c_{req} = n$, $c_{idle} = 1$ and $c_q = 0$ for all other states q. The system should satisfy the following invariant: when the monitor is in state *busy* there no processes can be in state *use*. In other words, $S_f(n)$ (the set of unsafe states) is the set of global states G with at least one process in *busy* and one in *use*, i.e., $C_f(n)$ is the set of tuples with $c_{use} \geq 1$, $c_{busy} \geq 1$, and arbitrary values for the remaining counters. C_0 is the set of tuples with $c_{req} \geq 1$, $c_{idle} = 1$ and $c_q = 0$ for all other states q, whereas C_f is the set of tuples with $c_{use} \geq 1$, $c_{busy} \geq 1$ and $c_q \geq 0$ for any other state q.

The counters $x_1, \ldots, x_K$ assume potentially *unbounded* values during a run of $\mathcal{R}_{\mathcal{S}}$. Thus, the set of (backward or forward) reachable states of $\mathcal{R}_{\mathcal{S}}$ is potentially *infinite*, even if the initial (unsafe) set of states is finite, i.e., $\mathcal{R}_{\mathcal{S}}$ is an *infinite-state* machine. We show next that the reachability problem for re-allocation machines can always be decided when C_f is an *upward-closed* set of tuples over $\mathbb{N}^K$. Following [ACJT96,AJ99,FS99], we call this instance of the reachability problem *control state reachability*. Note that $S \subseteq \mathbb{N}^K$ is upward-closed if for all $\mathbf{c} \in S$ if $\mathbf{c}' \geq \mathbf{c}$ then $\mathbf{c}' \in S$ (where $\geq$ is the pointwise ordering of tuples of natural numbers). As for C_f in Example 3, set of unsafe states can often be represented as upward-closed sets. This result is obtained using the general methodology to prove well-structuredness of systems given in [ACJT96,AJ99,FS99] and applied to subclass of re-allocation machines like Vector Addition Systems and broadcast protocols, e.g., in [AJ99,BM99,FS99,EFM99]

Theorem 1. The control state reachability problem is decidable for re-allocation machines.

Proof. According to the methodology proposed in [AJ99], it is enough to prove that the class of upward-closed sets over $\mathbb{N}^K$ is closed under application of *pre*. We first note that an upward-closed set can be represented as a collection of sets like $S = \{\mathbf{m} \mid \mathbf{m} \geq \mathbf{e}\}$ ($\mathbf{e}$ is a minimal point). Let us consider a transition r with $G(\mathbf{x}, \mathbf{x}') \equiv \mathbf{x} \geq \mathbf{d} \wedge D_1 \cdot \mathbf{x}' \geq D_2 \cdot \mathbf{x}$ and $T(\mathbf{x}, \mathbf{x}') \equiv A \cdot \mathbf{x}' = B \cdot \mathbf{x} + \mathbf{c}$. From Def. 2, for each guard $x' \geq x$ there exists a transformation $x' + E' = x + E$ (E', E sums of variables respectively over $\mathbf{x}'$ and $\mathbf{x}$). The guard is equivalent then to $E \geq E'$. Thus, we can write the condition $D_1 \cdot x' \geq D_2 \cdot \geq x$ (in combination with $T(\mathbf{x}, \mathbf{x}')$) as $M_1 \cdot \mathbf{x} \geq M_2 \cdot \mathbf{x}'$ where M_1, M_2 are matrices with unit vectors as columns. Then, $pre_{|r}(S)$ corresponds to the set $S' = \{\mathbf{n} \mid \mathbf{n} \geq \mathbf{d}\ \} \cap \{\mathbf{n} \mid M_1 \cdot \mathbf{n} \geq M_2 \cdot \mathbf{e}\ \wedge\ B \cdot \mathbf{n} \geq max(A \cdot \mathbf{e} - \mathbf{c}, \mathbf{0})\ \}$, where $max(\mathbf{c}, \mathbf{0}) = \langle max(c_1, 0), \ldots, max(c_K, 0)\rangle$. Since B and M_1 are matrices with unit vectors as columns, it is immediate to see that the resulting set is still expressible as *union* of upward-closed sets. □

3 Data structures for symbolic model checking

The decidabilty result for the control reachability problem allows us to consider re-allocation machines as a sort of *finite-state systems*. An abstract algorithm for backwards reachability has been given in [ACJT96,FS99]. Given a upward-closed set C_f of states, the algorithm repeatedly applies the predecessor operator until a fixpoint is reached, corresponding to the set of all predecessors of C_f. If this set contains some of the initial configurations, then C_f is reachable. Upward-closed sets equipped with the subset relation form a well-quasi ordering. This property ensures the theoretical termination of the algorithm.

Towards compact data structures. In order to turn the abstract algorithm into a practical method, it is necessary to find efficient and compact representation of upward-closed sets. In previous works integer constraint systems (for Petri Nets [AJ99,AJKP98,DEP99] and Broadcast Protocols [DEP99]) and graph-based data structures (for Petri Nets [DR00]) have been proposed as symbolic representation for upward-closed sets. Integer constraint systems (e.g. linear arithmetic constraints interpreted over the integers) provide a natural representation of the semantics of parameterized systems, whereas graph-based structures like the Sharing Trees used in [DR00] provide for a compact representation of collections of minimal points.

In this paper, as an alternative to the previous methods we apply the following *relaxation* of the constraint operations: we solve linear problems over the domain of *reals* instead that over the domain of *integers*. This way, we can exploit efficient (polynomial-time) operations to handle symbolic representation of upward-closed sets (in contrast to the exponential-time operations required by integer constraints [DEP99]). The algorithm of [ACJT96] turns out to be robust under the new interpretation of constraints. Furthermore, similarly to the approach in [DR00], the subclass of linear constraints we introduce below provides a compact representation of collections of upward-closed sets. Formally, the class $\mathcal{AD}$ of constraints we are interested is that of *additive constraints* [DEP99], i.e., linear arithmetic constraints of the following form:

$$\varphi ::= \varphi \wedge \varphi \mid x_{i_1} + \ldots + x_{i_k} \geq c,$$

where $x_1, \ldots, x_K$ are variables (ranging over positive integers or reals) and c is a positive integer. We give next the properties of additive constraints when interpreted over *integers* and then consider the case in which they are interpreted over the *reals*.

Notation: In the rest of the paper we will use the lower-case letters $\varphi, \psi, \ldots$ to denote constraints and the upper-case letters $\Psi, \Phi, \ldots$ to denote sets *(disjunctions) of constraints.*

Following [AJ99], the *denotation* of a constraint φ is defined as $[\![\varphi]\!] = \{\mathbf{t} \mid \mathbf{t} \in \mathbb{N}^K$ satisfies $\varphi\}$. The definition is extended to sets in the natural way. A constraint $\varphi = \varphi_1\ \wedge\ \ldots\ \wedge\ \varphi_k$ (φ_i atomic) is *minimal* if there are no φ_j such that $[\![\varphi_j]\!] \subseteq [\![\varphi \setminus \varphi_j]\!]$ (e.g. $x + y \geq 4 \wedge x \geq 2 \wedge y \geq 2$ is not minimal). In the rest of the paper we will mainly consider minimal constraints. We say that a constraint ψ *entails* a constraint φ, written $\varphi \sqsubseteq \psi$, if and only if $[\![\psi]\!] \subseteq [\![\varphi]\!]$. *Full entailment* bewteen two sets of $\mathcal{AD}$-constraints Φ and Ψ is defined as follows: $\Phi \sqsubseteq \Psi$ if and only if $[\![\Psi]\!] \subseteq [\![\Phi]\!]$. We give sufficient conditions for the full test via the *pairwise entailment* test defined as follows: $\Phi \preccurlyeq \Psi$ if and only if for all $\psi \in \Psi$ there exists $\varphi \in \Phi$ such that $\varphi \sqsubseteq \psi$. It is important to note that, while $\Phi \preccurlyeq \Psi$ implies $\Phi \sqsubseteq \Psi$, the reverse implication does not hold. As a counterexample, take $\Phi = \{x \geq 1, y \geq 1\}$ and $\Psi = \{x + y \geq 1\}$. Then, $\Phi \sqsubseteq \Psi$ holds but neither $x \geq 1 \sqsubseteq x + y \geq 1$ nor $y \geq 1 \sqsubseteq x + y \geq 1$ holds. An $\mathcal{AD}$-constraint is equivalent to a disjunction of constraints built up from atomic constraints without additions, i.e., having form $x_1 \geq c_1 \wedge \ldots \wedge x_n \geq c_n$, $c_i \in \mathbb{N}$. We will call this class $\mathcal{WA}$. Formally, let $dec : \mathcal{AD} \rightsquigarrow \mathcal{P}(\mathcal{WA})$ be the following map: $dec(x_1 + \ldots + x_n \geq c) = \{x_1 \geq c_1 \wedge \ldots \wedge x_n \geq c_n \mid c_1 + \ldots + c_n = c,\ c_i \in \mathbb{N}\ \}$, $dec(\varphi_1 \wedge \varphi_2) = \{\ \gamma_1 \wedge \gamma_2 \mid \gamma_1 \in dec(\varphi_1),\ \gamma_2 \in dec(\varphi_2)\ \}$. Given an $\mathcal{AD}$-constraint φ, $dec(\varphi)$ is a set of $\mathcal{WA}$-constraints such that $[\![\varphi]\!] = [\![dec(\varphi)]\!]$. The cardinality of $dec(\varphi)$ may be exponential in the size of φ (constants and number of conjuncts). As an example, note that $dec(x_1 + x_2 \geq 1 \wedge \ldots \wedge x_n + x_{n+1} \geq 1)$ for an odd natural number n, is a set of $\mathcal{WA}$-constraints of size 2^n. The constraints of this set correspond to the paths of a complete binary tree with two outcoming edges per node labeled, respectively, $x_i \geq 1$ and $x_{i+1} \geq 1$, for $i : 1, 3, 5 \ldots, n$. The constraint obtained conjoining the labels on one path is not comparable (w.r.t. $\sqsubseteq$) with the constraints associated to the other paths. Note that disjunctions of $\mathcal{WA}$-constraints are in one-to-one relation (modulo logical equivalences) with the *minimal points of upward-closed sets*. This property makes $\mathcal{AD}$-constraints good candidates to concisely represent finite unions of upward-closed sets.

Example 4. The additive constraint $x_{busy} \geq 1 \wedge x_{use} \geq 1$ represents the set C_f of Example 3. With constraints, we compact further the representation of $\mathcal{S}$ global state: an additive constraints like $x_{high} + x_{low} \geq 2$ represents the set of $\mathcal{R}_{\mathcal{S}}$-states consisting of tuples where either $c_{high} = 2$ and $c_{low} = 0$, $c_{high} = 1$ and $c_{low} = 1$ or $c_{high} = 0$ and $c_{low} = 2$. In turn, each of these states represents sets of $\mathcal{S}$ global states (2 processes are in state *high*, or in state *low*, etc).

Symbolic predecessor operator. Following [BGP97,Bul00,DP99,HHW97], we lift the predecessor operator *pre* associated to a re-allocation machine to its symbolic version **sym_pre** (such that $[\![\mathbf{sym\_pre}(\Phi)]\!] = pre([\![\Phi]\!])$) defined as follows. We first note that an $\mathcal{AD}$-constraint φ can be written as $M \cdot \mathbf{x} \geq \mathbf{e}$ where M is a matrix with 0's and 1's as coefficients. Let $\Phi = \{M_i \cdot \mathbf{x} \geq \mathbf{e}_i\}_{i \in I}$ (I=index set). Then,

$$\mathbf{sym\_pre}(\textstyle\bigvee_{i \in I} M_i \cdot \mathbf{x}' \geq \mathbf{e_i}) = \ \bigvee_{i \in I} \ \exists\, \mathbf{x}'.\ G(\mathbf{x}, \mathbf{x}') \ \wedge \ T(\mathbf{x}, \mathbf{x}') \ \wedge \ M_i \cdot \mathbf{x}' \geq \mathbf{e_i},$$

where $\mathbf{x}$ and $\mathbf{x}'$ range over *natural numbers*. As stated in the following proposition, additive constraints are still adequate to implicitly represent upward-closed sets of states of parameterized system.

Proposition 2. The class $\mathcal{AD}$ is closed under application of **sym_pre**.

Proof. Let us consider a transition r with $G(\mathbf{x}, \mathbf{x}') \equiv \mathbf{x} \geq \mathbf{d} \wedge D_1 \cdot \mathbf{x}' \geq D_2 \cdot \mathbf{x}$ and $T(\mathbf{x}, \mathbf{x}') \equiv A \cdot \mathbf{x}' = B \cdot \mathbf{x} + \mathbf{c}$. Then, $\mathbf{sym\_pre}_{|r}(M \cdot \mathbf{x}' \geq \mathbf{e})$ corresponds to (1) $\exists \mathbf{x}'.\ \mathbf{x} \geq \mathbf{d} \wedge D_1 \cdot \mathbf{x}' \geq D_2 \cdot \mathbf{x} \wedge A \cdot \mathbf{x}' = B \cdot \mathbf{x} + \mathbf{c} \wedge M \cdot \mathbf{x}' \geq \mathbf{e}$. Since A has unit-vectors as columns, $A \cdot \mathbf{x}' = B \cdot \mathbf{x} + \mathbf{c}$ can be rewritten as (2) $\mathbf{x}' = B \cdot \mathbf{x} - C \cdot \mathbf{x}' + \mathbf{c}$ for a matrix C with unit-vectors as columns. By replacement of equals, from (1) and (2) we obtain $C_1 \cdot \mathbf{x} \ \geq \ C_2 \cdot \mathbf{x}' + \mathbf{f}$, where $C_1 = M \cdot B$ and $C_2 = M \cdot C$ are matrices with 0's and 1's as coefficients (the columns might be non-unit vectors) and $\mathbf{f} = \mathbf{e} - M \cdot \mathbf{c}$. Furthermore, $G(\mathbf{x}, \mathbf{x}')$ can be equivalently written as $\mathbf{x} \geq \mathbf{d} \wedge N_1 \cdot \mathbf{x} - D_2 \cdot \mathbf{x} \geq N_2 \cdot \mathbf{x}' + \mathbf{g}$ where $N_1 = D_1 \cdot B$, $N_2 = D_1 \cdot C$ are matrices with unit-vectors as columns and $\mathbf{g} = D_1 \cdot \mathbf{c}$. From Def. 2, the $(i,j) - th$ element of N_1 is 1 iff the $(i,j) - th$ element of D_2 is 1 (in fact, for each guard $x'_i \geq x_i$ we assume that $x'_i + \ldots = x_i + \ldots$ is part of the transition). Thus, $N_1 \cdot \mathbf{x} - D_2 \cdot \mathbf{x}$ can be written as $N_3 \cdot \mathbf{x}$ where N_3 has unit-vectors as columns. Putting together the previous observations, we obtain that (1) is equivalent to $\mathbf{x} \geq \mathbf{d} \wedge \exists \mathbf{x}'.\ N_3 \cdot \mathbf{x} \geq N_2 \cdot \mathbf{x}' + \mathbf{g} \ \wedge \ M_1 \cdot \mathbf{x} \geq M_2 \cdot \mathbf{x}' + \mathbf{f}$. Now, by hypothesis, all variables range over positive values, thus the previous expression is equivalent to $\mathbf{x} \geq \mathbf{d} \wedge N_3 \cdot \mathbf{x} \geq max(\mathbf{g}, \mathbf{0}) \ \wedge \ M_1 \cdot \mathbf{x} \geq max(\mathbf{f}, \mathbf{0})$ where all matrices have unit-vectors as columns, i.e., to an additive constraint. We obtain the thesis by extending the proof to sets of constraints. □

Following [AJ99], by lifting the operations over sets of states to operations over $\mathcal{AD}$-constraints, we obtain the symbolic version Symb-Reach of the abstract algorithm of [ACJT96] shown in Fig 3. In Fig. 3, $sat(\cdot)$ denotes the satisfiability test for integer constraints. Note that Φ_o is *not* necessarily a set of additive constraints (otherwise the test is always satisfied). For instance, in Example 3 of the previous

```
Proc Symb-Reach(Φ_o, Φ_f : set of additive constraints)
   Φ := Φ_f;  Ψ := ∅;
   while Φ ≠ ∅ do
      choose φ ∈ Φ;   Φ := Φ \ {φ};
      if sat(φ ∧ Φ_o) then return reachable
      else if  ∄ψ ∈ Ψ. ψ ⊑ φ then Ψ := Ψ ∪ {φ};   Φ := Φ ∪ sym_pre(φ);
   end_while;
   return unreachable;
end.
```

Fig. 3. Symbolic reachability.

section $\Phi_o = x_{req} \geq 1 \wedge x_{idle} = 1$. The termination of the algorithm is ensured by the fact that $\langle \mathcal{AD}, \sqsubseteq \rangle$ is a well-quasi ordering as shown in the rest of the section. To prove the desired result, we need the following lemma.

Lemma 1. Let φ and ψ be two $\mathcal{AD}$-constraints in minimal form defined over the same sums, i.e., such that $\varphi = E_1 \geq d_1 \wedge \ldots \wedge E_m \geq d_m$ and $\psi = E_1 \geq c_1 \wedge \ldots \wedge E_m \geq c_m$ where $E_i \equiv x_{i,1} + \ldots + x_{1,p_i}$ is a sum over $x_1, \ldots, x_K$, and $c_i > 0$. Then, $\varphi \sqsubseteq \psi$ iff $c_i \geq d_i$ for $i : 1, \ldots, m$.

Note: the property does not hold for non-minimal constraints. E.g. $x + y \geq 1 \wedge x \geq 2 \sqsubseteq x + y \geq 2 \wedge x \geq 1$, however, $x + y \geq 1 \sqsubseteq x \geq 2$. We can prove now the following proposition.

Proposition 3. $\langle \mathcal{AD}, \sqsubseteq \rangle$ is a well-quasi-ordering.

Proof. We reason only on minimal additiv constraints. The proof is per absurdum. Let $\{\varphi_i\}_{i \geq 0}$ be an infinite sequence of $\mathcal{AD}$-constraints over the variables $x_1, \ldots, x_n$, such that for all i and $j < i$, $\varphi_j \not\sqsubseteq \varphi_i$. Since there are finitely many sums $x_{i_1} + \ldots + x_{i_k}$, there exists a subsequence $\{\varphi_{i_j}\}_{j \geq 0}$ such that: (1) for all q and all $p < q$, $\varphi_{i_p} \not\sqsubseteq \varphi_{i_q}$; (2) $\varphi_{i_j} = \wedge_{k=1}^{m} s_k \geq c_{k,j}$, i.e., all constraints in the subsequence are defined over the same sums. Together with Prop. 1, this implies that there exists an infinite sequence of tuples of natural numbers $\{t_j\}_{j \geq 0}$, $t_j = \langle c_{1,j}, \ldots, c_{m,j} \rangle$ such that, from (1), for all $j < i$, t_i is not greater than t_j w.r.t. the componentwise ordering, in contradiction to Dickson's lemma (see e.g. [AJ99]). As an aside, note that in general *well-quasi orderings* are not preserved under *finite unions* [AJ99]. □

Note that the algorithm Symb-Reach of Fig. 3 is a refinement of the following algorithm: at step 0 we set $\Theta_0 = \Phi_f, \Psi = \emptyset$; at step $i + 1$ we set $\Theta_{i+1} = \textbf{sym\_pre}(\Theta_i)$ and $\Psi = \Psi \cup \Theta_{i+1}$ and continue; we stop the loop whenever either $\Psi \preccurlyeq \Theta_{i+1}$ or there exits $\varphi \in \Psi$ such that $sat(\varphi \wedge \Phi_o)$ holds. The complexity of *each step* of the algorithm Symb-Reach is thus determined by (1) the cost of the application of **sym_pre**, (2) the cost of checking $sat(\varphi \wedge \Phi_o)$, and (3) the cost of the termination test of the algorithm based on the comparison (via the pairwise entailment test) of the sets of constraints obtained during the application of **sym_pre**. We first note that (because of the special class of constraints we consider) the application of **sym_pre** is polynomial in the size and constants occurring in the input constraints (see Prop. 2) The size (number of constraints) of the set obtained via the application of **sym_pre** to a set of additive constraints Φ is linear in the size of Φ. Due to the generality of Φ_o, the satisfiability test $sat(\varphi \wedge \Phi_o)$ may be exponential (the cost of solving an integer linear problem). [DEP99] Concerning the termination test, as shown in [DEP99] checking $\varphi \sqsubseteq \psi$ is co-NP hard in the size of φ and ψ. Entailment is polynomial for $\mathcal{AD}$-constraints in *special normal form* like whenever *every single constraint has no repeated occurences of the same variable* [DEP99]. However, the normalization may be exponential in the size (and constants) of the original $\mathcal{AD}$-constraints. (unless we consider subclasses of re-allocation machines with weaker guards and transitions as in [DEP99]). As an alternative to previous approaches, in this paper we apply a relaxation of the constraint operations.

Towards efficient operations. Following techniques from *integer linear programming*, in order to reduce the complexity of the manipulation of $\mathcal{AD}$-constraints, everytime we need to solve a system of inequalities $A \cdot \mathbf{x} \leq \mathbf{b}$ we can relax the condition that $\mathbf{x}$ is a vector of positive *integers* and look

```
Proc Symb-Reach-over-ℝ(Φ_o, Φ_f : set of additive constraints)
  Φ := Φ_f;  Ψ := ∅;
  while Φ ≠ ∅ do
    choose φ ∈ Φ;   Φ := Φ \ {φ};
    if sat_ℝ(φ ∧ Φ_o) then return reachable
    else if  ∄ψ ∈ Ψ. ψ ⊑_ℝ φ then Ψ := Ψ ∪ {φ};   Φ := Φ ∪ sym_pre_ℝ(φ);
  end_while;
  return unreachable;
end.
```

Fig. 4. Symbolic reachability.

for a *real* solution of the corresponding linear problem. This way, we obtain an *approximation* of the original set of integer solutions that, however, can be computed in polynomial time [Sch98]. Formally, the relaxation is defined as follows. Given a constraint φ, we define $[\![\varphi]\!]_{\mathbb{R}}$ as the set of real solutions $\{\mathbf{c} \in \mathbb{R}_+ \mid \mathbf{c}\ \textit{satisfies}\ \varphi\ \}$. The entailment relation over $\mathbb{R}_+$ is defined then as $\varphi \sqsubseteq_{\mathbb{R}} \psi$ if and only if $[\![\psi]\!]_{\mathbb{R}} \subseteq [\![\varphi]\!]_{\mathbb{R}}$. Full entailment over $\mathbb{R}_+$ is defined as $\Phi \sqsubseteq_{\mathbb{R}} \Psi$ if and only if $[\![\Psi]\!]_{\mathbb{R}} \subseteq [\![\Phi]\!]_{\mathbb{R}}$; pairwise entailment as $\Phi \preccurlyeq_{\mathbb{R}} \Psi$ if and only if for all $\psi \in \Psi$ there exist $\phi \in \Phi$ s.t. $\phi \sqsubseteq_{\mathbb{R}} \psi$. When applying the above relaxation to the symbolic predecessor operator, we obtain:

$$\mathbf{sym\_pre}_{\mathbb{R}}(\textstyle\bigvee_{i \in I} M_i \cdot \mathbf{x}' \geq \mathbf{e_i}) = \ \bigvee_{i \in I} \ \ \exists_{\mathbb{R}}\, \mathbf{x}'.\ G(\mathbf{x}, \mathbf{x}') \ \wedge \ T(\mathbf{x}', \mathbf{x}) \ \wedge \ M_i \cdot \mathbf{x}' \geq \mathbf{e_i},$$

where $\mathbf{x}$ and $\mathbf{x}'$ range over *positive real numbers*, and $\exists_{\mathbb{R}}$ indicates that elimination of the variables in $\mathbf{x}'$ must be performed over $\mathbb{R}_+$. It is interesting to note that, whenever we consider sets of $\mathcal{AD}$-constraints, an application of $\mathbf{sym\_pre}_{\mathbb{R}}$ does not lose precision w.r.t. **sym_pre**. In fact, the following property holds.

Proposition 4. The class $\mathcal{AD}$ of additive constraints is closed under application of $\mathbf{sym\_pre}_{\mathbb{R}}$.

Proof. The result follows by noting that to compute $\mathbf{sym\_pre}(\Phi)$ in Prop. 2 we actually used the Fourier-Motzkin elimination algorithm for constraints over $\mathbb{R}_+$. In fact, we basically apply a replacement by equals and then remove variables that do not occur on the left hand side of a $\geq$. Thus, the constraints obtained as the result of $\mathbf{sym\_pre}_{\mathbb{R}}(\Phi)$ are the same as the result of $\mathbf{sym\_pre}(\Phi)$. □

As a consequence, the following property holds (note: $[\![\Phi]\!]$ denotes the set *integer* solutions of Φ).

Corollary 2. Given a set Φ of *additive* constraints, $pre([\![\Phi]\!]) = [\![\mathbf{sym\_pre}_{\mathbb{R}}(\Phi)]\!]$.

On the basis of the newly defined operators we obtain the implementation of the algorithm of [ACJT96] shown in Fig. 4. To prove that the algorithm of [ACJT96] is *robust* when lifting the domain of interpretation of $\mathcal{AD}$-constraint from $\mathbb{N}$ to $\mathbb{R}_+$, it remains to show that Symb-Reach-over-$\mathbb{R}$ terminates on input formed by additive constraints. This is a consequence of the following proposition.

Proposition 5. $\langle \mathcal{AD}, \sqsubseteq_{\mathbb{R}} \rangle$ is a well-quasi-ordering.

Proof. We first note that Lemma 1 holds even when constraints are interpreted over $\mathbb{R}_+$. Thus, we can apply the same argument used in the proof of Proposition 3. □

Let us now study the complexity of the newly obtained algorithm and compare it with that of the procedure Symb-Reach of Fig. 3. Specifically, (1) we study the complexity of $\mathbf{sym\_pre}_{\mathbb{R}}$; (2) we define a class of initial conditions that allows us to obtain an exact implementation of the algorithm [ACJT96]; (3) we study the complexity of the fixpoint test. Prop. 4 shows that, in our setting, *variable elimination* can be equivalently computed over the reals and over the integers. Thus, in both cases the size of intermediate results will remain polynomial in the size of the set of constraints compute at the previous steps. Let us consider now the 'reachability' test $sat_{\mathbb{R}}(\varphi \wedge \Phi_0)$. As mentioned in the previous sections, in our model the initial set of states is not necessarily an upward-closed sets of states. In order to ensure that the algorithm $\mathbf{sym\_pre}_{\mathbb{R}}$ gives only accurate answers for the control reachability problem we must find a class of constraints that (a) is powerful enough to express interesting *initial sets of states* for parameterized systems; (b) when conjoined with $\mathcal{AD}$-constraints the resulting constraint is satisfiable over $\mathbb{N}$ iff it is satisfiable over $\mathbb{R}_+$. For the sake of this paper, we consider the class of $\mathcal{IC}$-onstraints (*initial condition*) defined as follows.

$$\varphi ::= \varphi \wedge \varphi \mid x_i = c_i \mid \psi,$$

where c_i is a positive integer, and ψ is an $\mathcal{AD}$-constraint. It is easy to check that an $\mathcal{IC}$-constraint φ is satisfiable in $\mathbb{N}$ if and only it is satisfiable in $\mathbb{R}_+$. Furthermore, we can use $\mathcal{IC}$-constraints to express parameterized initial configuration as in the example of Section 2. Let us consider now the termination test. The main advantage of the new procedure is that the entailment test can be performed in *polynomial* time. In fact, note that $\phi \sqsubseteq_{\mathbb{R}} (\psi_1 \wedge \psi_2)$ holds if and only if $\phi \wedge \neg\psi_1$ and $\phi \wedge \neg\psi_2$ are not satisfiable. Thus, the entailment test can be reduced to a (polynomial) number of linear problems.

Following from the previous observations, we have the following proposition.

Theorem 2. Symb-Reach-over-$\mathbb{R}(\Phi_o, \Phi_f)$ solves the control reachability problem for a given re-allocation machine $\mathcal{R}$ wrt. the set Φ_o of $\mathcal{IC}$-constraints (initial conditions), and the set Φ_f of $\mathcal{AD}$-constraints (unsafe states). Furthermore, each step of the procedure is polynomial in the size of the set of constraints computed in the previous steps.

A last remark on the termination test. As in the integer case, pairwise entailment $\Phi \preccurlyeq_{\mathbb{R}} \Psi$ (used in the procedure Symb-Reach-over-$\mathbb{R}$) gives only sufficient conditions for the corresponding full test $\Phi \sqsubseteq_{\mathbb{R}} \Psi$ (take the same example used for $\sqsubseteq$ and $\preccurlyeq$). Thus, if polynomial, $\Phi \sqsubseteq_{\mathbb{R}} \Psi$ would give better sufficient conditions to test $\Phi \sqsubseteq \Psi$ than $\Phi \preccurlyeq_{\mathbb{R}} \Psi$. (Note that, $\Phi \sqsubseteq \Psi$ does not imply $\Phi \sqsubseteq_{\mathbb{R}} \Psi$, as it can be seen by taking $\Phi = \{x \geq 1, y \geq 1\}$ and $\Psi = \{x + y \geq 1\}$). Unfortunately, the following proposition shows that the full test over $\mathbb{R}_+$ is exponential.

Proposition 6. Checking $\Phi \sqsubseteq_{\mathbb{R}} \Psi$ is co-NP hard in the size of the sets of $\mathcal{AD}$-constraints Φ and Ψ.

Proof. The proof given in [DEP99] for the integer case does not apply to this case (it makes a strong use of the hypothesis that variables assume integer values). The proof of [Sri93] for co-NP hardness of entailment for sets of linear arithmetic constraints makes use of constraints of the form $X_i \leq 0$ (outside $\mathcal{AD}$). We prove the proposition via a direct reduction from *taulology* (co-NP complete): given a propositional formula F in disjunctive normal form over the variables $p_1, \ldots, p_n$, is F true for every possible assignment? We encode this problem as follows. We assume that each literal always occurs in one of the following forms: p_i, $\neg p_i$, $\tilde{p_i}$, where the latter stands for 'any value' fo p. For each symbol p_i, we introduce two (real-valued) variables x_i, y_i. We will use them to encode truth value assignments to p_i as half-planes, namely, $p_i = true$ becomes $x_i \geq 1 \wedge y_i \geq 0$, and $p_i = false$ becomes $x_i \geq 0 \wedge y_i \geq 1$ (note: the two half-planes are not in the containment relation). We use the half-plane $x_i + y_i \geq 2$ (contained in the other two) to encode *any value* for p_i. The constraint $\varphi(F)$ is built by induction on F as follows: $\varphi(F_1 \wedge F_2) = \varphi(F_1) \wedge \varphi(F_2)$, $\varphi(F_1 \vee F_2) = \varphi(F_1) \vee \varphi(F_2)$, $\varphi(p_i) = (x_i \geq 1 \wedge y_i \geq 0)$, $\varphi(\neg p_i) = (x_i \geq 0 \wedge y_i \geq 1)$, $\varphi(\tilde{p_i}) = x_i + y_i \geq 2$. Note that $\varphi(true) = \bigwedge_i x_i + y_i \geq 2$. The encoding gives us a cosntraints polynomial in the size of F and such that F is a tautology iff $\varphi(F) \sqsubseteq_{\mathbb{R}} \varphi(true)$. □

In practical applications the 'symbolic' state-space (in our setting consisting of a large disjunction of $\mathcal{AD}$-constraints) may become very large. In order to face this form of 'symbolic state-explosion problem' it can be useful to define operations to compact the state-space during the execution of Symb-Reach-over-$\mathbb{R}$.

Accelerations. Let us start from an example. It is easy to check that $\mathcal{WA}$-constraints (defined at the beginning of Section 3) are closed under application of the operator $\mathbf{sym\_pre}_{\mathbb{R}}$ associated to a Vector Addition System (i.e. a Petri Net, see e.g. [BF99]). This means that for Petri Nets whenever Φ_f is a set of $\mathcal{WA}$-constraints there will be no $\mathcal{AD}$-constraints in the intermediate results of the procedure Symb-Reach-over-$\mathbb{R}$, causing a potential explosion of the state space (see e.g. [DEP99]). In this section we introduce *acceleration* rules that can be used to *generate* new $\mathcal{AD}$-constraints from an existing set of $\mathcal{AD}$-constraints during the execution of Symb-Reach-over-$\mathbb{R}$. The acceleration rule returns accurate results and preserves the termination of the algorithm Symb-Reach-over-$\mathbb{R}$. Technically, the rule is a

```
function Compact (φ: AD-constraint, r: transition rule): AD-constraint
 begin
  if φ = E1 ≥ c1 ∧ E2 ≥ c2 ∧ ψ and var(E1) ∩ var(E2) = ∅
    and sym_pre_R|r(φ) = E1 ≥ d1 ∧ E2 ≥ d2 ∧ ψ
    and there exists γ  s.t.  sym_pre^k_R|r(ψ) ⊑_R γ for any k
  then
    if d1 = c1 + 1, d2 = c2 − 1  return E1 + E2 ≥ c1 + c2 ∧ E2 ≥ c2 ∧ γ;
    if d1 = c1 − 1, d2 = c2 + 1  return E1 + E2 ≥ c1 + c2 ∧ E1 ≥ c1 ∧ γ;
  return φ
 end
```

Fig. 5. Acceleration for $\mathcal{AD}$-constraints.

sort of restricted convex-hull operator that, when applied, generates $\mathcal{AD}$-constraints only. The rule is applied to a single constraint computed during the execution of Symb-Reach and to a transition rule that can be part of the original system or that can be inferred by means of techniques like *fusion* (see e.g. [FO97]) or simply by unfolding the original rules (a *meta transition* in the terminology of [BW98]) The acceleration rule is defined in Fig. 5, where r denotes a meta-transition and $\mathbf{sym\_pre}^k_{\mathbb{R}|r}(r)(\varphi)$ denotes k applications of the predecessor operator (restricted to r) to a constraint φ. The following proposition states the *correctness* and *accuracy* of the rule.

Proposition 7. Let φ be an $\mathcal{AD}$-constraint then $\bigvee_{k\geq 0} \mathbf{sym\_pre}^k_{\mathbb{R}|r}(\varphi) \sqsubseteq_{\mathbb{R}} \text{Compact}(\varphi, r)$.

Note that the condition $\mathbf{sym\_pre}^k_{\mathbb{R}|r}(\psi) \sqsubseteq_{\mathbb{R}} \gamma$ in Fig. 5 is satisfied, e.g., if $\gamma = \psi$ and ψ is invariant under application of the predecessor operator, i.e., $\mathbf{sym\_pre}_{\mathbb{R}|r}(\psi) \equiv \psi$, The acceleration rule can be embedded in the procedure Symb-Reach-over-$\mathbb{R}$ either by statically deriving *meta-transitions* as in [BW98] or by *dynamically* inferring *loops* as in DMC [DP99]. Having fixed the rule r, 'Compact' can be applied everytime a new constraint has to be added to the visited states. Determining the candidate conjuncts $E_1 \geq c_1$ and $E_2 \geq c_2$ may require a quadratic number of comparisons (thus the more compact the representation is the less comparisons are required). Note that the computation of the *convex-hull* of a given set of constraints, besides being a costly operation, may return an over-approximation of the set of reachable states (i.e., it can be used when precision is not the main issue), whereas Prop. 7 ensures that the newly obtained procedure gives accurate results only. The termination of the modified algorithm is guaranteed by the fact that 'Compact' produces $\mathcal{AD}$-constraints only and by Prop. 5. Similar techniques are discussed in [CH78,HH95,LHR97].

4 Experimental results

The verification algorithm based on real arithmetics of Fig. 4 is basically the *backward reachability* procedure implemented in existing symbolic model checkers like e.g. [DP99,Hal93,HPR94,HHW97], and methods for program analysis like e.g.[CH78]. We have applied DMC [DP99], based on the real constraint solver of Holzbaur [Hol95] (where we implemented the accelerations mentioned in the previous section), and HyTech [HHW97], based on Halbwachs' polyhedra library [Hal93], on several examples of parameterized systems and integer-valued systems that belong to the classes of systems we have defined in this paper (see Section 2 and Def. 2). [3] For instance, we automatically check safety properties for parameterized versions of the *Load Balancing Monitor* of Example 1, the *Write-once* and the *Synapse N+1* cache coherence protocols [Han93], and the *Central Server System* of [ABC+95]. A detailed description of these results is available on the web at the address

`http://www.disi.unige.it/person/DelzannoG/parameterized.html`.

The method works well for examples where the intermediate results contain many additive constraints (e.g., when the transformations are more complicated than simple increments/decrements of variables).

[3] HyTech and DMC allow transitions of the form $x' + y' + \ldots = \ldots$.

For 'pathologic' examples like Petri Nets it is necessary to go one step further and represent 'constraints' using techniques like the one discussed in [ST99,DR00]. We have also applied the same method (i.e. real-based tools) for a different class of integer systems that extend Broadcast Protocols with more powerful guards (but with the same transformations as Broadcast Protocols) for which the decidability of the control state reachability problem does not hold any more. These experiments are treated in a separate paper [Del00].

5 Related Works

As mentioned in the introduction, our method is inspired to the previous works [GS92,EN98,EFM99]. The decision procedure for families of asynchronous processes of [GS92] is based on that for forward reachability of Vector Addition Systems (with state). The verification technique proposed by Emerson and Namjoshi in [EN98] is an extension of the *covering graph construction* of [KM69] (an sort of forward reachability). For broadcast protocols (a subclass of re-allocation machines), this construction is not guaranteed to terminate [EFM99]. In [DEP99], Delzanno, Esparza and Podelski propose to use additive constraints as a symbolic representation of upward-closed sets. However, they interpret the constraints over *integers* and prove that operatione like *entailment* may be exponential in this case. In contrast, in this paper we apply a relaxation of the constraint operations from integer to reals and prove that the new symbolic procedure still *decides* the control reachability problem for a class of systems that is more general than that studied in [DEP99]. In [Del00], we apply constraint-based methods for a different extension of broadcast protocols where guards may contain *tests for zero* or constants (transformations are as for broadcast protocols, see Remark 3): in this case the termination and precision of the analysis is guaranteed only for subclasses. Automated generation of abstract transition graphs for parameterized systems has been studied in [LS97]. In [LS97], Lesens and Saïdi compute *abstract networks* using *counters* to keep track of the number of processes in a given state. Diffently from our approach, they consider only linear transformations (as in Broadcast Protocols); they do not give decision procedures for subclasses of the resulting systems. Bérard and Friboug in [BF99] apply symbolic model checking over reals (namely, HyTech) for forward analysis (without guarantee of termination) of Petri Nets (Vector Addition Systems) and show that computing *pre* on sets of *difference constraints* returns accurate results. In this paper we consider extensions of Petri Nets, a different class of constraints (additive constraints) and show that backward analysis always terminates in this case. Relaxation methods (abstractions with polyhedra) for integer or hybrid systems have been studid in [CH78,DP99,Hal93,HPR94,HH95,HHW97]. Several other approaches exist to attack the verification problem of parameterized *concurrent systems*. Among semi-automatic methods we mention [BCG89,CGJ97,KM89,LD90,WL89]. Automated generation of invariants has been studied, e.g., in [CGJ97,LHR97]. The theory of well-quasi orderings has been applied to networks of timed systems in [AJ98].

6 Conclusions

We have proposed a methodology to verify safety properties for parameterized system with many identical processes based on the following points. We first reduce the state-space of the original system by representing *global states* modulo *symmetries*. The systen we obtain can be represented via infinite-state machines whose states are tuples of (potentially unbounded integers); each tuple represents a collection of global states. We reduce further the state-space by representing collection of tuples via linear arithmetic constraints whose 'natural' (i.e. induced by the semantics of re-allocation machines) interpretation is the domain of positive integers. At this point, we obtain efficient verification procedures by executing the analysis over the reals, i.e., by means of the relaxation integer-reals applied to the constraint operations. Relaxation of integer problems into linear problems is typical of integer programming and program analysis [CH78,Sch98]. For the class of systems we consider in this paper, we obtain the following nice property: the procedure working over the reals is a decision procedure for the problem originally formulated over the integers, i.e., the relaxation preserves the termination and the accuracy of the method. We are currently working on application of this methodology (even for more general classes where termination is no more guaranteed [Del00]) for the verification of practical examples of parameterized systems like communication and cache coherence protocols. It would be important to a make further step in the direction of efficient and compact manipulation of constraints, e.g., to handle examples like Petri Nets where, due to the dimension of the disjunctions generated during the analysis, even efficient constraint libraries like [Hal93] (used by HyTech) may get into troubles. Recent works like Interval Decision Diagrams [ST99], and Sharing Trees [DR00] indicate possible BDD-like structures for the representation of subclasses of linear constraints. It would be interesting to extend them to consider constraints over reals like the one we study in this paper.

Acknoledgments. We would like to thank Parosh A. Abdulla and Howard Wong-Toi for their suggestions.

References

[ACJT96] P. A. Abdulla, K. Cerāns, B. Jonsson and Y.-K. Tsay. General Decidability Theorems for Infinite-State Systems. In *Proc. 10th Symp. on Logic in Computer Science (LICS'96)*, pp. 313–321, 1996.

[AJ99] P. A. Abdulla, and B. Jonsson. Ensuring Completeness of Symbolic Verification Methods for Infinite-State Systems. *Theoretical Computer Science*, 1999. To appear.

[ABJN99] P. A. Abdulla, A. Bouajjani, B. Jonsson, and M. Nilsson. Handling Global Conditions in Parameterized System Verification. In *Proc. 11th Conf. on Computer Aided Verification (CAV'99)*, LNCS 1633, pp. 134-145, 1999.

[AJ98] P. A. Abdulla and B. Jonsson. Verifying Networks of Timed Processes. In *Proc. 4th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98)*, LNCS 1384, pp. 298-312, 1998.

[AJKP98] P. A. Abdulla, B. Jonsson, M. Kindahl, and D. Peled. A General Approach to Partial Order Reductions in Symbolic Verification. In *Proc. 10th Conf. on Computer Aided Verification (CAV'98)*, *LNCS* 1427, pp. 379–390. Springer, 1998.

[ABC+95] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* Series in Parallel Computing. John Wiley & Sons, 1995.

[AK86] K. Apt and D. Kozen. Limits for Automatic Verification of Finite-state Concurrent Systems. Information Processing Letters 15, pp. 307-309, 1986.

[BCG89] M. C. Browne, E. M. Clarke, O. Grumberg. Reasoning about Networks with Many Identical Finite State Processes. Information and Computation 81(1): 13–31, 1989.

[BF99] B. Bérard, and L. Fribourg. Reachability analysis of (timed) Petri nets using real arithmetic. In *Proc. 10th Int. Conf. on Concurrency Theory (CONCUR'99)*, LNCS 1664, pp. 178-193.

[BW98] B. Boigelot and P. Wolper. Verifying Systems with Infinite but Regular State Space. In *Proc. of the 10th Conf. on Computer Aided Verification (CAV'98)*, *LNCS* 1427, pp. 88–97. Springer, 1998.

[BM99] A. Bouajjani and R. Mayr. Model Checking Lossy Vector Addition Systems. In *Proc. 16th Annual Symp. on Theoretical Aspects of Computer Science (STACS'99)*, LNCS 1563, pp. 323–333. Springer, 1999.

[Bul00] T. Bultan. BDD vs. Constraint-Based Model Checking: An Experimental Evaluation for Asynchronous Concurrent Systems. To appear in *Proceedings of TACAS 2000*, September 2000.

[BGP97] T. Bultan, R. Gerber, and W. Pugh. Symbolic Model Checking of Infinite-state Systems using Presburger Arithmetics. In *Proc. 9th Conf. on Computer Aided Verification (CAV'97)*, LNCS 1254, pp. 400–411. Springer, 1997.

[Cia94] G. Ciardo. Petri nets with marking-dependent arc multiplicity: properties and analysis. In *Proc. 15th Int. Conf. on Application and Theory of Petri Nets*, LNCS 815, pp. 179-198. Springer, 1994.

[CGJ97] E. Clarke, O. Grumberg, S. Jha. Verifying Parameterized Networks. TOPLAS 19(5): 726-750 (1997).

[CH78] P. Cousot and N. Halbwachs. Automatic Discovery of Linear Restraints among Variables of a Program. In *Proc. 5th Annual Symp. on Principles of Programming Languages (POPL'78)*, pp. 84–96, 1978.

[Del00] G. Delzanno. Automated Verification of Parameterized Cache Coherence Protocols. Technical Report DISI-TR-00-1, DISI, University of Genova, January 2000. Available at the address http://www.disi.unige.it/person/DelzannoG/papers.

[DEP99] G. Delzanno, J. Esparza, and A. Podelski. Constraint-based Analysis of Broadcast Protocols. In *Proc. of the Annual Conf. of the European Association for Computer Science Logic (CSL'99)*, LNCS 1683, pag. 50–66. Springer, 1999.

[DP99] G. Delzanno and A. Podelski, Model Checking in CLP. In *Proc. 5th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99)*, LNCS 1579, pp. 223–239. Springer, 1999. An extended version is available at the address http://www.disi.unige.it/person/DelzannoG/papers.

[DR00] G. Delzanno and J. F. Raskin. Symbolic Representation of Upward-Closed Sets. To appear in *Proc. of TACAS 2000*, September 2000. Also technical report UCB/CSD-99-1705, Computer Science Division, University of Berkely, October 1999.

[EN96] E. A. Emerson and K. S. Namjoshi. Automatic Verification of Parameterized Synchronous Systems. In *Proc. 8th Conf. on Computer Aided Verification (CAV'96)*, *LNCS* 1102, pp. 87–98. Springer, 1996.

[EN98] E. A. Emerson and K. S. Namjoshi. On Model Checking for Non-deterministic Infinite-state Systems. In *Proc. 13th Annual Symp. on Logic in Computer Science (LICS '98)*, pp. 70–80, 1998.

[EN98b] E. A. Emerson and K. S. Namjoshi. Verification of Parameterized Bus Arbitration Protocol, In *Proc. 10th Conf. on Computer Aided Verification (CAV'98)*, *LNCS* 1427, pp. 452–463. Springer, 1998.

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the Verification of Broadcast Protocols. In *Proc. 14th Annual Symp. on Logic in Computer Science (LICS'99)*, pp. 352–359, 1999.

[FS99] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! To appear in *Theoretical Computer Science*, 1999. Also Technical Report LSV-98-4, Laboratoire Spécification et Vérification, Ecole Normale Supérieure de Cachan, April 1998.

[FO97] L. Fribourg and H. Olsen. A Decompositional Approach for Computing Least Fixed Point of Datalog Programs with Z-counters. *Journal of Constraints*, 2(3-4):305–336, 1997.

[GS92] S. M. German, A. P. Sistla. Reasoning about Systems with Many Processes. JACM 39(3): 675-735 (1992)

[GS97] S. Graf and H. Saïdi. Construction of Abstract State Graphs with PVS. In *Proc. 9th Conf. on Computer Aided Verification (CAV'97)*, LNCS 1254, pp. 72-83. Springer, 1997.

[Hal93] N. Halbwachs. Delay analysis in synchronous programs. In *Proc. 5th Conf. on Computer-Aided Verification (CAV'93)*, LNCS 697, pp. 333-346. Springer, 1993.

[HPR94] N. Halbwachs and Y.-E. Proy and P. Raymond. Verification of Linear Hybrid Systems by Means of Convex Approximations. In *Proc. Static Analysis Symposium (SAS'94)*, LNCS 864, pp. 223-237. Springer, 1994

[Han93] J. Handy. The Cache Memory Book. Academic Press, 1993.

[HHW97] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: a Model Checker for Hybrid Systems. In *Proc. 9th Conf. on Computer Aided Verification (CAV'97)*, LNCS 1254, pp. 460–463. Springer, 1997.

[HH95] T. A. Henzinger, P.-H. Ho. A Note on Abstract-interpretation Strategies for Hybrid Automata. In *Proc. of Hybrid Systems II*, LNCS 999, pp. 252-264. Springer, 1995.

[Hol95] C. Holzbaur. OFAI CLP(Q,R), Manual, Edition 1.3.3. Technical Report TR-95-09, Austrian Research Institute for Artificial Intelligence, Vienna, 1995.

[KM69] R. M. Karp and R. E. Miller. Parallel Program Schemata. *Journal of Computer and System Sciences*, 3, pp. 147-195, 1969.

[KM89] R. P. Kurshan and K. McMillan. A Structural Induction Theorem for Processes. In *Proc. 8th ACM Symp. on Principles of Distributed Computing*, pp. 239-247, 1989.

[LHR97] D. Lesens and N. Halbwachs and P. Raymond. Automatic Verification of Parameterized Linear Networks of Processes. In *Proc. 24th ACM Symp. on Principles of Programming Languages (POPL'97)*, pp. 346-357, 1997.

[LS97] D. Lesens and H. Saidi. Automatic Verification of Parameterized Networks of Processes by Abstraction. In *Proc. Int. Workshop on Verification Infinite State Systems (INFINITY'97)*, 1997.

[LD90] P. Loewenstein and D. Dill. Verification of Multiprocessor Cache Protocols using Simulation Relations and Higher-Order Logics. In *Proc. 2nd Int. Workshop on Computer Aided Verification (CAV'90)*, LNCS 531, pp. 302–311. Springer, 1990.

[PD95] F. Pong, M. Dubois. A New Approach for the Verification of Cache Coherence Protocols. IEEE Transactions on Parallel and Distributed Systems 6(8), August 1995.

[Sch98] A. Schrijver. Theory of Linear and Integer Programming. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley and Sons, 1998.

[Sri93] D. Srivistava. Subsumption and Indexing in Constraint Query Languages with Linear Arithmetic Constraints. Annals of Mathematics and Artificial Intelligence, 8(3-4): 315-343, 1993.

[ST99] K. Strehl, L. Thiele. Interval Diagram Techniques For Symbolic Model Checking of Petri Nets. In *Proc. of the Design, Automation and Test in Europe Conf. (DATE'99)*, pp. 756–757, 1999.

[WL89] P. Wolper, and V. Lovinfosse. Verifying Properties of Large Sets of Processes with Network Invariants. In *Proc. Int. Workshop on Automatic Verification Methods for Finite State Systems*, pp. 68–80, 1989.