Unité Mixte de Recherche 5104 CNRS - INPG - UJF
Centre Equation
2, avenue de VIGNATE
F-38610 GIERES
tel : +33 456 52 03 40
fax : +33 456 52 03 50
http://www-verimag.imag.fr

Compositional Deadlock Detection and Verification for Component-based Systems

Saddek Bensalem *Marius Bozga* *Thanh-Hung Nguyen*
Joseph Sifakis

Verimag Research Report n^o TR-2008-5

April 16, 2008

Compositional Deadlock Detection and Verification for Component-based Systems

Saddek Bensalem   *Marius Bozga*   *Thanh-Hung Nguyen*   *Joseph Sifakis*

April 16, 2008

Abstract

We present a heuristic method for compositional deadlock detection and verification of component-based systems described in a subset of the BIP language encompassing multi-party rendezvous without data transfer. The method consists in using automatically generated invariants to prove non-satisfiability of the predicate characterizing global deadlocks.Two kinds of invariants are generated. Component invariants which are over-approximations of components' reachability sets. Interaction invariants which are constraints on the states of the components involved in interactions. Interaction invariants are obtained by computing traps and locks of finite state abstractions of the verified system. The method is supported by D-Finder, an interactive tool that takes as input BIP programs and applies proof strategies to eliminate potential deadlocks by computing increasingly stronger invariants. The experimental results on non-trivial examples allow either to prove deadlock-freedom or to identify very few deadlock configurations that can be analyzing by using state space exploration.

Keywords: deadlock-detection, compositional methods, invariant generation, abstraction

Reviewers: Joseph Sifakis, Marius Bozga, Saddek Bensalem, Thanh-Hung Nguyen

Notes:

How to cite this report:

@techreport { ,
title = { Compositional Deadlock Detection and Verification for Component-based Systems},
authors = { Saddek Bensalem   Marius Bozga   Thanh-Hung Nguyen   Joseph Sifakis},
institution = { Verimag Research Report },
number = {TR-2008-5},
year = { },
note = { }
}

1 Introduction

Deadlock-freedom is an essential property for the correctness of concurrent systems. It characterizes their ability to execute some action over their lifetime. Deadlocks are the most common source of errors in systems involving processes sharing common resources or subject to strong synchronization constraints. System designers need scalable methods and tools for deadlock detection or verification, especially for component-based systems. For these systems, it is essential that correctness could be checked *compositionally*, that is global deadlock-freedom could be inferred from properties of subsystems.

The literature on deadlock analysis techniques not involving state-space exploration is relatively poor. One direction for avoiding state explosion is through the compositional computation of deadlock preserving abstractions e.g. [1]. There also exist results ensuring deadlock-freedom for classes of systems such as Petri nets [2] or data flow systems [3]. Other methods for deadlock detection are based on structural analysis of the interactions between components. These include analysis of dependency graphs between tasks [4, 5] or heuristics for the detection of deadlocks in multi-threaded programs [6, 7]. All these methods are based on coarse grain approximations of the behavior. They are too conservative as they often produce large numbers of potential deadlock configurations that may not be feasible.

The paper presents a heuristic method for compositional deadlock detection and verification of component-based systems described in a subset of the BIP (*Behavior-Interaction-Priority*) language [8]. In BIP, a system is the composition of a set of atomic components which are automata extended with data and functions written in C. In previous papers, [5, 9], we have provided sufficient conditions for deadlock-freedom based on the analysis of dependencies between interactions in the case of pure synchronization, that is without transfer of data. Potential deadlocks correspond to cycles of a dependency graph. We proposed to use invariants in order to show non-feasability of deadlocks characterized by state predicates. This paper shows how to efficiently generate such invariants and use them for deadlock analysis. The main results are the following:

- We propose an heuristic method for detecting potential deadlock configurations and possibly proving global deadlock-freedom. The method consists in iteratively conjuncting the predicate characterizing global deadlocks of a composite system with two kinds of invariants: invariants of atomic components and interaction invariants. If the conjunction is false then the system is deadlock-free. Otherwise, to eliminate unfeasible deadlocks, new invariants are computed until either the conjunction becomes false or the method fails to establish deadlock-freedom. In this case, additional reachability techniques can be used for deadlock confirmation.

- We provide heuristics for computing the two types of invariants. Invariants for atomic components are generated by simple forward analysis of their behavior. These are over-approximations of the set of their reachable states. Interaction invariants characterize constraints on the global state space induced by synchronizations between components. They are computed on compositional abstractions of the global system. They are generalizations of the notions of trap and lock used for the analysis of deadlocks in Petri nets[2].

- The heuristic method has been fully implemented in the D-Finder toolset. D-Finder takes as input BIP models and progressively eliminates deadlocks by generating invariants. For this, it cooperates with two tools: Omega [10] for quantifier elimination and Yices [11] for checking satisfiability of predicates. It is also connected to the state space exploration tool of the BIP platform, for finer analysis when the heuristic fails to prove deadlock-freedom. We provide non trivial examples showing the capabilities of D-Finder as well as the efficiency of the method.

The paper is organized as follows. Section 2 describes the basic semantic model for the considered subset of BIP. Section 3 presents the principle of the method. The heuristics for computing invariants for atomic components and interaction invariants are presented in Section 4. D-Finder is described in Section 5 as well as experimental results. Section 6 presents concluding remarks and future work. Due to space limitations, we provide proofs and experimental data in the Annex.

2 Basic semantic model

We present a model for component-based systems used in the *Behaviour-Interaction-Priority* (BIP) component framework developed at Verimag [8].

This framework has been implemented in a language and a toolset. The BIP language offers primitives and constructs for modelling and composing components. An atomic component consists of a set of ports used for the synchronization with other components, a set of transitions and a set of local variables. Transitions describe the behavior of the component. The BIP toolset includes an editor and a compiler for generating from BIP programs, C++ code executable on a dedicated platform.

We provide a formalization of atomic components in BIP and their composition by using interactions.

Definition 1 (Atomic Component) *An atomic component is a transition system extend with data $B = (L, P, \mathcal{T}, X, \{g_\tau\}_{\tau\in\mathcal{T}}, \{f_\tau\}_{\tau\in\mathcal{T}})$, where:*

- *$(L, P, \mathcal{T})$ is a transition system, that is*
  - *$L = \{l_1, l_2, \ldots, l_k\}$ is a set of control locations,*
  - *P is a set of ports,*
  - *$\mathcal{T} \subseteq L \times P \times L$ is a set of transitions,*
- *$X = \{x_1, \ldots, x_n\}$ is a set of variables and for each $\tau \in \mathcal{T}$ respectively g_τ is a guard, a predicate on X, and $f_\tau(X, X')$ is an update function, a predicate on X (current) and X' (next) state variables.*

Definition 2 (Semantics) *The semantics of $B = (L, P, \mathcal{T}, X, \{g_\tau\}_{\tau\in\mathcal{T}}, \{f_\tau\}_{\tau\in\mathcal{T}})$, is a transition system $(Q, P, \mathcal{T}_0)$ such that*

- *$Q = L \times \mathbf{X}$ where $\mathbf{X}$ denotes the set of valuations of variables X.*
- *$\mathcal{T}_0$ is the set including transitions $((l, \mathbf{x}), p, (l', \mathbf{x}'))$ such that $g_\tau(\mathbf{x}) \wedge f_\tau(\mathbf{x}, \mathbf{x'})$ for some $\tau = (l, p, l') \in \mathcal{T}$. As usual, if $((l, \mathbf{x}), p, (l', \mathbf{x})') \in \mathcal{T}_0$ we write $(l, \mathbf{x}) \xrightarrow{p} (l', \mathbf{x}')$.*

Given a transition $\tau = (l, p, l') \in \mathcal{T}$, l and l' are respectively, the *source* and the *target* location denoted respectively by ${}^\bullet\tau$ and $\tau^\bullet$. We also write

- ${}^\bullet l = \{\tau \mid l = \tau^\bullet\}$ for the set of transitions leading to the control location l.
- $l^\bullet = \{\tau \mid l = {}^\bullet\tau\}$ for the set of transitions leaving the control location l.

For a location l, we use the predicate at_l which is $true$ only if the system is at location l. A state predicate Φ is a boolean expression involving location predicates and predicates on X. Any state predicate can be put in the form $\bigvee_{l\in L} at\_l \wedge \varphi_l$. Notice that predicates on locations are disjoint and their disjunction is true.

We define below a parallel composition parameterized by a set of interactions. We consider only pure synchronizations, that is interactions do not involve data transfer between components.

Definition 3 (Interactions) *Given a set of components $B_1, B_2, \ldots, B_n$, where $B_i = (L_i, P_i, \mathcal{T}_i, X_i, \{g_\tau\}_{\tau\in\mathcal{T}_i}, \{f_\tau\}_{\tau\in\mathcal{T}_i})$, an interaction a is a set of ports, subset of $\bigcup_{i=1}^n P_i$, such that $\forall i = 1, \ldots, n$ $|a \cap P_i| \leq 1$.*

Definition 4 (Parallel Composition) *Given n components $B_i = (L_i, P_i, \mathcal{T}_i, X_i, \{g_\tau\}_{\tau\in\mathcal{T}_i}, \{f_\tau\}_{\tau\in\mathcal{T}_i})$ and a set of interactions γ, we define $B = \gamma(B_1, \ldots, B_n)$ as the component $(L, \gamma, \mathcal{T}, X, \{g_\tau\}_{\tau\in\mathcal{T}}, \{f_\tau\}_{\tau\in\mathcal{T}})$, where:*

- *$(L, \gamma, \mathcal{T})$ is a transition system that is*
  - *$L = L_1 \times L_2 \times \ldots \times L_n$ is a set of control locations,*
  - *$\mathcal{T} \subseteq L \times \gamma \times L$ contains transitions $\tau = ((l_1, \ldots, l_n), a, (l'_1, \ldots, l'_n))$ obtained by synchronization of a set of transitions $\{\tau_i = (l_i, p_i, l'_i) \in \mathcal{T}_i\}_{i\in I}$ such that $\{p_i\}_{i\in I} = a \in \gamma$ and $l'_j = l_j$ if $j \notin I$*

- $X = \bigcup_{i=1}^{n} X_i$ *and for a transition* τ *resulting from the synchronization of a set of transitions* $\{\tau_i\}_{i\in I}$*, the associated guard and function are respectively* $g_\tau = \bigwedge_{i\in I} g_{\tau_i}$ *and* $f_\tau = \bigwedge_{i\in I} f_{\tau_i} \wedge \bigwedge_{i\notin I}(X'_i = X_i)$*.*

Definition 5 (System) *A system* $\mathcal{S}$ *is a pair* $\langle B, Init\rangle$ *where* B *is a component and* $Init$ *is a state predicate characterizing the initial states of* B*.*

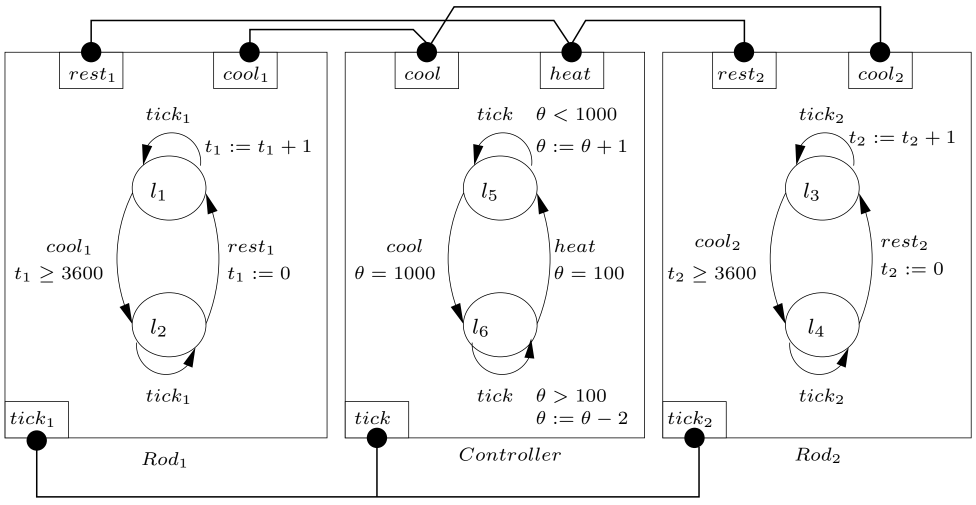

Figure 1: Temperature Control System

Example 1 (Temperature Control System) *[12]*

This system controls the coolant temperature in a reactor tank by moving two independent control rods. The goal is to maintain the coolant between the temperatures θ_m *and* θ_M*. When the temperature reaches its maximum value* θ_M*, the tank must be refrigerated with one of the rods. The temperature rises at a rate* v_r *and decreases at rate* v_d*. A rod can be moved again only if* T *time units have elapsed since the end of its previous movement. If the temperature of the coolant cannot decrease because there is no available rod, a complete shutdown is required.*

We provide a discretized model of the Temperature Control System in BIP, decomposed into three atomic components: a Controller and two components Rod1, Rod2 modeling the rods. We take $\theta_m = 100°$*,* $\theta_M = 1000°$*,* $T = 3600$ *seconds. Furthermore, we assume that* $v_r = 1°/s$ *and* $v_d = 2°/s$*. The Controller has two control locations* $\{l_5, l_6\}$*, a variable* θ*, three ports* $\{tick, cool, heat\}$ *and four transitions: 2 loop transitions labeled by tick which increase or decrease the temperature as time progresses and 2 transitions triggering moves of the rods. The components Rod1 and Rod2 are identical, up to the renaming of states and ports. Each one has two control locations and four transitions: two loop transitions labeled by* $tick$ *and two transitions synchronized with transitions of the Controller. The components are composed by using the following set of interactions, indicated by connectors in the figure:* $\{tick, tick_1, tick_2\}$*,* $\{cool, cool_1\}$*,* $\{cool, cool_2\}$*,* $\{heat, rest_1\}$*,* $\{heat, rest_2\}$*.*

In our model, complete shutdown corresponds to a deadlock. Throughout the paper we verify deadlock-freedom of this example by taking $Init = at\_l_5 \wedge (\theta = 100) \wedge at\_l_1 \wedge (t_1 = 3600) \wedge at\_l_3 \wedge (t_2 = 3600)$*.*
$\square$

3 The Heuristic Method for Checking Deadlock-Freedom

3.1 Invariants and Their Properties

In this subsection, $B = (L, P, \mathcal{T}, X, \{g_\tau\}_{\tau\in\mathcal{T}}, \{f_\tau\}_{\tau\in\mathcal{T}})$ represents a component.

Definition 6 (Deadlock-free States) *Given a component B, we represent by DFS the state predicate characterizing deadlock-free states*
$DFS = \bigvee_{l \in L} \bigvee_{\tau \in l^\bullet} en(\tau)$ *where* $en(\tau) = at\_l \wedge g_\tau$

The predicate $en(\tau)$ characterizes the states from which transition τ is enabled. The following lemma gives a useful characterization of DFS.

Lemma 1 *Given a component B,*
$DFS = \bigwedge_{l \in L}(at\_l \Rightarrow \bigvee_{\tau \in l^\bullet} g_\tau) = \bigwedge_{l \in L}(at\_l \wedge \bigvee_{\tau \in l^\bullet} g_\tau)$

For a component B, we recall here the definition of the $post$ predicate transformer allowing to compute successors of global states represented symbolically by state predicates. Given a state predicate $\Phi = \bigvee_{l \in L} at\_l \wedge \varphi_l$, we define $post(\Phi) = \bigvee_{l \in L}(\bigvee_{\tau=(l,p,l')} at\_l' \wedge post_\tau(\varphi_l))$ where $post_\tau(\varphi)(X) = (\exists X'.g_\tau(X') \wedge f_\tau(X', X) \wedge \varphi(X'))$. Equivalently, we have that $post(\Phi) = \bigvee_{l \in L} at\_l \wedge (\bigvee_{\tau=(l',p,l)} post_\tau(\varphi_{l'}))$. which allows computing $post(\Phi)$ by forward propagation of the assertions associated with control locations in Φ.

We define in a similar way the pre_τ predicate transformer for a transition τ, $pre_\tau(\varphi)(X) = \exists X'.g_\tau(X) \wedge f_\tau(X, X') \wedge \varphi(X')$.

Definition 7 (Invariants) *Given a system $\langle B, Init\rangle$ a state predicate Φ is*

- *an inductive invariant iff $(Init \vee post(\Phi)) \Rightarrow \Phi$.*
- *an invariant iff there exists an inductive invariant Φ_0 such that $\Phi_0 \Rightarrow \Phi$.*
- *a deadlock-free invariant iff it is an invariant and $\Phi \Rightarrow DFS$*

Notice that invariants are over-approximations of the set of the reachable states from $Init$. We extensively use the following well-known results about invariants.

Proposition 1 *Let Φ_1, Φ_2 be two invariants of a component B. Then $\Phi_1 \wedge \Phi_2$, $\Phi_1 \vee \Phi_2$ are invariants of B. Furthermore, if Φ_1 is a deadlock-free invariant then $\Phi_1 \wedge \Phi_2$ is a deadlock-free invariant too.*

Example 2 *For the Temperature Control System of figure 1, the predicates $\Phi_1 = (at\_l_1 \wedge t_1 \geq 0) \vee (at\_l_2 \wedge t_1 \geq 3600)$, $\Phi_2 = (at\_l_3 \wedge t_2 \geq 0) \vee (at\_l_4 \wedge t_2 \geq 3600)$ and $\Phi_3 = (at\_l_5 \wedge 100 \leq \theta \leq 1000) \vee (at\_l_6 \wedge 100 \leq \theta \leq 1000)$ are respectively deadlock-free invariants of the atomic components Rod1, Rod2 and Controller.* □

3.2 The Method

The aim is to show deadlock-freedom for systems of the form $\langle\gamma(B_1, \ldots, B_n), Init\rangle$ by using a heuristic computing more and more precise over-approximations of the set of the reachable states.

Definition 8 (Deadlock States) *We define the predicate DIS characterizing the set of the states of $\gamma(B_1, \ldots, B_n)$ from which all interactions are disabled:*
$DIS = \bigwedge_{a \in \gamma} \neg \bigwedge_{port(\tau) \in a} en(\tau)$
where $port(\tau)$ is the port labeling the transition τ.

Example 3 *For the Temperature Control System (see figure 1), we have:*

$$\begin{array}{rcl} DIS & = & (\neg(at\_l_5 \wedge \theta < 1000)) \bigwedge (\neg(at\_l_6 \wedge \theta = 100) \vee \neg at\_l_2) \\ & \bigwedge & (\neg(at\_l_6 \wedge \theta > 100)) \bigwedge (\neg(at\_l_5 \wedge \theta = 1000) \vee \neg(at\_l_3 \wedge t_2 \geq 3600)) \quad \square \\ & \bigwedge & (\neg(at\_l_5 \wedge \theta = 1000) \vee \neg(at\_l_1 \wedge t_1 \geq 3600)) \bigwedge (\neg(at\_l_6 \wedge \theta = 100) \vee \neg at\_l_4) \end{array}$$

The system $\langle\gamma(B_1, \ldots, B_n), Init\rangle$ is deadlock-free if the predicate $\neg DIS$ is an invariant of the system. Our method relies on standard invariant-based proof techniques. That is, in order to check that $\neg DIS$ is an invariant, we need a stronger invariant Φ such that $\Phi \Rightarrow \neg DIS$ or equivalently $\Phi \wedge DIS = false$.

Figure 2 presents the verification heuristic for a system $\langle\gamma(B_1, \ldots, B_n), Init\rangle$.

```
Input:   $\mathcal{S} = \langle \gamma(B_1, \ldots, B_n), Init \rangle$
Output:  $\mathcal{S}$ is deadlock-free or has a set of potential deadlocks.
1. Find $\Phi$ an invariant of $\mathcal{S}$
2. Compute $DIS$ for $\gamma(B_1, \ldots, B_n)$.
3. If $\Phi \wedge DIS = false$ then return "$\mathcal{S}$ is deadlock-free" else go to 4 or 6
4. Find $\Phi'$ an invariant of $\mathcal{S}$
5. $\Phi := \Phi \wedge \Phi'$ go to 3
6. return the set of the solutions that satisfy $\Phi \wedge DIS$
```

Figure 2: Heuristic for Deadlock Verification

Example 4 $\Phi = \Phi_1 \wedge \Phi_2 \wedge \Phi_3$ *is the conjunction of the deadlock-free invariants given in example 2. The predicate* $\Phi \wedge DIS$*, where* DIS *is given in example 3, is satisfiable and it is the disjunction of the following terms:*

1. $(at\_l_1 \wedge 0 \leq t_1 < 3600) \wedge (at\_l_3 \wedge 0 \leq t_2 < 3600) \wedge (at\_l_6 \wedge \theta = 100)$
2. $(at\_l_1 \wedge 0 \leq t_1 < 3600) \wedge (at\_l_4 \wedge t_2 \geq 3600) \wedge (at\_l_5 \wedge \theta = 1000)$
3. $(at\_l_1 \wedge 0 \leq t_1 < 3600) \wedge (at\_l_3 \wedge 0 \leq t_2 < 3600) \wedge (at\_l_5 \wedge \theta = 1000)$
4. $(at\_l_2 \wedge t_1 \geq 3600) \wedge (at\_l_3 \wedge 0 \leq t_2 < 3600) \wedge (at\_l_5 \wedge \theta = 1000)$
5. $(at\_l_2 \wedge t_1 \geq 3600) \wedge (at\_l_4 \wedge t_2 \geq 3600) \wedge (at\_l_5 \wedge \theta = 1000)$

Each one of the above terms represents a family of possible deadlocks. To decrease the number of potential deadlocks, we find a new invariant Φ' *stronger than* Φ*, such that* $\Phi' = \Phi \wedge \Phi_{int}$*, where* Φ_{int} *is an invariant on the states of Rod1, Rod2 and Controller induced by the interactions:*

$$\begin{array}{l} (\ (at\_l_2 \wedge t_1 \geq 3600) \vee (at\_l_4 \wedge t_2 \geq 3600) \vee (at\_l_5 \wedge 100 \leq \theta \leq 1000)\ ) \\ \bigwedge (\ (at\_l_1 \wedge t_1 \geq 0) \vee (at\_l_2 \wedge t_1 \geq 3600) \vee (at\_l_3 \wedge t_2 \geq 0) \vee (at\_l_4 \wedge t_2 \geq 3600)\ ) \\ \bigwedge (\ (at\_l_3 \wedge t_2 \geq 1) \vee (at\_l_4) \vee (at\_l_5 \wedge \theta = 100)\ ) \\ \bigwedge (\ (at\_l_1 \wedge t_1 \geq 0) \vee (at\_l_3 \wedge t_2 \geq 0) \vee (at\_l_6 \wedge \theta = 1000) \vee (at\_l_6 \vee 100 \leq \theta \leq 998)\ ) \\ \bigwedge (\ (at\_l_1 \wedge t_1 \geq 1) \vee (at\_l_2) \vee (at\_l_5 \wedge \theta = 100)\ ) \end{array}$$

The predicate $\Phi' \wedge DIS$ *is reduced to:*

6. $(at\_l_1 \wedge 1 \leq t_1 < 3600) \wedge (at\_l_3 \wedge 1 \leq t_2 < 3600) \wedge (at\_l_5 \wedge \theta = 1000)$
7. $(at\_l_1 \wedge 1 \leq t_1 < 3600) \wedge (at\_l_4 \wedge t_2 \geq 3600) \wedge (at\_l_5 \wedge \theta = 1000)$
8. $(at\_l_2 \wedge t_1 \geq 3600) \wedge (at\_l_3 \wedge 1 \leq t_2 < 3600) \wedge (at\_l_5 \wedge \theta = 1000)$

Finally, it can be checked by using finite state reachability analysis on an abstraction of the system without variables, that only the first term represents feasible deadlocks, the two other being spurious. This term characterizes deadlock configurations leading to complete shutdown. $\square$

4 Automatic Generation of Invariants

In this section, we present techniques for computing invariants for atomic components and interaction invariants.

4.1 Computing Component Invariants

We compute sequences of inductive invariants for atomic components by using the proposition below.

Proposition 2 *Given a system $\mathcal{S} = \langle B, Init \rangle$, the following iteration defines a sequence of increasingly stronger inductive invariants:*

$$\Phi_0 = true \quad \Phi_{i+1} = Init \vee post(\Phi_i)$$

In our heuristic, we use different strategies for producing component invariants. We usually iterate until we find deadlock-free invariants. Their use guarantees that global deadlocks are exclusively due to synchronization.

A key issue is the efficient computation of component invariants as the precise computation of $post$ requires quantifier elimination. An alternative to quantifier elimination is to compute over-approximations of $post$ based on syntactic analysis of the predicates. In this case, the obtained invariants are not inductive.

We give here a very brief description of a syntactic technique used for approximating $post_\tau$ for a fixed transition τ. A more detailed presentation, as well as much stronger techniques for generating component invariants are given in [13].

Consider a transition $\tau = (l, p, l')$ of $B = (L, P, \mathcal{T}, X, \{g_\tau\}_{\tau \in \mathcal{T}}, \{f_\tau\}_{\tau \in \mathcal{T}})$. Assume that its guard is of the form $g_\tau(Y)$ and the associated update function f_τ is of the form $Z_1' = e_\tau(U) \wedge Z_2' = Z_2$ where $Y, Z_1, Z_2, U \subseteq X$ and $\{Z_1, Z_2\}$ is a partition of X.

For an arbitrary predicate φ find a decomposition $\varphi = \varphi_1(Y_1) \wedge \varphi_2(Y_2)$ such that $Y_2 \cap Z_1 = \emptyset$ i.e. which has a conjunct not affected by the update function f_τ. We apply the following rule to compute over-approximations $post_\tau^a(\varphi)$ of $post_\tau(\varphi)$

$$post_\tau^a(\varphi) = \varphi_2(Y_2) \wedge \left\{ \begin{array}{ll} g_\tau(Y) & \text{if } Z_1 \cap Y = \emptyset \\ true & \text{otherwise} \end{array} \right\} \wedge \left\{ \begin{array}{ll} Z_1 = e_\tau(U) & \text{if } Z_1 \cap U = \emptyset \\ true & \text{otherwise} \end{array} \right\}$$

Proposition 3 *If τ and φ are respectively a transition and a state predicate as above, then $post_\tau(\varphi) \Rightarrow post_\tau^a(\varphi)$.*

4.2 Computing Interaction Invariants

As shown in example 4, component invariants do not suffice for removing unfeasible deadlocks. We need stronger invariants called *interaction invariants* because they involve state variables from different atomic components.

Consider a system $\mathcal{S} = \langle \gamma(B_1, \ldots, B_n), Init \rangle$ and a set of invariants $\Phi_1 \ldots \Phi_n$ corresponding to its components. We show below, for each component B_i and its associated invariant Φ_i, how to define a finite state abstraction α_i and to compute an abstract transition system $B_i^{\alpha_i}$. Then, we compute interaction invariants for $\mathcal{S}$ by analyzing, without constructing explicitely the state space, the parallel composition $B^\alpha = \gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n})$.

4.2.1 Computing abstractions

Definition 9 (Abstraction Function) *Consider an invariant Φ of a system $\langle B, Init \rangle$ written in disjunctive form $\Phi = \bigvee_{l \in L} at\_l \wedge (\bigvee_{m \in M_l} \varphi_{lm})$ such that atomic predicates of the form $at\_l \wedge \varphi_{lm}$ are disjoint. An abstraction function α is an injective function associating with each atomic predicate $at\_l \wedge \varphi_{lm}$ a symbol $\phi = \alpha(at\_l \wedge \varphi_{lm})$ called abstract state. We denote by Φ^α the set of the abstract states.*

Definition 10 (Abstract System) *Given a system $\mathcal{S} = \langle B, Init \rangle$, an invariant Φ and an associated abstraction function α, we define the abstract system $\mathcal{S}^\alpha = \langle B^\alpha, Init^\alpha \rangle$ where*

- *$B^\alpha = (\Phi^\alpha, P, \leadsto)$ is a transition system with $\leadsto$ such that for any pair of abstract states $\phi = \alpha(at\_l \wedge \varphi)$ and $\phi' = \alpha(at\_l' \wedge \varphi')$ we have $\phi \overset{p}{\leadsto} \phi'$ iff $\exists \tau = (l, p, l') \in \mathcal{T}$ and $\varphi \wedge pre_\tau(\varphi') \neq false$,*
- *$Init^\alpha = \bigvee_{\phi \in \Phi_0^\alpha} at\_\phi$ where $\Phi_0^\alpha = \{\phi \in \Phi^\alpha \mid \alpha^{-1}(\phi) \wedge Init \neq false\}$ is the set of the initial abstract states.*

We apply the following method presented in [14] and implemented in the tool InVeSt [15] in order to compute an abstract transition system B^α of a component B. We proceed by elimination starting from the universal relation on abstract states. We eliminate pairs of abstract states in a conservative way. To check whether $\phi \overset{p}{\leadsto} \phi'$, where $\phi = \alpha(at\_l \wedge \varphi)$ and $\phi' = \alpha(at\_l' \wedge \varphi')$, can be eliminated, we check that for all concrete transitions $\tau = (l, p, l')$ we have $\varphi \wedge pre_\tau(\varphi') = false$.

Proposition 4 *If B^α is an abstraction of B with respect to Φ and abstraction function α, then B^α simulates B. Moreover, if Φ^α is an invariant of $\langle B^\alpha, Init^\alpha\rangle$ then $\alpha^{-1}(\Phi^\alpha)$ is an invariant of $\langle B, Init\rangle$.*

Example 5 *The table below provides the abstract states constructed from the components invariants Φ_1, Φ_2, Φ_3 of respectively Rod1, Rod2, Controller given in example 2.*

$\phi_{11} = at\_l_1 \wedge t_1 = 0$	$\phi_{51} = at\_l_5 \wedge \theta = 100$	$\phi_{31} = at\_l_3 \wedge t_2 = 0$
$\phi_{12} = at\_l_1 \wedge t_1 \geq 1$	$\phi_{52} = at\_l_5 \wedge 101 \leq \theta \leq 1000$	$\phi_{32} = at\_l_3 \wedge t_2 \geq 1$
$\phi_{21} = at\_l_2 \wedge t_1 \geq 3600$	$\phi_{61} = at\_l_6 \wedge \theta = 1000$	$\phi_{41} = at\_l_4 \wedge t_2 \geq 3600$
$\phi_{22} = at\_l_2 \wedge t_1 < 3600$	$\phi_{62} = at\_l_6 \wedge 100 \leq \theta \leq 998$	$\phi_{42} = at\_l_4 \wedge t_2 < 3600$

Figure 3 presents the computed abstraction of the Temperature Control System with respect to the considered invariants.□

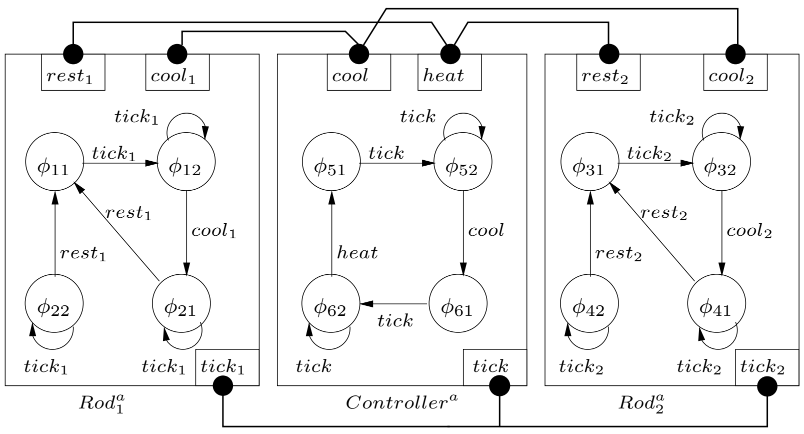

Figure 3: Abstraction of the Temperature Control System.

The following proposition is a well-known result about preservation of abstraction by parallel composition.

Proposition 5 *If $B_i^{\alpha_i}$ is an abstraction of B_i with respect to invariants Φ_i and abstraction functions α_i for $i = 1, ..., n$, then $B^\alpha = \gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n})$ is an abstraction of $B = \gamma(B_1, ..., B_n)$ with respect to $\bigwedge_{i=1}^n \Phi_i$ and an abstraction function α obtained as a composition of the α_i.*

4.2.2 Computing Traps and Locks.

In this subsection, we provide results for computing traps and locks and the associated interaction invariants. They are computed directly from the parallel composition $\gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n})$ of the abstract components.

Definition 11 (Forward and Backward Interaction Sets) *Given the parallel composition $\gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n})$ where $B_i^{\alpha_i} = (\Phi_i^{\alpha_i}, P_i, \leadsto_i)$, we define for a set of abstract states $\Psi \subseteq \bigcup_{i=1}^n \Phi_i^{\alpha_i}$*

- *its forward interaction set $\Psi^+ = \bigcup_{\phi \in \Psi} \phi^+$ where*
  $\phi^+ = \{(\tau_1, \ldots, \tau_k) \mid \forall i. \tau_i \in \leadsto_i \ \wedge\ \exists i. {}^\bullet\tau_i = \phi \ \wedge\ \{port(\tau_i)\}_{i=1}^k \in \gamma\}$

```
Input:   $\mathcal{S} = \langle \gamma(B_1, \ldots, B_n), Init \rangle$, component invariants $\Phi_i$, for $i = 1, ..., n$
Output:  Interaction invariant $\Phi$.

1. For each $B_i$, compute the corresponding abstraction $B_i^{\alpha_i}$.
3. For $\gamma(B_1^{\alpha_1}, ..., B_n^{\alpha_n})$, compute the traps $\{\Psi_1, \Psi_2, \ldots, \Psi_m\}$
   containing some abstract initial state
4. For each trap $\Psi_k$, compute the corresponding interaction invariant $\Phi_k = \bigvee_{\phi \in \Psi_k} \alpha^{-1}(\phi)$.
5. Return $\Phi = \bigwedge_{k=1}^{m} \Phi_k$
```

Figure 4: Computing Interaction Invariants

- *its backward interaction set* ${}^{+}\Psi = \bigcup_{\phi \in \Psi} {}^{+}\phi$ *where*
  ${}^{+}\phi = \{(\tau_1, \ldots, \tau_k) \mid \forall i.\tau_i \in \leadsto_i \ \wedge \ \exists i \ \tau_i^{\bullet} = \phi \ \wedge \ \{port(\tau_i)\}_{i=1}^{k} \in \gamma\}$

Notice that the computation of forward/backward interaction sets does not require the explicit computation of the abstract product $B^\alpha = \gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n})$. Interaction sets consist of sets of component transitions involved in some interaction. These sets can also be viewed as transitions in a Petri net. Ψ^+ (resp. ${}^{+}\Psi$) contains sets of transitions which have some abstract state of some $\phi \in \Psi$ as precondition (resp. postcondition).

Definition 12 (Traps and Locks) *Given a parallel composition $\gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n})$ where $B_i^{\alpha_i} = (\Phi_i^{\alpha_i}, P_i, \leadsto_i$), a trap (resp. lock) is a set Ψ of abstract states $\Psi \subseteq \bigcup_{i=1}^{n} \Phi_i^{\alpha_i}$ such that $\Psi^+ \subseteq^+ \Psi$ (resp. ${}^{+}\Psi \subseteq \Psi^+$).*

Proposition 6 *Given an abstract system $\mathcal{S}^\alpha = \langle \gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n}), Init^\alpha \rangle$, if the set of abstract states $\Psi \subseteq \bigcup_{i=1}^{n} \Phi_i^{\alpha_i}$ is a trap containing an initial state of some abstract component (resp. lock containing no initial state of any abstract component) then $\bigvee_{\phi \in \Psi} at\_\phi$ (resp. $\bigwedge_{\phi \in \Psi} \neg at\_\phi$) is an invariant of $\mathcal{S}^\alpha$ and $\bigvee_{\phi \in \Psi} \alpha^{-1}(\phi)$ (resp. $\bigwedge_{\phi \in \Psi} \neg \alpha^{-1}(\phi)$) is an invariant of $\mathcal{S} = \langle \gamma(B_1, \ldots, B_n), Init \rangle$.*

The following result given in [16] characterizes traps and locks as solution of a system of implications.

Proposition 7 *Let $\gamma(B_1^{\alpha_1}, ..., B_n^{\alpha_n})$ and a boolean valuation $\mathbf{v} : \bigcup_{i=1}^{n} \Phi^{\alpha_i} \to \mathbb{B}$.*
• *If* **v** *satisfies the following set of the implications, then $\{\phi_i\}_{\mathbf{v}(\phi_i)}$ is a trap:*

$$\{\phi_i \ \Rightarrow \bigwedge_{\{\tau_1,\ldots,\tau_k\} \in \phi_i^+} (\bigvee_{\phi_j \in \bigcup_{m=1}^{k} \tau_m^{\bullet}} \phi_j)\} \text{ for each } \phi_i \in \bigcup_{i=1}^{n} \Phi^{\alpha_i}$$

• *If* **v** *satisfies the following set of the implications, then, $\{\phi_i\}_{\mathbf{v}(\phi_i)}$ is a lock:*

$$\{\phi_i \ \Rightarrow \bigwedge_{\{\tau_1,\ldots,\tau_k\} \in {}^{+}\phi_i} (\bigvee_{\phi_j \in \bigcup_{m=1}^{k} {}^{\bullet}\tau_m} \phi_j)\} \text{ for each } \phi_i \in \bigcup_{i=1}^{n} \Phi^{\alpha_i}$$

Example 6 *The set of implications characterizing traps are given in section B of the appendix. From these implications we compute the minimal traps containing initial abstract states is: $\Psi_1 = \{\phi_{21}, \phi_{41}, \phi_{51}, \phi_{52}\}$, $\Psi_2 = \{\phi_{11}, \phi_{12}, \phi_{21}, \phi_{31}, \phi_{32}, \phi_{41}\}$, $\Psi_3 = \{\phi_{32}, \phi_{41}, \phi_{42}, \phi_{51}\}$, $\Psi_4 = \{\phi_{11}, \phi_{12}, \phi_{31}, \phi_{32}, \phi_{61}, \phi_{62}\}$ and $\Psi_5 = \{\phi_{12}, \phi_{21}, \phi_{22}, \phi_{51}\}$. By using proposition 6, we obtain the interaction invariant $\bigwedge_{i=1}^{5} \bigwedge_{\phi \in \Psi_i} \alpha^{-1}(\phi)$ which is exactly the invariant Φ_{int} given in example 4.* □

We give the principle of an algorithm for computing traps (dually for locks) of a system $\mathcal{S} = \langle \gamma(B_1, ..., B_n), Init \rangle$ and a set of component invariant Φ_i, for $i = 1, ..., n$.

5 Implementation and Experimentation

5.0.3 The D-Finder Toolset

The D-Finder toolset takes as input a BIP model and computes component invariants CI. This step may require quantifier elimination by using Omega. Then, it checks for deadlock-freedom of component invariants by using Yices. From the generated component invariants, it computes an abstraction of the BIP

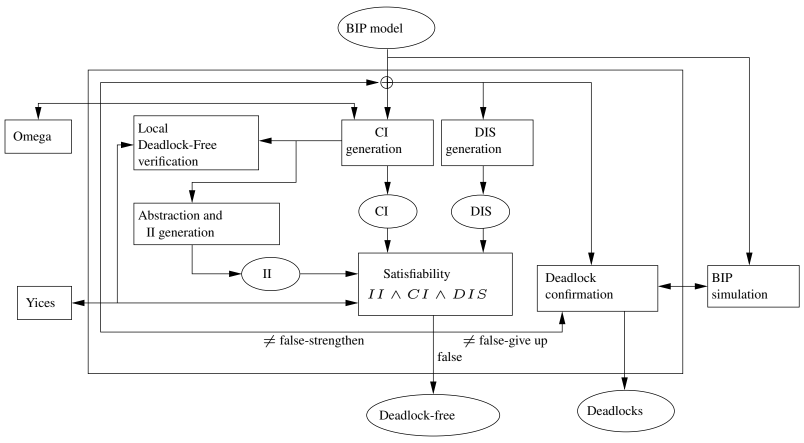

Figure 5: D-Finder

model and the corresponding interaction invariants II. Then, it checks satisfiability of the conjunction $II \wedge CI \wedge DIS$. If the conjunction is unsatisfiable, then there is no deadlock else either it generates stronger component and interaction invariants or it tries to confirm the detected deadlocks by using reachability analysis techniques.

5.0.4 Experimental results

The first example is the Temperature Control System extensively presented in the paper. The second example is Utopar, an industrial case study of the European Integrated project SPEEDS (http://www.speeds.eu.com/) about an automated transportation system. A succinct description of Utopar can be found at http://www.artist-embedded.org/COMBEST-draft/. The system is the composition of three types of components: autonomous vehicles, called U-cars, a centralized Automatic Control System and Calling Units. The latter two types have (almost exclusively) discrete behavior. U-cars are equipped with a local controller, responsible for handling the U-cars sensors and performing various routing and driving computations depending on users' requests. We analyzed a simplified version of Utopar by abstracting from data exchanged between components as well as from continuous dynamics of the cars. In this version, each U-Car is modeled by a component having 7 control locations and 6 integer variables. The Automatic Control System has 3 control locations and 2 integer variables. The Calling Units have 2 control locations and no variables. Finally, as third example, we consider Readers-Writer systems in order to evaluate how the method scales up for components without data.

The table below gives an overview of the experimental results obtained for the three examples. For the columns: n is the number of BIP components in the example, q is the total number of control locations, x_b (resp. x_i) is the total number of boolean (resp. integer) variables, D is the number of potential deadlock configurations (disjuncts) in DIS, D_c (resp. D_{ci}) is the number of potential deadlock configurations remaining in $DIS \wedge CI$ (resp. $DIS \wedge CI \wedge II$) and t is the total time for computing invariants and checking for satisfiability of $DIS \wedge CI \wedge II$. Detailed results are available at http://www-verimag.imag.fr/˜ thnguyen/tool.

6 Conclusion

The paper presents a heuristic method for compositional deadlock detection and verification of component-based systems. The principle of the method is very simple: it consists in computing more and more precise over-approximations of the reachability set of a composite system and showing that they do not contain deadlock states. The method innovates in that it efficiently combines two types of invariants: invariants of atomic components and interaction invariants. As shown through several examples, the use of only one

example	n	q	x_b	x_i	D	D_c	D_{ci}	t
Temperature Control System (2 rods)	3	6	0	3	8	5	3	3s
Temperature Control System (4 rods)	5	10	0	5	32	17	15	1m05s
Utopar System (4 U-Cars, 9 Calling Units)	14	45	4	26	-	-	0	1m42s
Utopar System (8 U-Cars, 16 Calling Units)	25	91	8	50	-	-	0	22m02s
Readers-Writer (50 readers)	52	106	0	1	$\sim 10^{15}$	$\sim 10^{15}$	0	1m15s
Readers-Writer (100 readers)	102	206	0	1	$\sim 10^{30}$	$\sim 10^{30}$	0	15m28s
Readers-Writer (130 readers)	132	266	0	1	$\sim 10^{39}$	$\sim 10^{39}$	0	29m13s

type of invariants does not suffice to eliminate unfeasible deadlocks. On the contrary, their combination led either to proving deadlock-freedom or to the detection of only a few feasible deadlock configurations.

The method uses only lightweight analysis techniques that do not involve computation of fixpoints and avoids global state space exploration. Component invariants are easy to compute by forward propagation of predicates. The computation of interaction invariants requires the compositional computation of finite state abstractions for components and the enumeration of solutions of systems of implications. Here there is a risk of explosion, if exhaustivity of solutions is necessary in the analysis process. There are no restrictions on the type of data as long as we stay within theories for which there exist efficient decision procedures.

The obtained experimental results for non trivial case studies are really convincing. The method can be adapted to interactions with data transfer. Data transfer with finite domains, can be encoded by creating individual interactions for each configuration of transferred data. Otherwise, the notion of component invariant and subsequently the notion of interaction invariant can be extended to take into account transferred data. Finally, an interesting work direction would be to study how this methodology can be adapted to prove safety properties other than deadlock-freedom.

References

[1] Chaki, S., Clarke, E.M., Ouaknine, J., Sharygina, N., Sinha, N.: Concurrent software verification with states, events, and deadlocks. FAC **17**(4) (2005) 461–483 1

[2] Peterson, J.: Petri Net theory and the modelling of systems. Englewood-Cliffs: Prentice Hall (1981) 1, A

[3] Zhou, Y., Lee, E.A.: A causality interface for deadlock analysis in dataflow. In: EMSOFT’06. (2006) 44–52 1

[4] Attie, P.C., Chockler, H.: Efficiently verifiable conditions for deadlock-freedom of large concurrent programs. In: VMCAI. (2005) 465–481 1

[5] Gossler, G., Sifakis, J.: Component-based construction of deadlock-free systems. In: FSTTCS, Invited talk,. Volume LNCS 2914., Mumbai (December 2003) 1

[6] Bensalem, S., Havelund, K.: Dynamic deadlock analysis of multi-threaded programs. In: HVC’05. (2005) 208–223 1

[7] Agarwal, R., Wang, L., Stoller, S.D.: Detecting potential deadlocks with static analysis and run-time monitoring. In: HVC’05. (2005) 191–207 1

[8] Basu, A., Bozga, M., Sifakis, J.: Modeling heterogeneous real-time components in bip. In: SEFM. (2006) 3–12 1, 2

[9] Gößler, G., Graf, S., Majster-Cederbaum, M.E., Martens, M., Sifakis, J.: An approach to modelling and verification of component based systems. In: SOFSEM (1). (2007) 295–308 1

[10] Team, O.: The omega library. Version 1.1.0 (November 1996) 1

[11] Dutertre, B., de Moura, L.: A fast linear-arithmetic solver for DPLL(T). In: CAV'06. Volume 4144 of LNCS. (2006) 81–94 1

[12] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. TCS **138**(1) (1995) 3–34 1

[13] Bensalem, S., Lakhnech, Y.: Automatic generation of invariants. FMSD **15**(1) (July 1999) 75–92 4.1

[14] Bensalem, S., Lakhnech, Y., Owre, S.: Computing abstractions of infinite state systems automatically and compositionally. In: CAV'98. Volume 1427 of LNCS. 319–331 4.2.1

[15] Bensalem, S., Lakhnech, Y., Owre, S.: Invest: A tool for the verification of invariants. In: CAV'98. Volume 1427 of LNCS. 505–510 4.2.1

[16] Sifakis, J.: Structural properties of petri nets. In: MFCS'78. Volume 64 of LNCS. (1978) 474–483 4.2.2

A Proofs

Proof of lemma 1 **Proof:** The proof is based on the fact that $\bigvee_{l \in L} at\_l$ and $\neg(at\_l \wedge at\_l')$ for $l \neq l'$. □

Proof of proposition 2 **Proof:** By induction. Φ_0 is an inductive invariant. If Φ_i is an inductive invariant then $Init \vee post(\Phi_i) \Rightarrow \Phi_i$. As $post$ is monotonic and distributes over disjunction, $post(\Phi_{i+1}) = post(Init \vee post(\Phi_i)) \Rightarrow post(\Phi_i) \Rightarrow \Phi_{i+1}$. Moreover, $Init \Rightarrow \Phi_{i+1}$. So Φ_{i+1} is an inductive invariant. □

Proof of proposition 3 **Proof:** We can over-approximate successively $post_\tau(\varphi)$ as follows:

$$\begin{array}{lcl}
post_\tau(\varphi)(X') & = & \exists X.\, (\varphi(X) \wedge g_\tau(X) \wedge f_\tau(X, X')) \\
 & = & \exists Z_1, Z_2.\, (\varphi_1(Y_1) \wedge \varphi_2(Y_2) \wedge g_\tau(Y) \wedge Z_1' = e_\tau(U) \wedge Z_2' = Z_2) \\
 & \Rightarrow & \exists Z_2.\, (\varphi_2(Y_2) \wedge Z_2' = Z_2) \bigwedge \exists Z_1, Z_2.\, (g_\tau(Y) \wedge Z_1' = e_\tau(U) \wedge Z_2' = Z_2) \\
 & = & \varphi_2(Y_2') \bigwedge \exists Z_1, Z_2.\, (g_\tau(Y) \wedge Z_1' = e_\tau(U) \wedge Z_2' = Z_2) \\
 & \Rightarrow & \varphi_2(Y_2') \bigwedge \left\{ \begin{array}{ll} g_\tau(Y') & \text{if } Z_1 \cap Y = \emptyset \\ true & \text{otherwise} \end{array} \right\} \bigwedge \left\{ \begin{array}{ll} Z_1' = e_\tau(U') & \text{if } Z_1 \cap U = \emptyset \\ true & \text{otherwise} \end{array} \right\} \\
 & = & post_\tau^a(\varphi)(X')
\end{array}$$

Proof of proposition 4 **Proof:** We show that the relation $(l, \mathbf{x}) R \phi$ is a simulation if $\alpha^{-1}(\phi) = at\_l \wedge \varphi$ and $\varphi(\mathbf{x})$ for the valuation $\mathbf{x}$. If $(l, \mathbf{x}) \xrightarrow{p} (l', \mathbf{x}')$ is a transition of B and $(l, \mathbf{x}) R \phi$ for some abstract state ϕ, then we show that there exists $\phi' = \alpha(at\_l' \wedge \varphi')$ such that $\phi \overset{p}{\rightsquigarrow} \phi'$. As Φ is an invariant of B, if $(l', \mathbf{x}')$ is reachable then $\exists \varphi'\ at\_l' \wedge \varphi' \Rightarrow \Phi$ such that $\varphi'(\mathbf{x}')$ and $\phi' = \alpha(at\_l' \wedge \varphi')$. Moreover, as $\varphi(\mathbf{x}) \wedge \varphi'(\mathbf{x}')$, we have $\varphi(\mathbf{x}) \wedge pre_\tau(\varphi)(\mathbf{x}) \neq false$ for $\tau = (l, p, l')$ and therefore $\phi \overset{p}{\rightsquigarrow} \phi'$. □

Proof of proposition 6 **Proof:** The abstract behaviour B^α is equally represented by a 1-safe Petri net where places correspond to abstract states of $\bigcup_{i=1}^n \Phi_i^{\alpha_i}$ and transitions correspond to interactions of γ. Moreover, the traps and locks previously introduced correspond precisely to the traps and the locks (aka siphons) in this Petri net.

Regarding traps and locks in Petri nets, the following invariance properties hold (1) *if a trap is initially marked, it remains marked through all computation of the net* and (2) dually, *if a lock is initially unmarked, it remains unmarked through all computations of the net* (see [2] for details). These properties can be lifted to the abstract system in order to obtain synchronization invariants as follows.

Any trap Ψ containing the initial state of some abstract component B_i^α corresponds to an initially marked trap in the underlying Petri net. Then, according to its characteristic property, it will stay marked in any execution. For the abstract system $\mathcal{S}^\alpha$ this implies that it always has at least one of its abstract components in some abstract state of Ψ. This simply means that the predicate $\bigvee_{\phi \in \Psi} at\_\phi$ is an invariant of B^α and consequently $\bigvee_{\phi \in \Psi} \alpha^{-1}(\phi)$ is an invariant of $\mathcal{S}$.

A dual proof can be provided for locks. □

B Temperature Control System

For the abstraction of the Temperature Control System given in figure 3 the following set of implications characterizes the set of traps:

$$\begin{array}{lll}
\phi_{11} \Rightarrow (\phi_{12} \vee \phi_{42} \vee \phi_{52}) \wedge & (\phi_{12} \vee \phi_{32} \vee \phi_{62}) \wedge & \phi_{12} \Rightarrow (\phi_{61} \vee \phi_{21}) \\
\qquad (\phi_{12} \vee \phi_{42} \vee \phi_{62}) \wedge & (\phi_{12} \vee \phi_{41} \vee \phi_{52}) \wedge & \phi_{21} \Rightarrow (\phi_{51} \vee \phi_{11}) \\
\qquad (\phi_{12} \vee \phi_{32} \vee \phi_{52}) \wedge & (\phi_{12} \vee \phi_{41} \vee \phi_{62}) & \phi_{22} \Rightarrow (\phi_{51} \vee \phi_{11})
\end{array}$$

$$\begin{aligned}
\phi_{31} \Rightarrow\ & (\phi_{22} \vee \phi_{32} \vee \phi_{52})\wedge \\
& (\phi_{22} \vee \phi_{32} \vee \phi_{62})\wedge \\
& (\phi_{12} \vee \phi_{32} \vee \phi_{52})\wedge \\
& (\phi_{12} \vee \phi_{32} \vee \phi_{62})\wedge \\
& (\phi_{21} \vee \phi_{32} \vee \phi_{52})\wedge \\
& (\phi_{21} \vee \phi_{32} \vee \phi_{62}) \\
\phi_{32} \Rightarrow\ & (\phi_{61} \vee \phi_{41}) \\
\phi_{41} \Rightarrow\ & (\phi_{51} \vee \phi_{31}) \\
\phi_{42} \Rightarrow\ & (\phi_{51} \vee \phi_{31}) \\
\phi_{51} \Rightarrow\ & (\phi_{22} \vee \phi_{42} \vee \phi_{52})\wedge \\
& (\phi_{22} \vee \phi_{32} \vee \phi_{52})\wedge \\
& (\phi_{12} \vee \phi_{42} \vee \phi_{52})\wedge \\
& (\phi_{12} \vee \phi_{32} \vee \phi_{52})\wedge \\
& (\phi_{22} \vee \phi_{41} \vee \phi_{52})\wedge \\
& (\phi_{21} \vee \phi_{42} \vee \phi_{52})\wedge \\
& (\phi_{21} \vee \phi_{41} \vee \phi_{52})\wedge \\
& (\phi_{21} \vee \phi_{32} \vee \phi_{52})\wedge \\
& (\phi_{12} \vee \phi_{41} \vee \phi_{52}) \\
\phi_{52} \Rightarrow\ & (\phi_{61} \vee \phi_{21})\wedge \\
& (\phi_{61} \vee \phi_{41}) \\
\phi_{61} \Rightarrow\ & (\phi_{22} \vee \phi_{42} \vee \phi_{62})\wedge \\
& (\phi_{22} \vee \phi_{32} \vee \phi_{62})\wedge \\
& (\phi_{12} \vee \phi_{42} \vee \phi_{62})\wedge \\
& (\phi_{12} \vee \phi_{32} \vee \phi_{62})\wedge \\
& (\phi_{22} \vee \phi_{41} \vee \phi_{62})\wedge \\
& (\phi_{21} \vee \phi_{42} \vee \phi_{62})\wedge \\
& (\phi_{21} \vee \phi_{41} \vee \phi_{62})\wedge \\
& (\phi_{21} \vee \phi_{32} \vee \phi_{62})\wedge \\
& (\phi_{12} \vee \phi_{41} \vee \phi_{62}) \\
\phi_{62} \Rightarrow\ & (\phi_{51} \vee \phi_{11})\wedge \\
& (\phi_{51} \vee \phi_{31})
\end{aligned}$$