Compositional Verification for Component-based Systems and Application

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

Verimag Laboratory

UJF/CNRS

Monday, June 9, 2008

Verification for concurrent systems

- Hard problem due to state explosion
- Compositional verification techniques limit state explosion. One example of compositional rules is

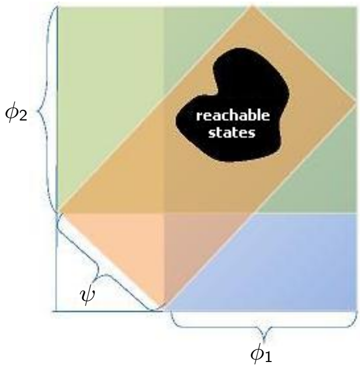
$$\frac{B_1 < \Phi_1 >,\ B_2 < \Phi_2 >,\ C(\Phi_1, \Phi_2, \Phi)}{B_1 \| B_2 < \Phi >}$$

- One approach is *assume-garantie* but many issues make it difficult such as finding decomposition into sub-systems, finding adequate assumptions... [Cobleigh et al., 2008]

Verification for concurrent systems

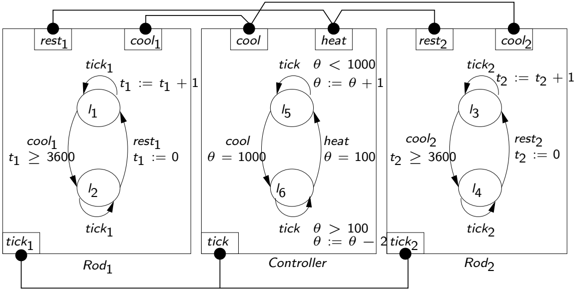
Our approach for compositional verification of invariants is based on the following rule:

$$\frac{B_1 < \Phi_1 >, B_2 < \Phi_2 >,\ \Psi \in II(B_1 \parallel B_2, \Phi_1, \Phi_2),\ \Phi_1 \wedge \Phi_2 \wedge \Psi \Rightarrow \Phi}{B_1 \parallel B_2 < \Phi >}$$



Outline

An example : Temperature Control System

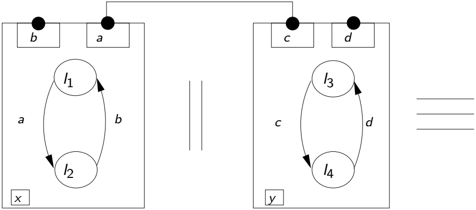


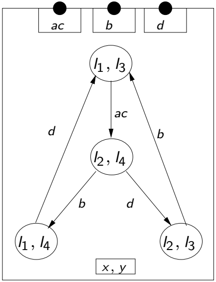
$Rod_1 = (L, P, T, X, \{g_\tau\}_{\tau \in T}, \{f_\tau\}_{\tau \in T})$
$L = \{l_1, l_2\}$
$P = \{rest_1, cool_1, tick_1\}$
$X = \{t_1\}$
$\tau_1 = (l_1, tick_1, l_1), g_{\tau_1} = true, f_{\tau_1} = (t_1 := t_1 + 1)$
$\tau_2 = (l_2, tick_1, l_2), g_{\tau_1} = true$
$\tau_3 = (l_1, cool_1, l_2), g_{\tau_1} = (t_1 \geq 3600)$
$\tau_4 = (l_2, rest_1, l_1), g_{\tau_1} = true, f_{\tau_1} = (t_1 := 0)$

Set of interactions $A = \{a_1, a_2, a_3, a_4, a_5\}$
$a_1 = \{cool, cool_1\}, G_{a_1} = true$
$a_2 = \{cool, cool_2\}, G_{a_2} = true$
$a_3 = \{heat, rest_1\}, G_{a_3} = true$
$a_4 = \{heat, rest_2\}, G_{a_4} = true$
$a_5 = \{tick, tick_1, tick_2\}, G_{a_5} = true$

Composition

$\gamma = \{a_1, a_2, a_3\}$
$a_1 = \{a, c\}, g_{a_1} = true$
$a_2 = \{b\}, g_{a_2} = true$
$a_3 = \{d\}, g_{a_3} = true$





$L = \{l_1, l_2\}$
$P = \{a, b\}$
$X = \{x\}$
$\tau_1 = \{l_1, a, l_2\}, g_{\tau_1} = true$
$\tau_2 = \{l_2, b, l_1\}, g_{\tau_2} = true$

$L = \{l_3, l_4\}$
$P = \{c, d\}$
$X = \{y\}$
$\tau_3 = \{l_3, c, l_4\}, g_{\tau_3} = true$
$\tau_4 = \{l_4, b, l_3\}, g_{\tau_4} = true$

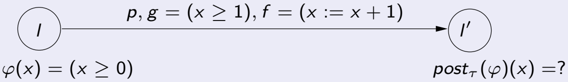
$L = \{(l_1, l_3), (l_2, l_4), (l_1, l_4), (l_2, l_3)\}$
$P = \{ac, b, d\}$
$X = \{x, y\}$
$\tau_1 = \{(l_1, l_3), ac, (l_2, l_4)\}, g_{\tau_1} = true$
$\tau_2 = \{(l_2, l_4), b, (l_1, l_4)\}, g_{\tau_2} = true$
$\tau_3 = \{(l_2, l_4), d, (l_2, l_3)\}, g_{\tau_3} = true$
$\tau_4 = \{(l_1, l_4), d, (l_1, l_3)\}, g_{\tau_4} = true$
$\tau_5 = \{(l_2, l_3), b, (l_1, l_3)\}, g_{\tau_5} = true$

Outline

Definitions

Given a system $B = (L, P, T, X, \{g_\tau\}_{\tau \in T}, \{f_\tau\}_{\tau \in T})$

Post predicate



$$post_\tau(\varphi)(x) = \exists x'.(x' \geq 0) \wedge (x' \geq 1) \wedge (x = x' + 1) = x \geq 2$$

Inductive Invariant and Invariant

a predicate ϕ is:

- an inductive invariant iff $(Init \wedge post(\phi)) \Rightarrow \phi$
- an invariant if there exists an inductive invariant ϕ_0 such that $\phi_0 \Rightarrow \phi$

The Method: The main Idea

Compositional verification rule

$$\frac{B_1 < \Phi_1 >, B_2 < \Phi_2 >, \Psi \in II(B_1 \parallel B_2, \Phi_1, \Phi_2)\ \Phi_1 \wedge \Phi_2 \wedge \Psi \Rightarrow \Phi}{B_1 \parallel B_2 < \Phi >}$$

- Φ is the component invariant of B_i
- Ψ is an interaction invariant of $\gamma(B_1, \ldots, B_n)$ computed from Φ_i and $\gamma(B_1, \ldots, B_n)$
- $\bigwedge_{i=1}^{n} \Phi_i \wedge \Psi$ is an over-approximation of reachable states of system

Automatic Generation Of Invariants

We provide heuristics for computing two types of invariants:

Component Invariants

Invariants for atomic components are generated by simple forward analysis of their behavior.

- over-approximations of the set of their reachable states.

Interaction Invariants

Invariants that characterize constraints on the global state space induced by synchronizations between components.

- Generalizations of the notions of trap in Petri nets.

Automatic Generation Of Invariants

We provide heuristics for computing two types of invariants:

Component Invariants

Invariants for atomic components are generated by simple forward analysis of their behavior.

- over-approximations of the set of their reachable states.

Interaction Invariants

Invariants that characterize constraints on the global state space induced by synchronizations between components.

- Generalizations of the notions of trap in Petri nets.

Automatic Generation Of Invariants

We provide heuristics for computing two types of invariants:

Component Invariants

Invariants for atomic components are generated by simple forward analysis of their behavior.

- over-approximations of the set of their reachable states.

Interaction Invariants

Invariants that characterize constraints on the global state space induced by synchronizations between components.

- Generalizations of the notions of trap in Petri nets.

Computing component invariants

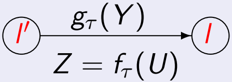
Definition (Inductive invariants)

Given a system $\langle B, Init \rangle$, the following iteration defines a sequences of increasingly stronger inductive invariants

$$\phi_0 = true \quad \phi_{i+1} = Init \vee post(\phi_i)$$

Efficient computation of component invariants

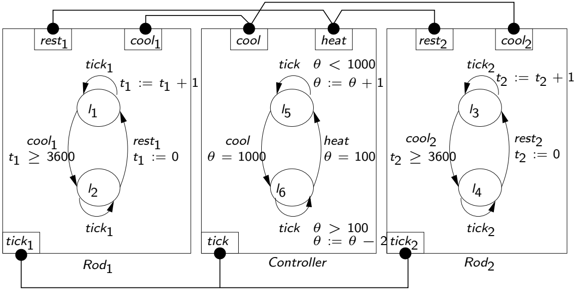
- Precise computation of post requires quantifier elimination
- An alternative is to compute over-approximations of post based on syntactic analysis of the predicates



For a predicate φ find $\varphi = \varphi_1(Y_1) \wedge \varphi_2(Y_2)$ such that $Y_2 \cap Z = \emptyset$

$$post^a_\tau(\varphi) = \varphi_2(Y_2) \wedge \left\{ \begin{array}{ll} g_\tau(Y) & \text{if } Z \cap Y = \emptyset \\ true & \text{otherwise} \end{array} \right\} \wedge \left\{ \begin{array}{ll} Z = f_\tau(U) & \text{if } Z \cap U = \emptyset \\ true & \text{otherwise} \end{array} \right\}$$

Component Invariants: An example

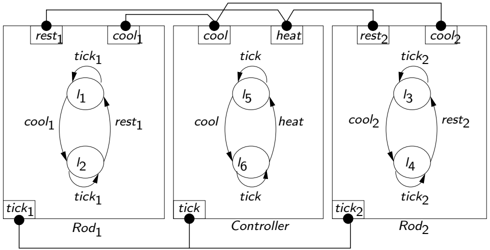


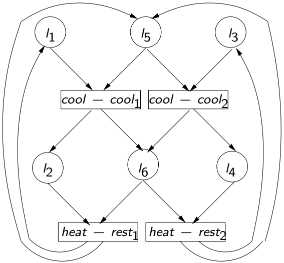
Example

For the Temperature Control System, 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 component invariants of the atomic components Rod1, Rod2 and Controller. □

Computing interaction invariants of finite systems

Trap: a set of places, if they have initially a token, thay will always have a token





The set of implications

$l_1 \Rightarrow l_2 \vee l_6$ $\quad l_2 \Rightarrow l_1 \vee l_5$

$l_3 \Rightarrow l_4 \vee l_6$ $\quad l_4 \Rightarrow l_3 \vee l_5$

$l_5 \Rightarrow (l_2 \vee l_6) \wedge (l_4 \vee l_6)$ $\quad l_6 \Rightarrow (l_1 \vee l_5) \wedge (l_3 \vee l_5)$

$\Psi_1 = \{l_1, l_3, l_6\}$ and $\Psi_2 = \{l_2, l_4, l_5\}$ are solutions (traps)

Interaction invariant is $II = (l_1 \vee l_3 \vee l_6) \wedge (l_1 \vee l_3 \vee l_6)$

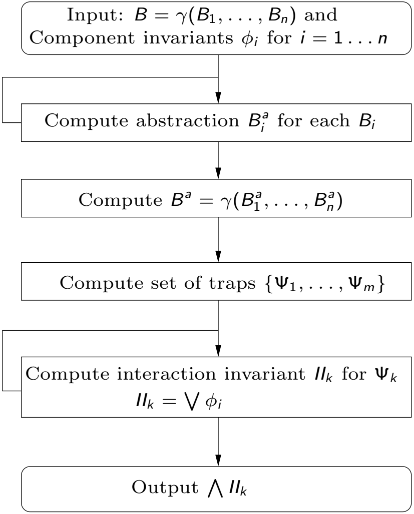
Computing Interaction Invariants of infinite systems

Main Idea

Given 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.

1. First, for each component B_i and its associated invariant Φ_i, we define a finite state abstraction α_i and compute an abstract transition system $B_i^{\alpha_i}$.
2. Then, we compute interaction invariants for $\mathcal{S}$ by analyzing, without constructing explicitely the state space, the parallel composition $B^{\alpha} = \gamma(B_1^{\alpha_1}, \ldots, B_n^{\alpha_n})$.

Computing interaction invariants of infinite systems



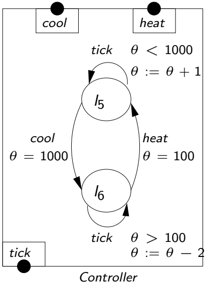
Abstraction

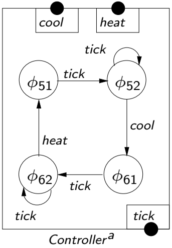
Abstract states constructed from Component Invariants of the Controller

$$\phi_{Controller} = \phi_5 \vee \phi_6$$

$$\phi_5 = at\_l_5 \wedge (\theta = 100 \vee 101 \leq \theta \leq 1000) = \underbrace{(at\_l_5 \wedge \theta = 100)}_{\phi_{51}} \vee \underbrace{(at\_l_5 \wedge 101 \leq \theta \leq 1000)}_{\phi_{52}}$$

$$\phi_6 = at\_l_6 \wedge (\theta = 1000 \vee 100 \leq \theta \leq 998) = \underbrace{(at\_l_6 \wedge \theta = 1000)}_{\phi_{61}} \vee \underbrace{(at\_l_6 \wedge 100 \leq \theta \leq 998)}_{\phi_{62}}$$





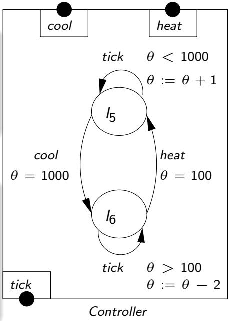
Outline

Definitions

Predicate **en** of a port

en is a set of states from which this port is enabled

$en(tick) = (at\_l_5 \wedge \theta < 1000) \vee (at\_l_6 \wedge \theta > 100)$



Predicate DIS_a of an interaction $a = \{a_1, \ldots, a_n\}$

DIS_a is a set of states from which this interaction is disabled

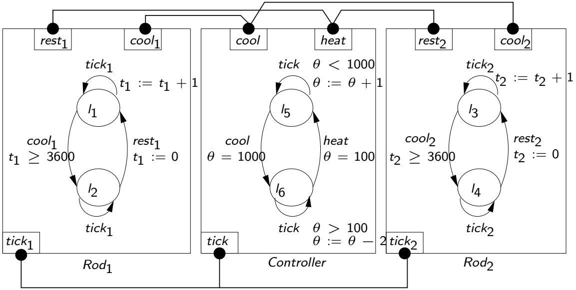
$$DIS_a = \neg \bigwedge_{i=1}^{n} en(p_i)$$

Predicate DIS of a system $\langle \gamma(B_1, \ldots, B_n) \rangle$

DIS is a set of states from which all the interactions are disabled

$$DIS = \neg \bigwedge_{a \in \gamma} DIS_a$$

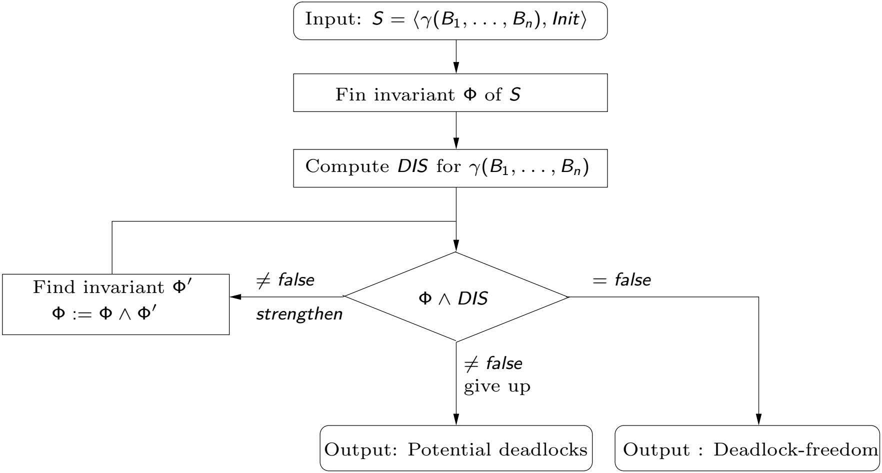
Temperature Control Sytem: DIS



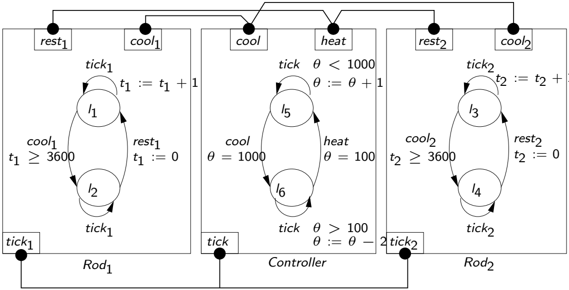
DIS state of Temperature Control System

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

Algorithm for detecting deadlocks

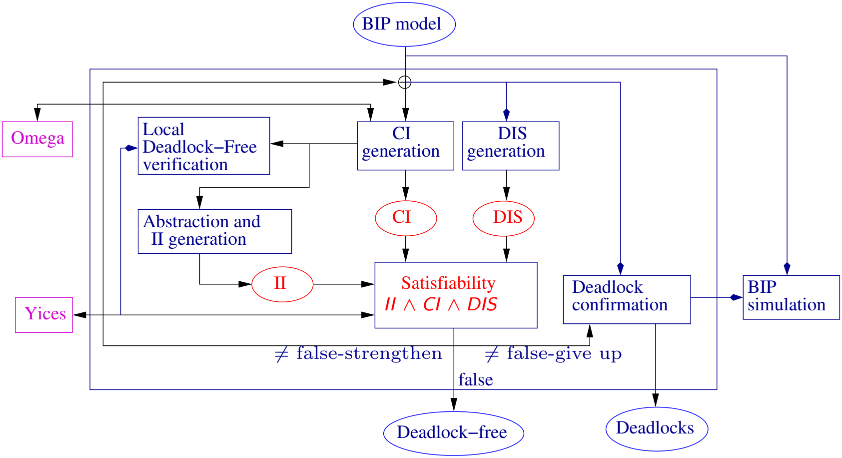


Temperature Control Sytem



Step	Deadlocks
$CI \wedge DIS$	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. $(at\_l_2 \wedge t_1 \geq 3600) \wedge (at\_l_4 \wedge t_2 \geq 3600) \wedge (at\_l_5 \wedge \theta = 1000)$
$CI \wedge II \wedge DIS$	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. $(at\_l_2 \wedge t_1 \geq 3600) \wedge (at\_l_3 \wedge 1 \leq t_2 < 3600) \wedge (at\_l_5 \wedge \theta = 1000)$

D-Finder


BIP model
Omega
Local Deadlock–Free verification
CI generation
DIS generation
Abstraction and II generation
CI
DIS
II
Satisfiability
II ∧ CI ∧ DIS
Yices
Deadlock confirmation
BIP simulation
≠ false-strengthen
≠ false-give up
false
Deadlock–free
Deadlocks

Case Studies

example	n	q	x_b	x_i	D	D_c	D_{ci}	t
Waterflow Control	4	8	2	0	11	11	0	0m1s
R-W(50 readers)	52	106	0	1	$\sim 10^{15}$	$\sim 10^{15}$	0	1m15s
R-W(100 readers)	102	206	0	1	$\sim 10^{30}$	$\sim 10^{30}$	0	15m28s
R-W(130 readers)	132	266	0	1	$\sim 10^{39}$	$\sim 10^{39}$	0	29m13s
T. Control(2 Rods)	3	6	0	3	8	5	3	0m3s
T. Control(4 Rods)	5	10	0	5	32	17	15	1m5s
UTSv1 (4 Cars, 9 UCal)	14	45	4	30	82961	41488	0	1m42s
UTSv3 (8 Cars, 16 UCal)	25	91	8	58			0	22m2s

- n — number of BIP components in example
- q — total number of control locations
- x_b — total number of boolean variables
- x_i — total number of integer variables
- D — estimated number of potential deadlocks configurations in *DIS*
- D_c — number of potential deadlock configurations remaining in $DIS \wedge CI$
- $D_c i$ — number of potential deadlock configurations remaining in $DIS \wedge CI \wedge II$

Outline

Conclusions and future work

Conclusions

- Innovates: using interaction invariant to characterize contexts of individual components.
- Efficiently combines two types of invariants (invariants of atomic components and interaction invariants).
- Using only lightweight analysis techniques
- No restrictions on the type of data as long as we stay within theories for which there exist efficient decision procedures.
- Can be adapted to interactions with data transfer

Current and future work

- Prove safety properties other than deadlock-freedom.
- Generate inductive invariant to eliminate potential deadlocks [Bradley and Manna, 2007]
- Adapt to interactions with data transfer

Reference

Bradley, A. R. and Manna, Z. (2007).
Checking safety by inductive generalization of counterexamples to induction.
In *FMCAD*, pages 173–180.

Cobleigh, J. M., Avrunin, G. S., and Clarke, L. A. (2008).
Breaking up is hard to do: An evaluation of automated assume-guarantee reasoning.
ACM Transactions on Software Engineering and Methodology, 17(2).