Adaptor synthesis for real-time components

Massimo Tivoli, Pascal Fradet, **Alain Girault**, and Gregor Goessler

University of l'Aquila and INRIA POP-ART team

L'Aquila (ITALY) and Grenoble (FRANCE)

Outline

- Introduction
- Component model
- Adaptor synthesis
- Concluding remarks

Introduction

Propose a lightweight component model:

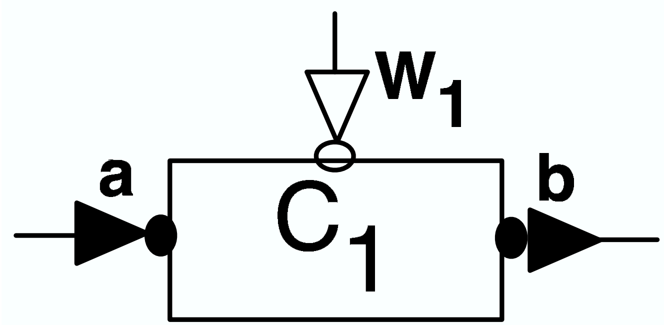
➪ To express real-time interaction properties

Develop a method to check incompatibilities between components (due to timing and/or protocol mismatch) and to produce adaptors incrementally

Implement this method inside a tool chain

Component model

Example 1:



A component has input ports (**a**), output ports (**b**), and one clock port ($\mathbf{w}_1$)

Interactions between two connected components is synchronous

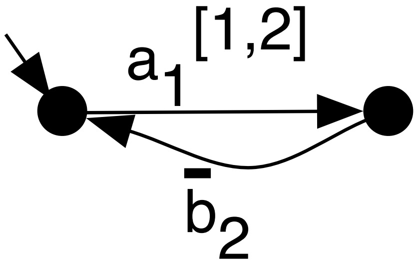
➪ Reading and writing operations are blocking

The clock port is a special input port $\in \{0, 1\}$ that tells whether the component is enabled (1) or disabled (0)

➪ Connected at the instanciation to a periodic clock

Component interaction protocol

Example 1:



The interaction protocol is specified as a LTS (fig A) with actions of the form: $x_l^{\{u\}[i,j]}$

controllability tag → $\{u\}$

duration (local time units) → $[i,j]$

action label (a is a read / $\bar{b}$ is a write) → x

latency (global time units) → l

The component is instanciated with a periodic clock (e.g. (10)): when disabled it can only let the global time elapse, when enabled it can also perform an action

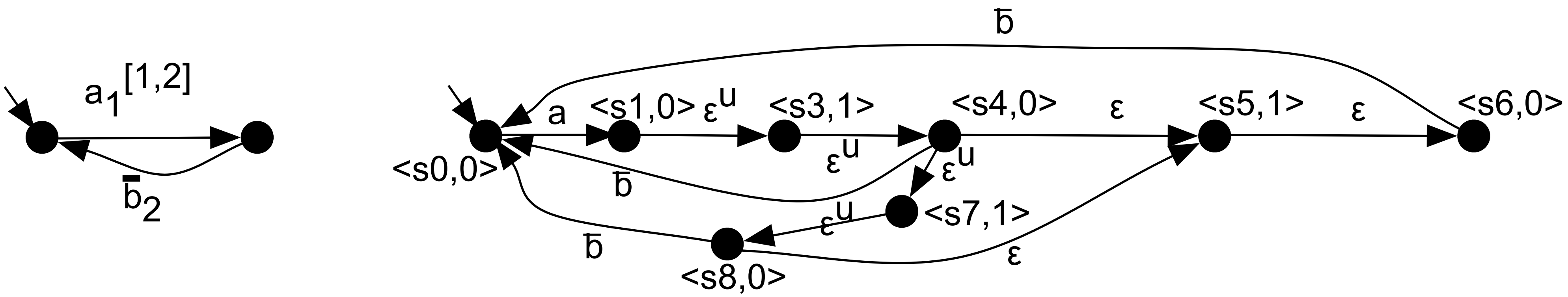
Component semantics

Latency is a QoS requirement:

➪ $p \xrightarrow{x_1} q$ is translated into $p \xrightarrow{\varepsilon} p' \xrightarrow{x} q$ and $p \xrightarrow{x} q$

Duration is a constraint of the execution platform:

➪ $p \xrightarrow{x^{[1,2]}} q$ with clock (1) is translated into $p \xrightarrow{x} p'$, $p' \xrightarrow{\varepsilon^u} p'' \xrightarrow{\varepsilon^u} q$ and $p' \xrightarrow{\varepsilon^u} q$

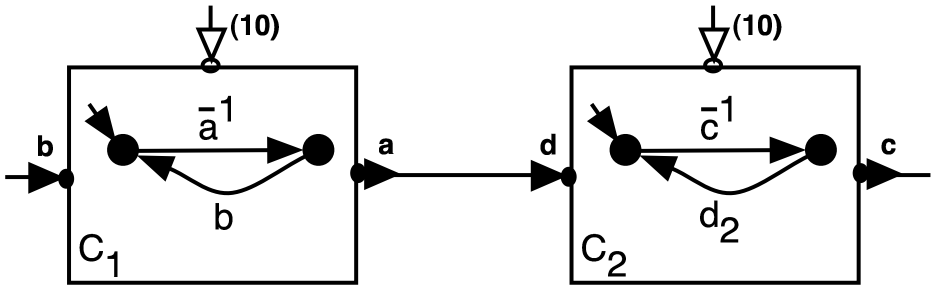


(A) C_1 (B) $\langle C_1,(10)\rangle$

Each state is labelled with the global time instant modulo the periodic clock's length (e.g., $\langle s_4, 0\rangle$ and $\langle s_5, 1\rangle$)

Example 2: A very simple assembly

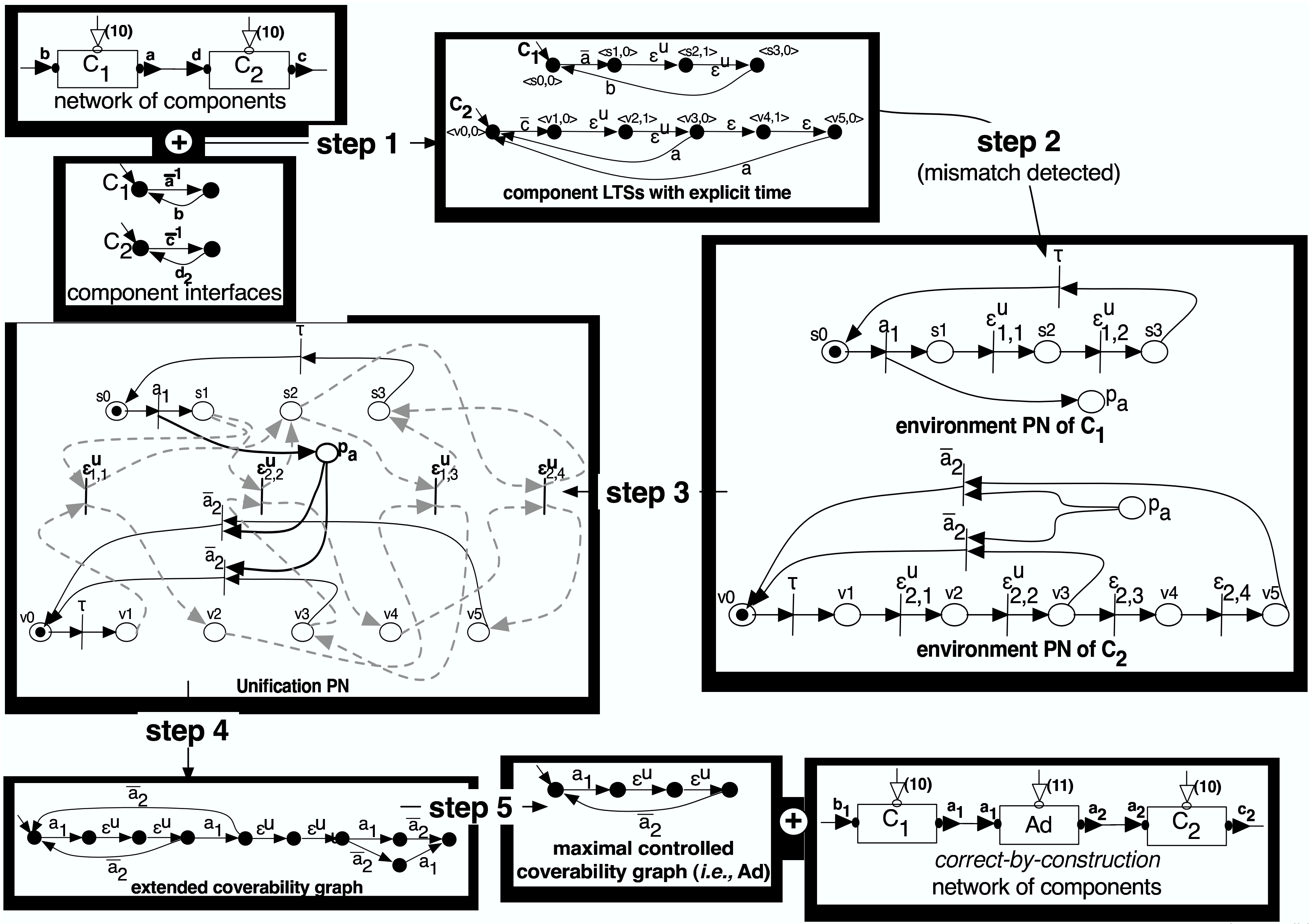
Assembly put together components and specify ports renaming



Here, the interaction protocols do not match

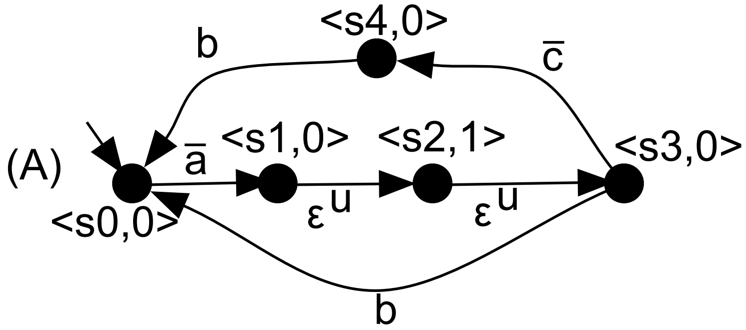
Proposal: to generate automatically a correct-by-construction and bounded adaptor in order to solve protocol/latency/duration/clock inconsistencies

Method overview with example 2

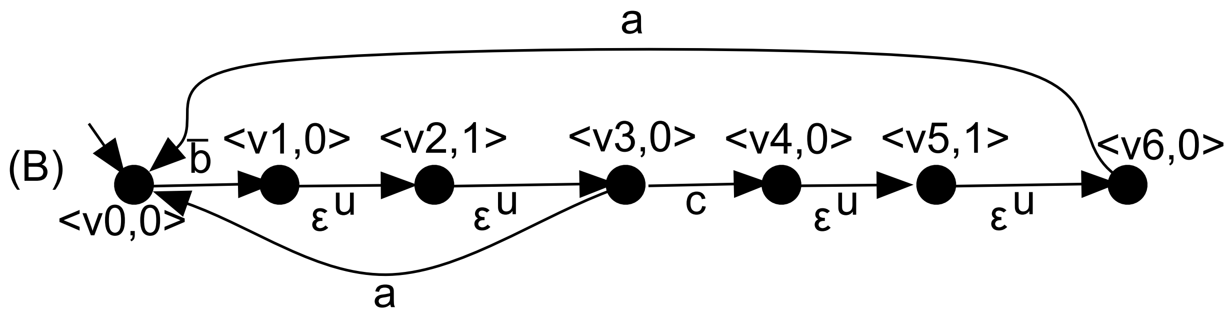


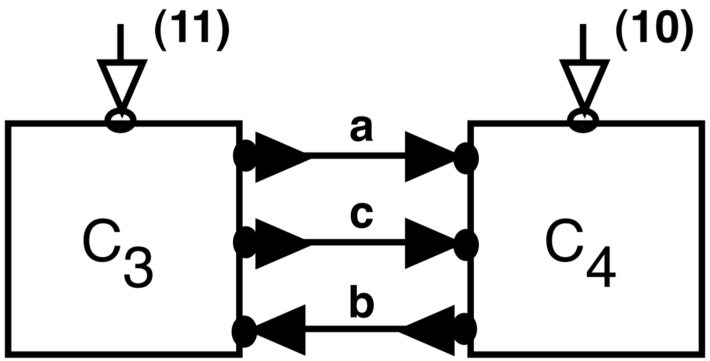
Example 3

Semantics LTS of $\langle C_3, (11) \rangle$



Semantics LTS of $\langle C_4, (10) \rangle$





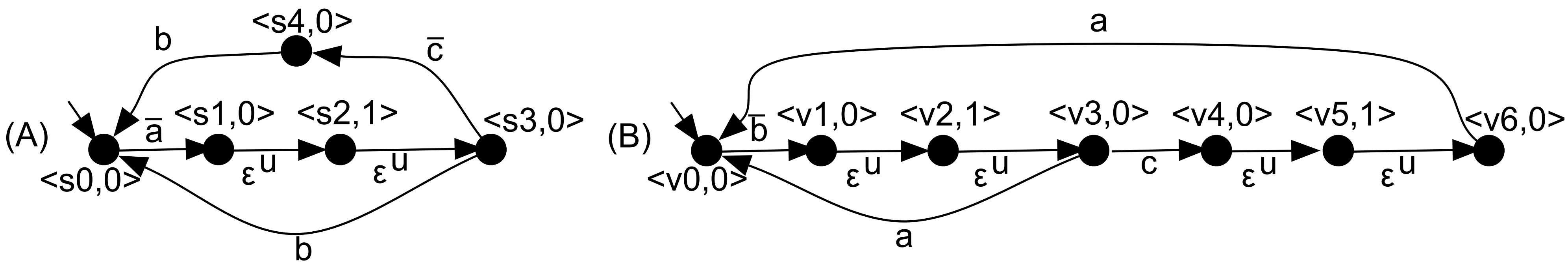
Step 2: mismatch detection

We compute the synchronous parallel composition of the LTSs

Deadlocks are sink states in this parallel composition

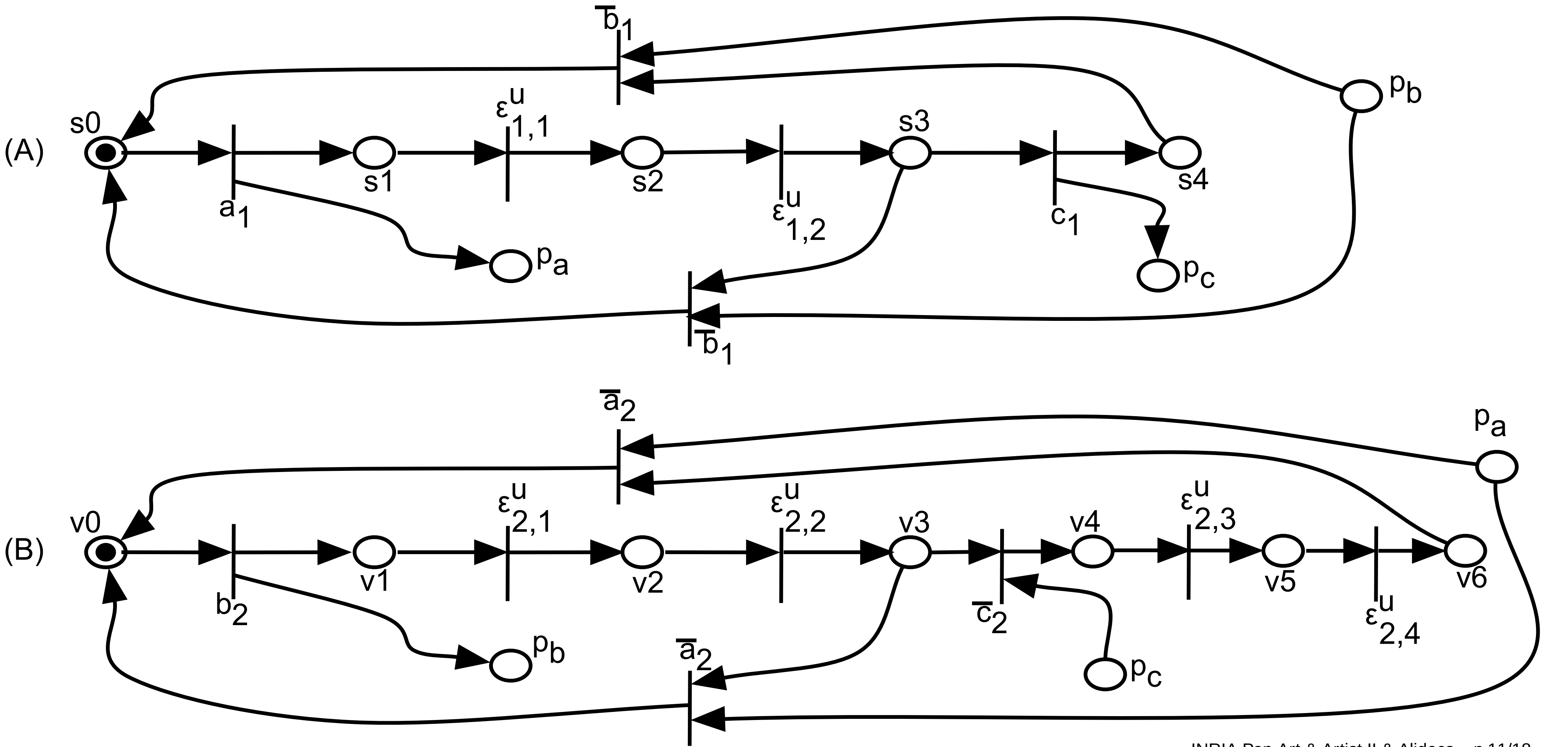
We use CADP to detect such sink states

In example 3, the initial state of the synchronous product is a sink state:



Step 3.1: component's expectation

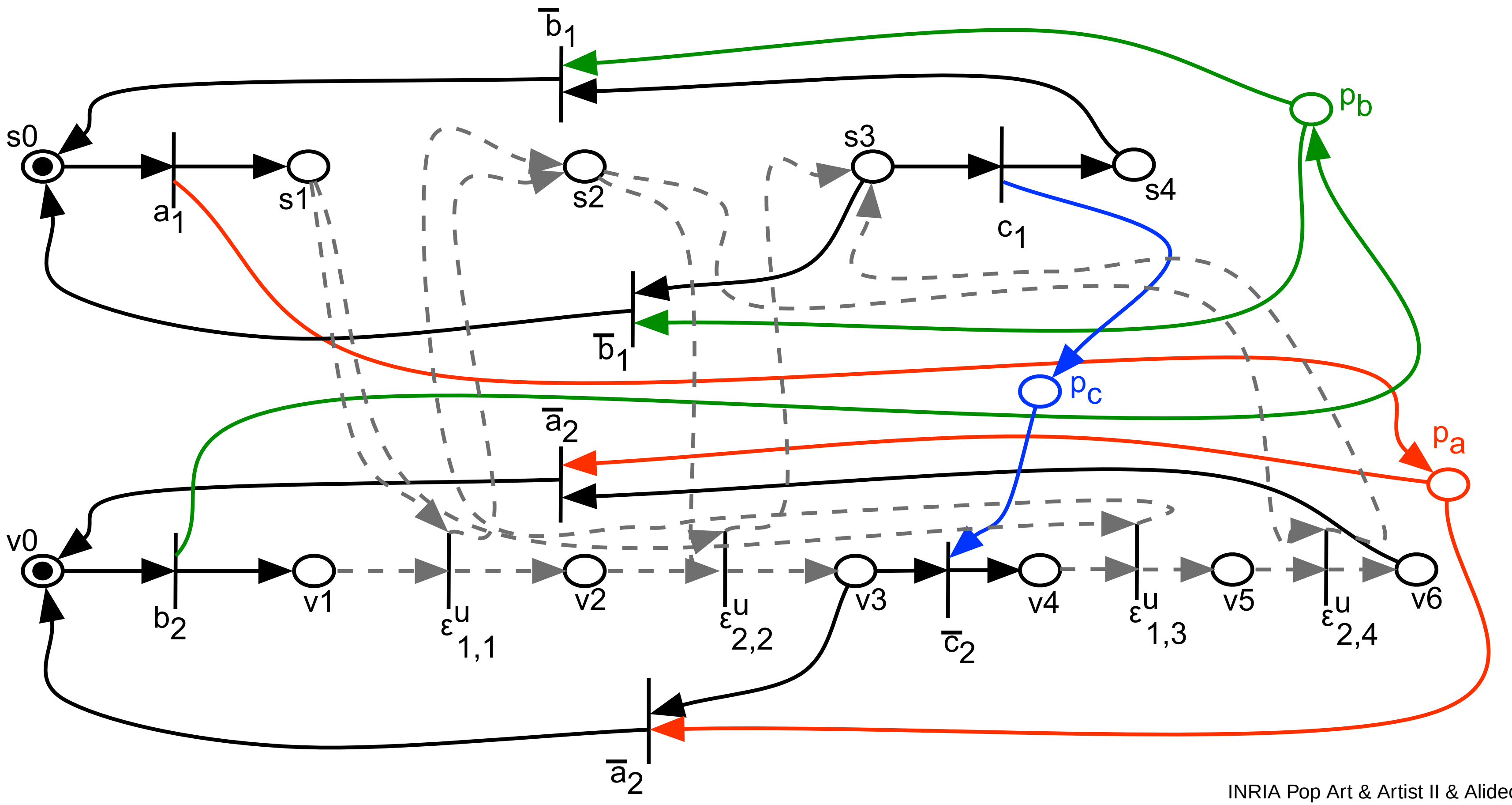
For each component, we compute one the Petri Net modeling what the component expects from its environment (i.e., the other components plus the environment)



Step 3.2: unification PN

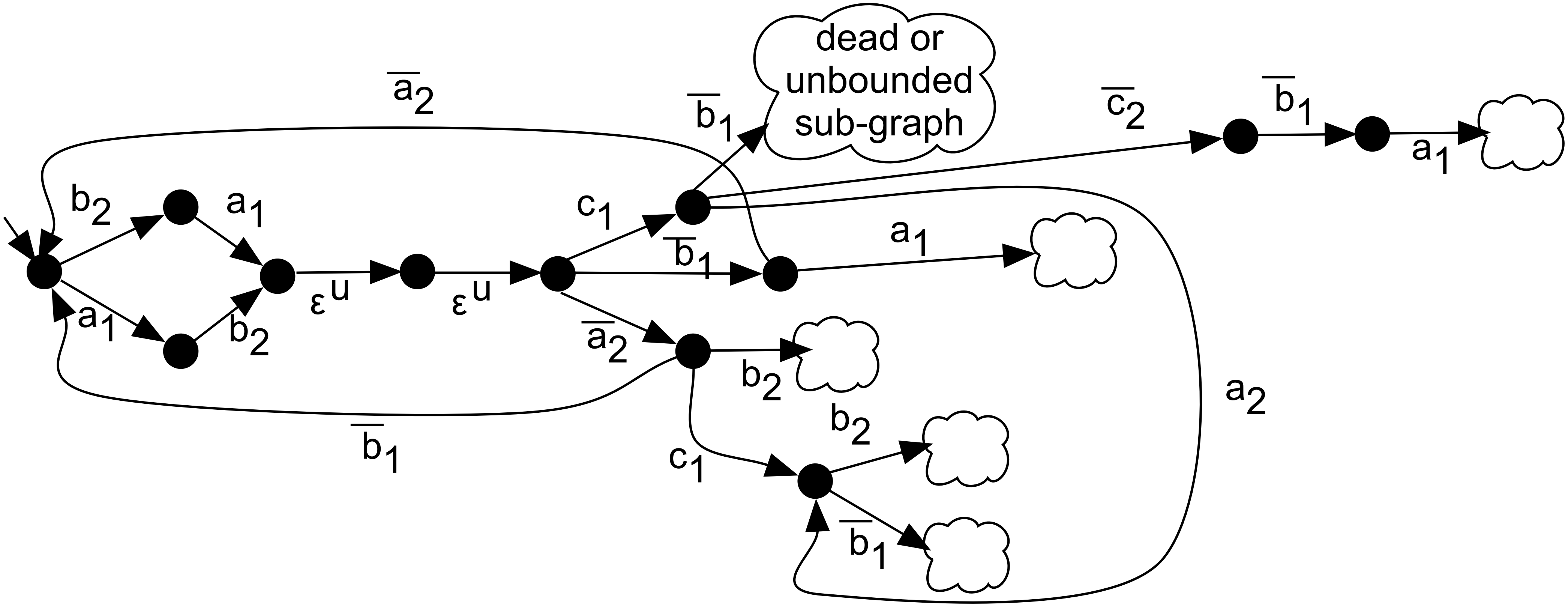
We build the causal dependencies (read/write) by unifying the pending places

And we compose synchronously the time-elapsing transitions (only the firable ones are shown)



Step 4: extended coverability graph

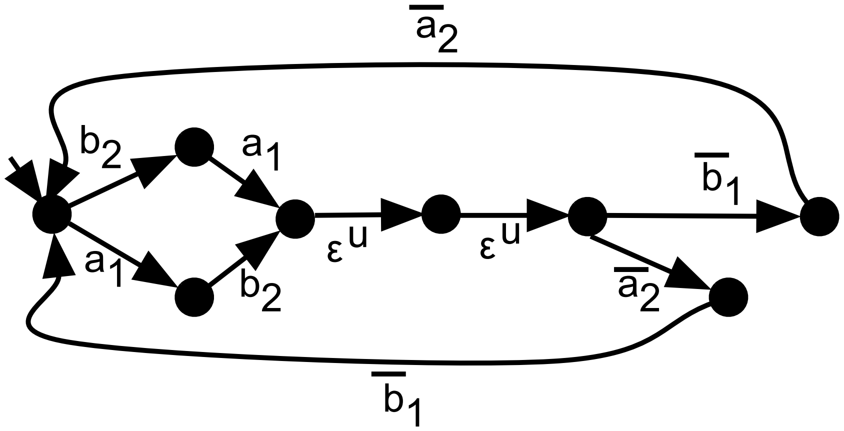
Each node is a new marking of the unification PN



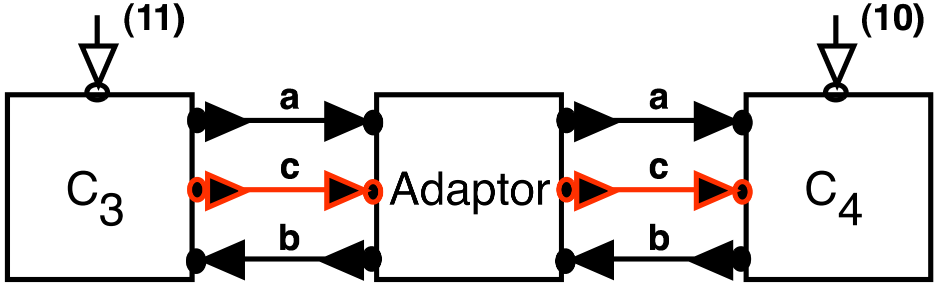
The “cloud-nodes” represent parts of the graph that contain only dead nodes or nodes with infinite markings

Step 5: controlled coverability graph

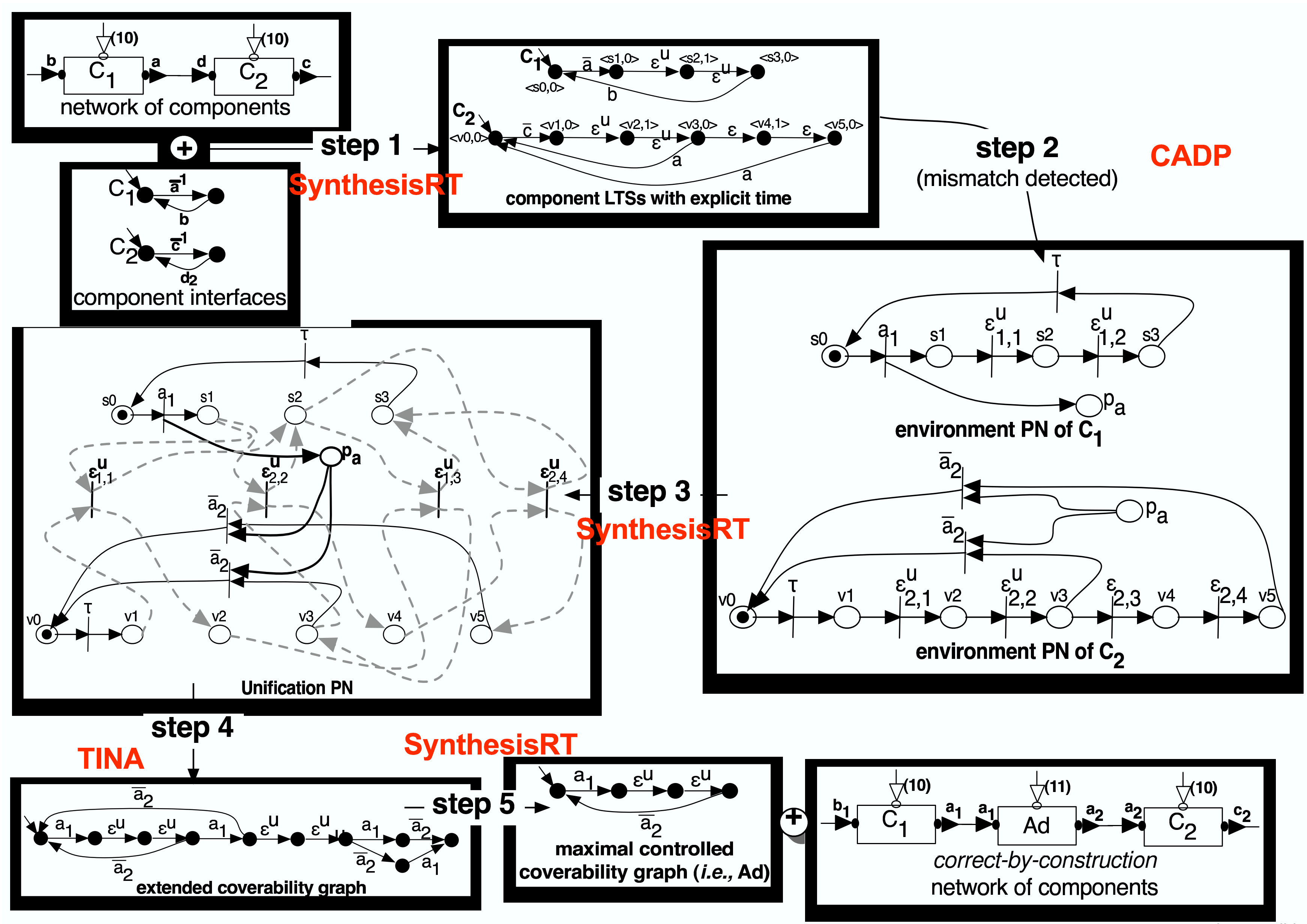
We prune the transitions that lead inevitably to cloud-nodes and that are controllable:



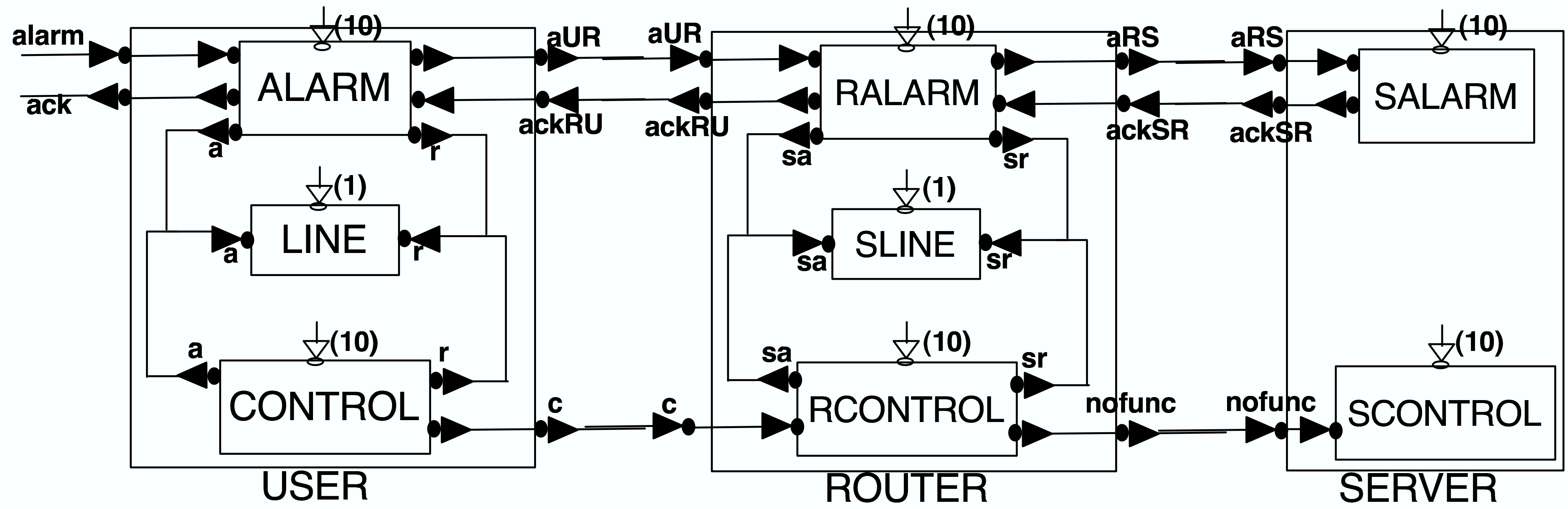
This is the correct by construction and bounded adaptor to be put between C3 and C4:



Tool chain


network of components
step 1
SynthesisRT
component interfaces
component LTSs with explicit time
step 2
(mismatch detected)
CADP
environment PN of C1
environment PN of C2
step 3
SynthesisRT
Unification PN
step 4
TINA
extended coverability graph
SynthesisRT
step 5
maximal controlled coverability graph (i.e., Ad)
correct-by-construction
network of components

Example 4: A case study



Remote medical care system (RMCS) for monitoring and assistance to disabled people

Related work

Component adaptation with PNs and the TINA tool
[Canal et al, FMOODS’06]

Synchronizing of different ultimately periodic clocks
[Cohen et al, EMSOFT’05]

Component interface compatibility [Passerone et al, ICCAD’02]

Extended coverability graph with a termination criterion
[Cortadella et al, IEEE TCAD’05]

Discrete controller synthesis [Ramadge & Wonham, Proc. IEEE’89]

Concluding remarks

Powerful features to specify key real-time properties: latency, duration, controlability, and clock

(Associated programming language DLiPA, a process algebra)

A correct by construction and bounded adaptor is synthesized automatically whenever two components interfaces do not match

Tool chain: SynthesisRT + CADP + TINA

Open problems: building incremental adaptor, clock independent adaptors...