THE FSM DOMAIN

main contributors
Alain Girault
Bilung Lee
Edward Lee

other contributors
Wan-Teh Chang
Stephen Edwards

OUTLINE

1. Motivations

2. Specification of FSMs

3. Embedding of FSMs:
   - into the SDF domain
   - into the DDF domain
   - into the SR domain

4. Adding hierarchy

5. Comparison with ARGOS

MOTIVATIONS

Control: clean control structure in PTOLEMY

Heterogeneity: keep the usual PTOLEMY philosophy: a new domain

DOMAIN of VALUES

Each signal has 3 states: “unknown” ($\perp$), “absent” (ε) and “present with value v” (v)

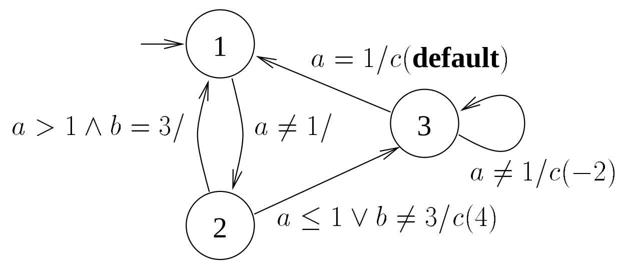
SPECIFICATION

$$\langle I, O, Q, q_0, T \rangle$$

where:

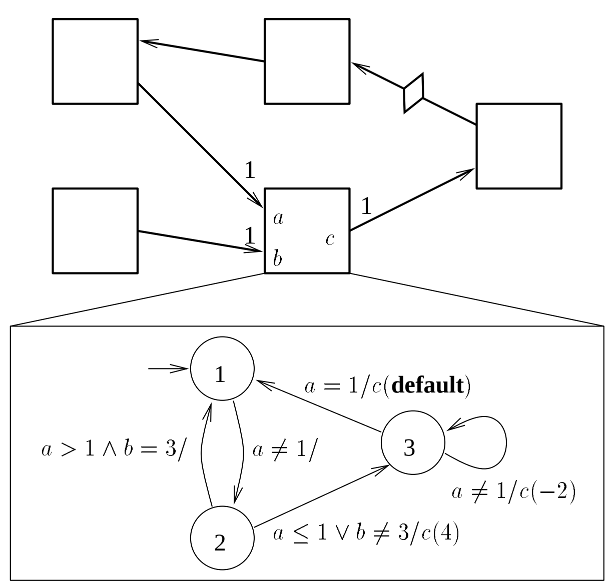
- I is a set of input signals,
- O is a set of output signals,
- Q is a set of states,
- $q_0 \in Q$ is the initial state, and
- T is a set of transitions of the form $guard\_part/action\_part$

EXAMPLE



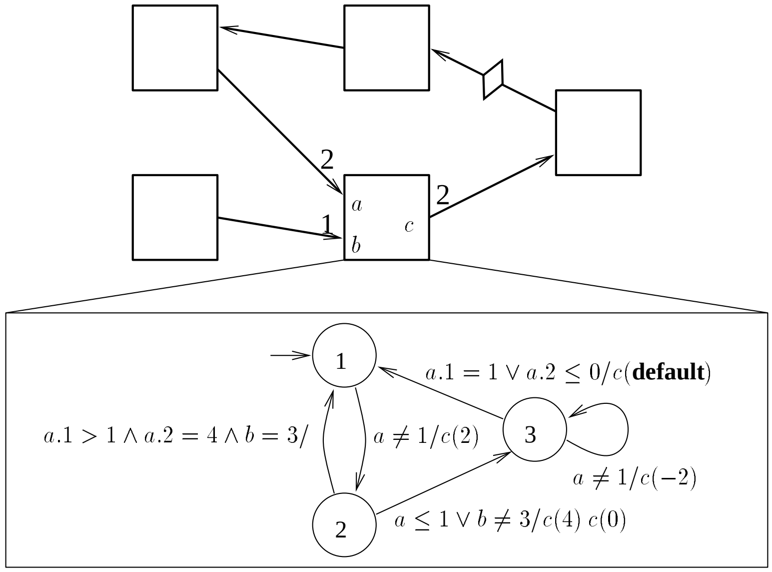
current state	1	2	3	3	3	1	1	2	…
a	0	0	3	8	1	1	7	4	…
b	1	5	0	-4	5	4	1	2	…
next state	2	3	3	3	1	1	2	3	…
c	ε	4	-2	-2	0	ε	ε	4	…

EMBEDDING into SDF



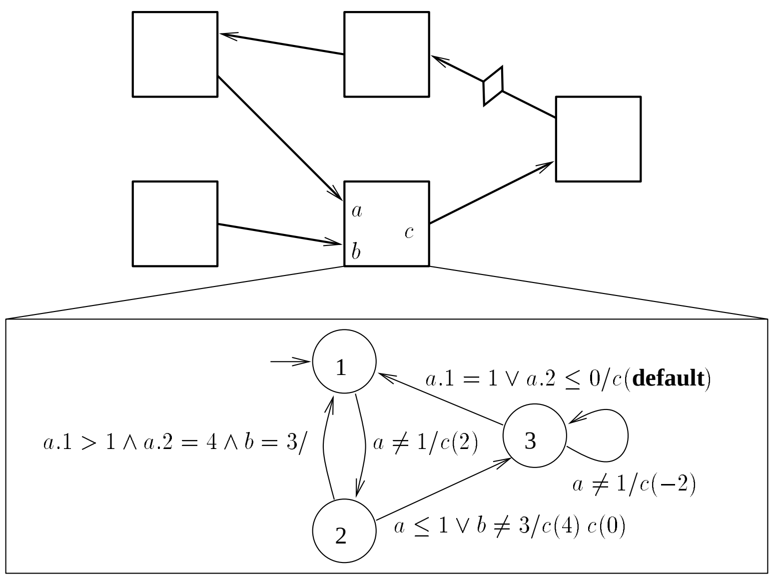
SDF semantics: a token **must** be produced on transition from state 1 to state 2 $\Rightarrow$ label $a \neq 1/c$(**default**)

NON HOMOGENEOUS CASE



SDF semantics: two tokens **must** be produced on transition from state 1 to state 2 $\Rightarrow$ label $a \neq 1/c(2)\ c($**default**$)$

EMBEDDING into DDF



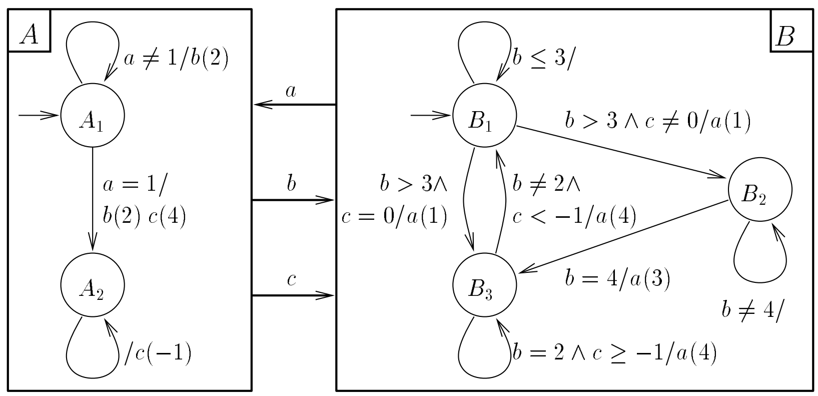
DDF semantics: additional tokens are **never** produced
Need to consume a number of token sufficient to evaluate all the guards

EMBEDDING into SR

Handling instantaneous loops:

1. partial order on the values of the wires
2. block functions must be monotonic w.r.t. this partial order

Partial evaluation algorithm for FSMs



PARTIAL EVALUATION

state 1: functions $\mathcal{O}_b^1$ and $\mathcal{O}_c^1$

a	$-\infty$	1	$+\infty$
b	2	2	2
c	ε	4	ε

For any state s and any output o:

$$\begin{cases} \text{if } \exists v : \forall x, \mathcal{O}_o^s(x) = v \text{ then } \mathcal{O}_o^s(\bot) = v \\ \qquad\qquad\qquad\qquad\quad \text{else } \mathcal{O}_o^s(\bot) = \bot \end{cases}$$

state 1: extended functions $\mathcal{O}_b^1$ and $\mathcal{O}_c^1$

a	$-\infty$	1	$+\infty$	$\bot$
b	2	2	2	2
c	ε	4	ε	$\bot$

EXECUTION EXAMPLE

1. start with $(a, b, c) = (\bot, \bot, \bot)$
2. execute B: $a = \mathcal{O}_a^1(\bot, \bot) = \bot$
3. execute A: $b = \mathcal{O}_b^1(\bot) = 2$ and $c = \mathcal{O}_c^1(\bot) = \bot$
4. execute B: $a = \mathcal{O}_a^1(2, \bot) = 1$
5. execute A: $b = \mathcal{O}_b^1(1) = 2$ and $c = \mathcal{O}_c^1(1) = 4$

Theorem: The extended output functions are monotonic w.r.t. to the partial order

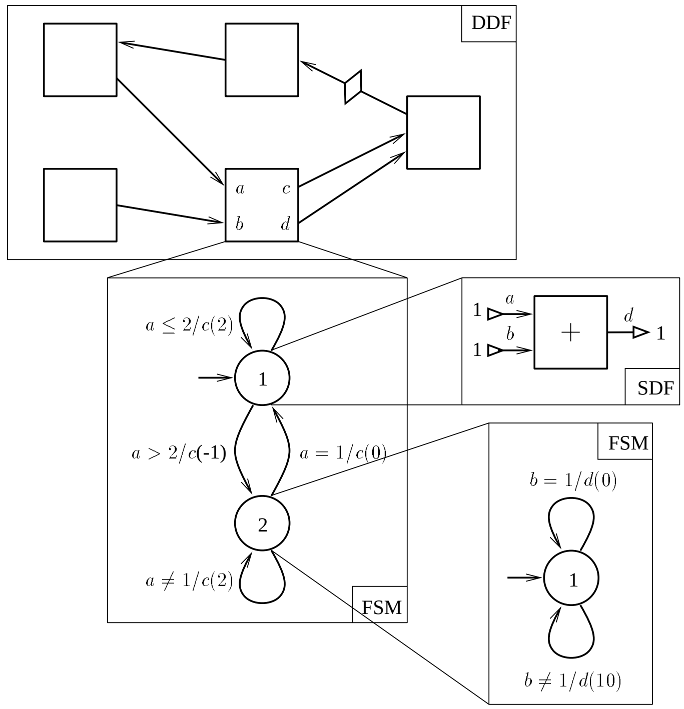
ADDING HIERARCHY

Executing an FSM: execute the current state & select and fire a transition (same set of inputs, distinct sets of outputs, internal events)

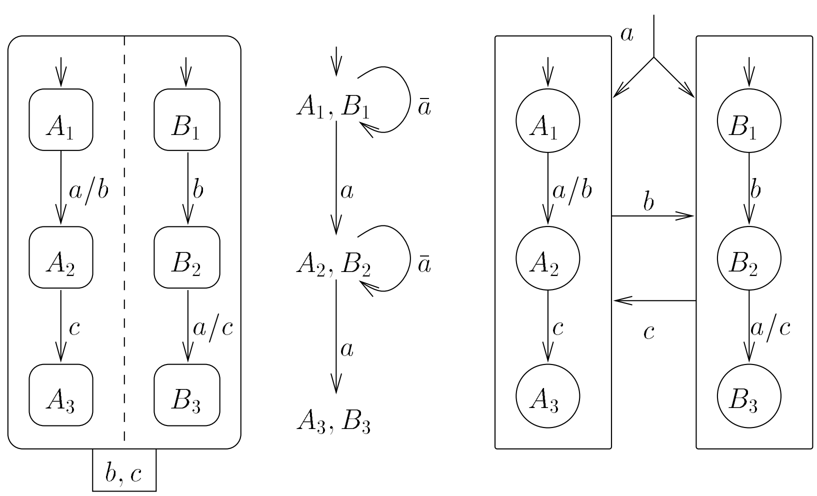
Executing a data-flow network: perform a complete execution cycle of the network (hierarchical blocks are executed according to their respective domain)

Executing an SR network: find the behavior (least fixed point) of the network (each block must compute a monotonic function)

HIERARCHICAL EXAMPLE



COMPARISON with ARGOS



(a) (b) (c)

The SR scheduler will generate $A.B.A$ or $B.A.B$

In both cases, the network goes from state (A_1, B_1) to (A_2, B_2)