Symbolic communicating automata

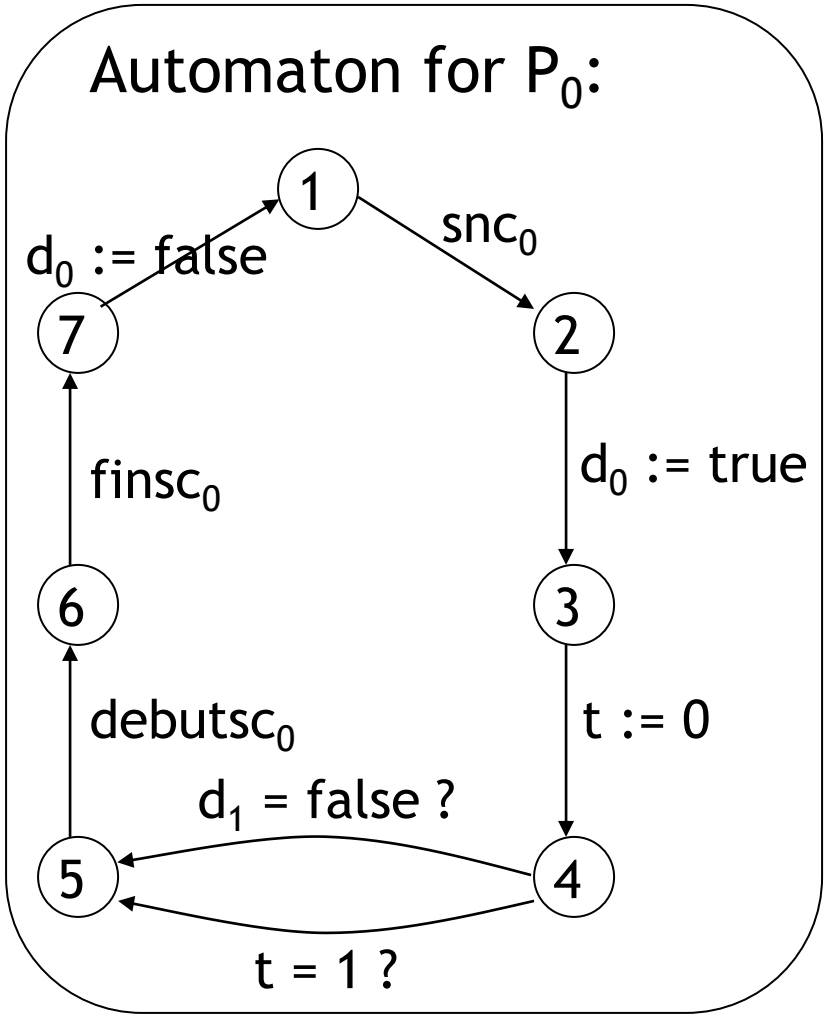
Reminder: the Peterson algorithm

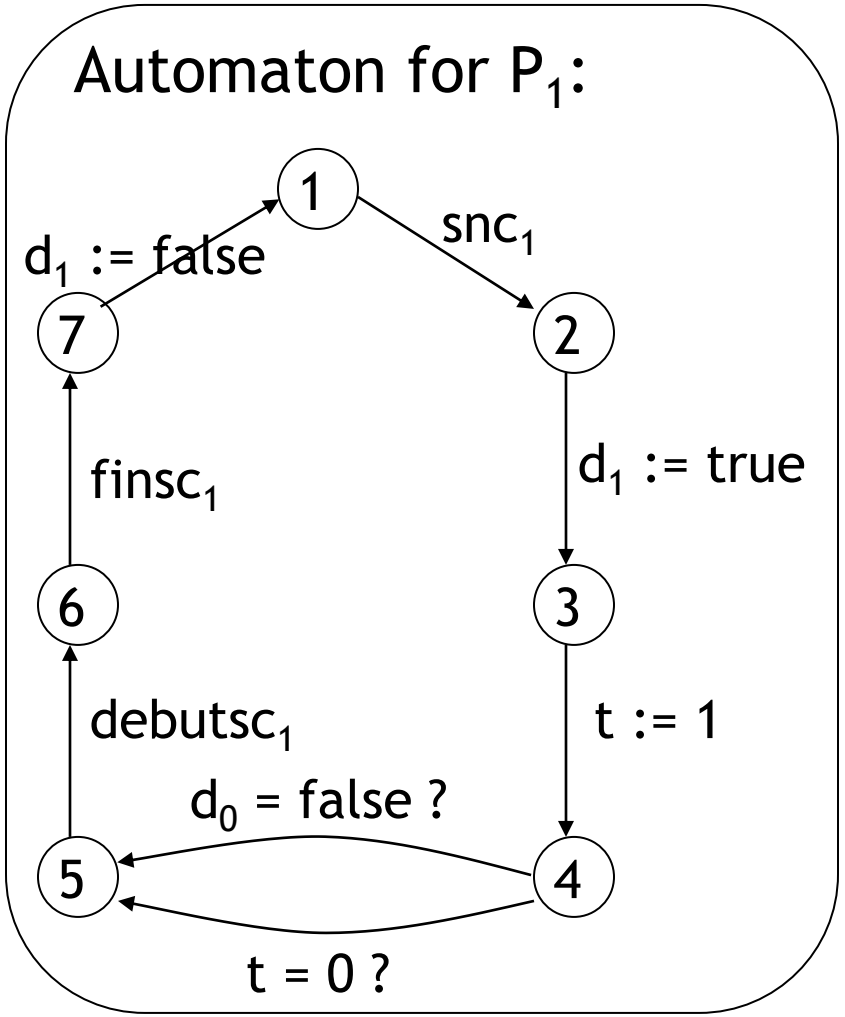
var d_0 : bool := false { read by P_1, written by P_0 }
var d_1 : bool := false { read by P_0, written by P_1 }
var t $\in$ {0, 1} := 0 { read/written by P_0 and P_1 }

loop forever { P_0 }
1 : { snc0 }
2 : d_0 := true
3 : t := 0
4 : **wait** (d_1 = false **or** t = 1)
5 : { $debutsc_0$ }
6 : { $finsc_0$ }
7 : d_0 := false
endloop

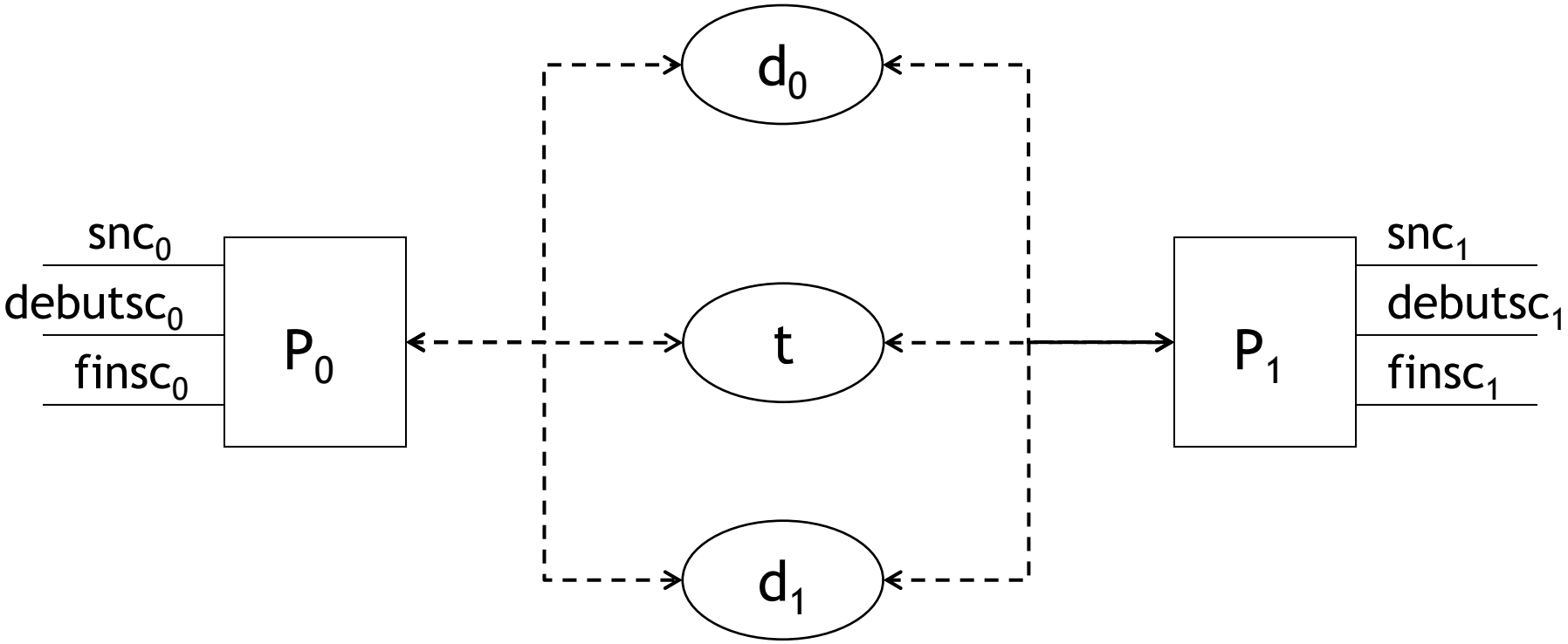
loop forever { P_1 }
1 : { snc_1 }
2 : d_1 := true
3 : t := 1
4 : **wait** (d_0 = false **or** t = 0)
5 : { $debutsc_1$ }
6 : { $finsc_1$ }
7 : d_1 := false
endloop

Symbolic automata for P_0 and P_1





Symbolic-system architecture



- Shared global variables (built-in): d_0, d_1, t
- No synchronized action
- Non-synchronized actions: snc_0, $debutsc_0$...

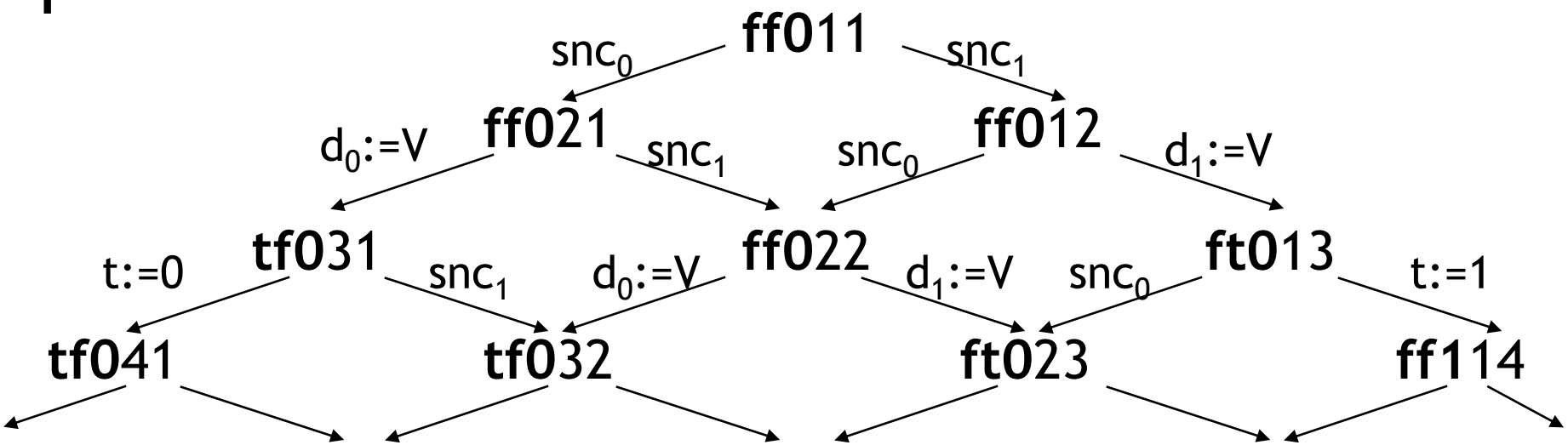
Product automaton

$S = (\{\mathbf{f}, \mathbf{t}\} \times \{\mathbf{f}, \mathbf{t}\} \times \{\mathbf{0}, \mathbf{1}\}) \times (\{1..7\} \times \{1..7\})$

$A = \{ \mathrm{snc}_0, \mathrm{snc}_1, \ldots, d_0 := \mathrm{true}, \ldots \}$

$s_0 = \langle\langle \mathbf{f}, \mathbf{f}, \mathbf{0}\rangle, \langle 1, 1\rangle\rangle = \mathbf{ff011}$

$T =$



Synthesis about symbolic CA

Advantages:

- Simpler model to describe parallelism
- More concise than simple CA
- Available CA manipulation tools:
  - Uppaal http://www.uppaal.org
    at ENSIMAG : /matieres/5MMMVSCT/UPPAAL/uppaal
- Some industrial applications

Limitations:

- State space explosion
- Hard to express: A then (B || C) then D

Beware modeling correctness (shared variables):
atomic read/write vs read followed by write