CS Seminar

23rd March 2012

Parameter Synthesis for Hierarchical Concurrent Real-Time Systems

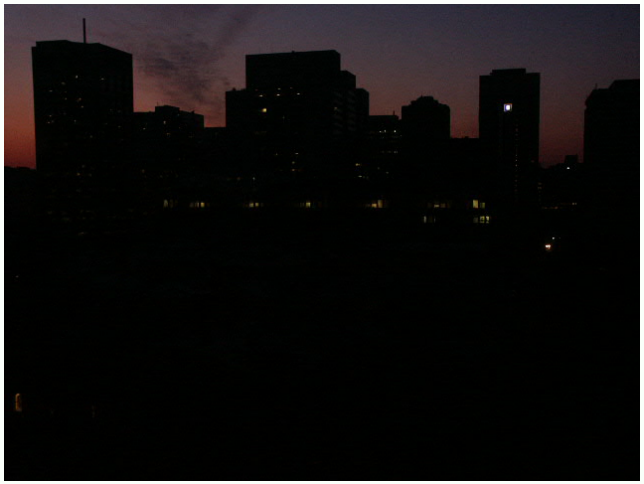
Étienne ANDRÉ

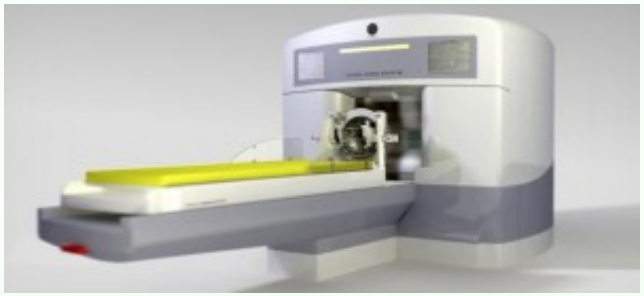
Laboratoire d'Informatique de Paris Nord
Université Paris XIII, France

Joint work with J.-S. Dong, L. Fribourg, Y. Liu, R. Soulat, J. Sun

Verification of Real Time Systems

- Motivation





Context: Model Checking Timed Systems (1/2)

- Input

A timed concurrent system

Context: Model Checking Timed Systems (1/2)

- Input

A timed concurrent system

A good behavior expected for the system

Context: Model Checking Timed Systems (1/2)

- Input

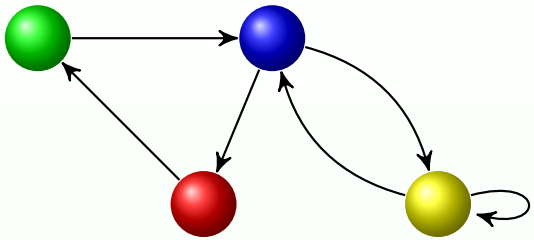
A timed concurrent system

A good behavior expected for the system

- Question: does the system always behave well?

Context: Model Checking Timed Systems (2/2)

- Use formal methods

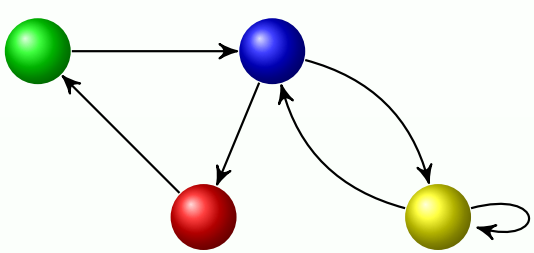


A finite model of the system

A formula to be satisfied

Context: Model Checking Timed Systems (2/2)

- Use formal methods



?

$\models$

$AG\neg$ 

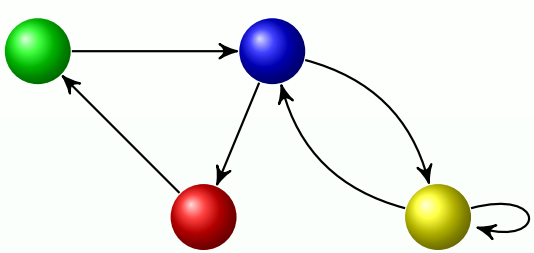
A finite model of the system

A formula to be satisfied

- Question: does the model of the system satisfy the formula?

Context: Model Checking Timed Systems (2/2)

- Use formal methods



$\stackrel{?}{\models}$

$AG\neg$ 

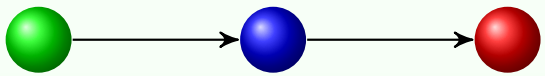
A finite model of the system

A formula to be satisfied

- Question: does the model of the system satisfy the formula?

Yes

No



Counterexample

Outline

Outline

Motivation

- Formal modeling language
- Quantitative verification of timed concurrent systems
- Hierarchical design
- Intuitive syntax

Features of Stateful Timed CSP

- Standard constructions of CSP [Hoare, 1978]
  - Events
  - Conditional and general choice
  - Sequential and parallel composition
  - Operations on shared variables
- Data structures
- Additional timed constructions [Schneider, 2000, Sun et al., 2009a]
  - `Wait[d]`: waits exactly `d` time units
  - `P timeout[d] Q`: the first observable event of `P` shall occur before `d` time units; otherwise, will behave like `Q`
  - `P interrupt[d] Q`: behaves like `P` until `d` time units; then, behaves like `Q`
  - `P within[d]`: the first observable event of `P` shall occur before `d` time units
  - `P deadline[d]`: `P` shall terminate before `d` time units

Semantics of Stateful Timed CSP

- Use implicit clocks
  - Clocks: real-valued variables increasing linearly at the same rate
  - Clocks are created and start when some processes are activated
  - Clocks are deleted when no longer used

- Configurations: Variables + Process + Value for each clock
  - Problem of real-time: infinite set of values
    $\leadsto$ Infinite representation of the state space

- Abstract configurations: Variables + Process + Clock zone [Sun et al., 2009a]
  - Clock zone: constraint on the clocks
    $\leadsto$ Finite representation of the state space

An Example (1/2)

- An Example of STCSP Process
  - (with no variable)

$$P \triangleq (a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]\ c \rightarrow P$$

- Intuitive behavior: b will never happen, because of $interrupt[3]$

An Example (2/2)

$$\mathbf{P} \triangleq (\mathbf{a} \rightarrow \mathbf{Wait}[\mathbf{5}]; \mathbf{b} \rightarrow \mathbf{Stop})\ \mathbf{interrupt}[\mathbf{3}]\ \mathbf{c} \rightarrow \mathbf{P}$$

- P

  true

An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]\ c \rightarrow P}$

- $(a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]\ \ c \rightarrow P$
  `true`

An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]\ c \rightarrow P}$

- $(a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$

An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]\ c \rightarrow P}$

- $(a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$
- $(Wait[5]\ \ ; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq 3$

An Example (2/2)

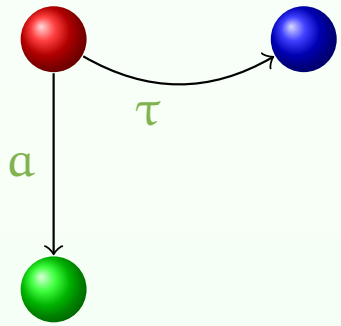
$\mathbf{P \triangleq (a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]\ c \rightarrow P}$

- $(a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$
- $(Wait[5]_{x_2}; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq 3 \wedge x_2 = 0$

An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]\ c \rightarrow P}$

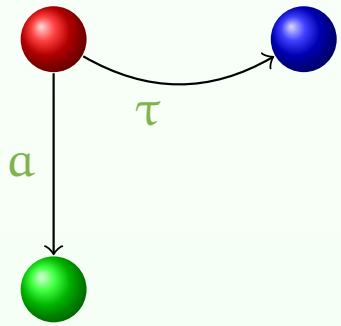
- $(a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$
- $(Wait[5]_{x_2}; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq 3 \wedge x_2 = 0$
- $c \rightarrow P$
  $x_1 \geq 0 \wedge x_1 = 3$



An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]\ c \rightarrow P}$

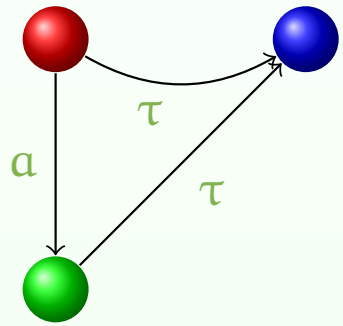
- $(a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$
- $(Wait[5]_{x_2}; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq 3 \wedge x_2 = 0$
- $c \rightarrow P$
  $\mathtt{true}$



An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]\ c \rightarrow P}$

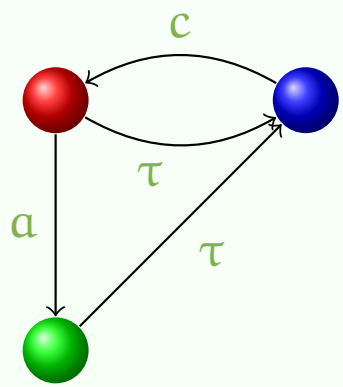
- $(a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$
- $(Wait[5]_{x_2}; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq 3 \wedge x_2 = 0$
- $c \rightarrow P$
  $\mathtt{true}$



An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]\ c \rightarrow P}$

- $(a \rightarrow Wait[5]; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$
- $(Wait[5]_{x_2}; b \rightarrow Stop)\ interrupt[3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq 3 \wedge x_2 = 0$
- $c \rightarrow P$
  $\mathtt{true}$



Related Work

- Timed Automata [Alur and Dill, 1994]
  - Expressiveness: incomparable with Stateful Timed CSP
  - Possible use of data structures (e.g., in UPPAAL)
  - Explicit set of clocks
  - Limited hierarchical definition
- Time Petri Nets [Merlin, 1974]
  - Expressiveness: more or less equivalent to Timed Automata
  - Possible use of data structures (e.g., in colored Time Petri Nets)
  - Possible use of hierarchical structures
  - Composition less straightforward
  - Some modular verification

Syntax of Parametric Stateful Timed CSP

- Goals
  - Avoid numerous verifications for many timing constants
  - Synthesis of parameters w.r.t. a given property
  - Optimize timing constants
  - Quantify the robustness of the system
- Extension of Stateful Timed CSP
  - Constants in timed constructs can be unkown, i.e., parameters

Configurations in Parametric Stateful Timed CSP

- Semantics of Stateful Timed CSP
  - Abstract configurations: Variables + Process + Clock zone
    $\rightsquigarrow$ Finite representation of the state space
- Semantics of Parametric Stateful Timed CSP
  - (Symbolic) configurations: Variables + Process + Constraint on the clocks and the parameters
  - Non-necessarily finite representation of the state space!

An Example (1/2)

- Remember the example of STCSP Process

$$P \triangleq (a \rightarrow Wait[5]; b \rightarrow Stop) \ interrupt[3] \ c \rightarrow P$$

- Now consider a parametric version of P
  - Use 2 parameters: u_3 and u_5

$$P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop) \ interrupt[u_3] \ c \rightarrow P$$

- Intuitive behavior: b may now happen
  - Depends on the relation between u_3 and u_5

An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

P

true

true

An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P$
  $\mathtt{true}$ $\mathtt{true}$

An Example (2/2)

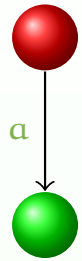
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ true

An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

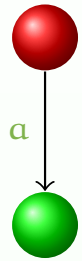
- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ — true
- $(Wait[u_5]\ ; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3$ — true



An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

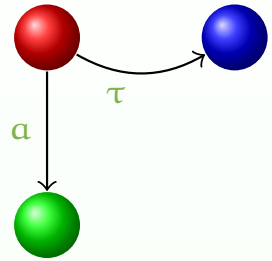
- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true



An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

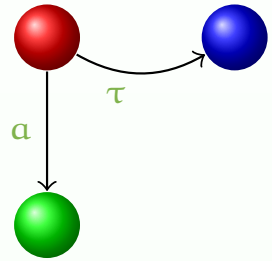
- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ true
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ true
- $c \rightarrow P$
  $x_1 \geq 0 \wedge x_1 = u_3$ true



An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

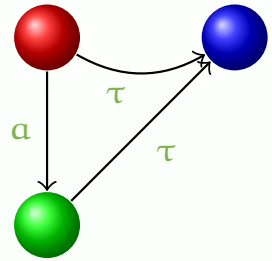
- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \rightarrow P$
  true — true



An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

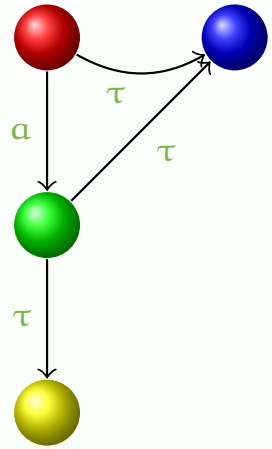
- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \rightarrow P$
  true — true



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

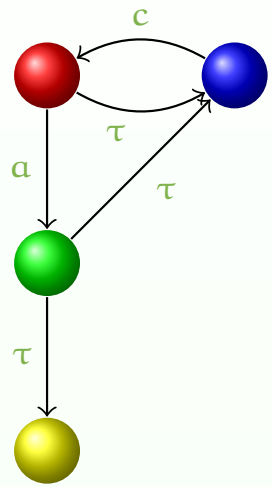
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

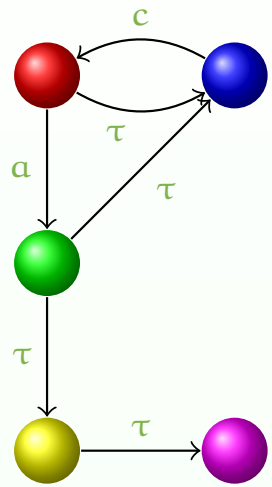
- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \rightarrow P$
  true — true
- $(Skip; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

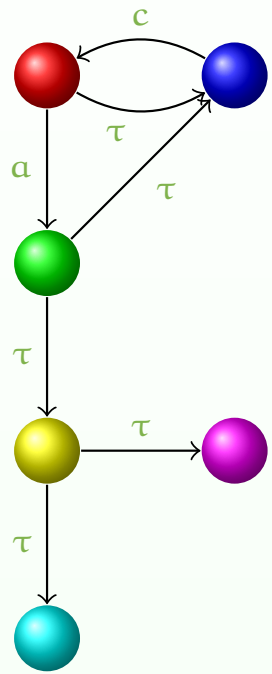
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

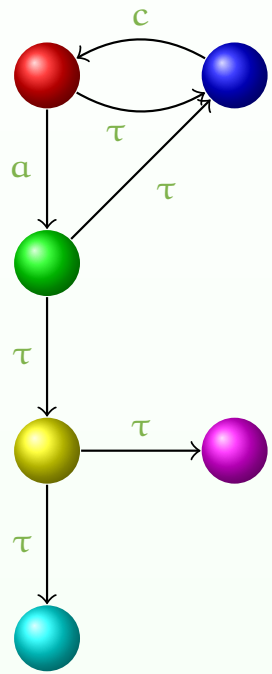
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq x_1 \wedge x_1 = u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

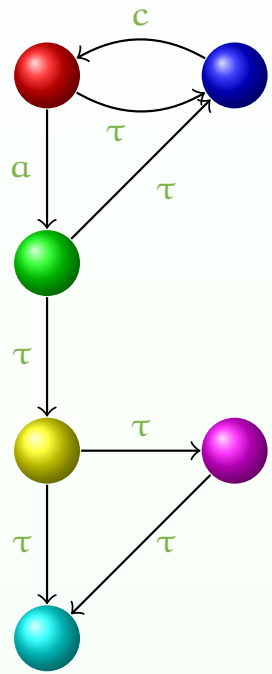
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

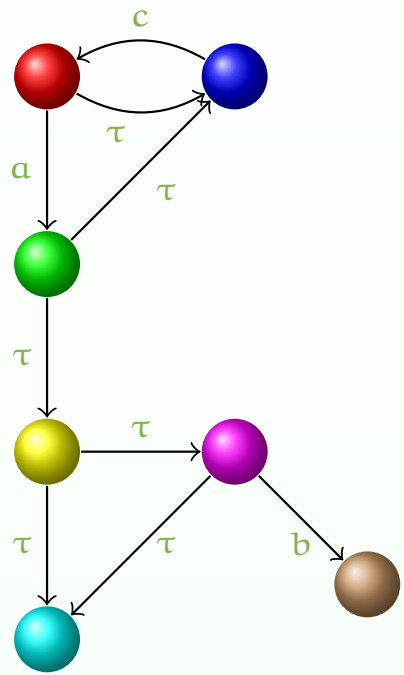
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

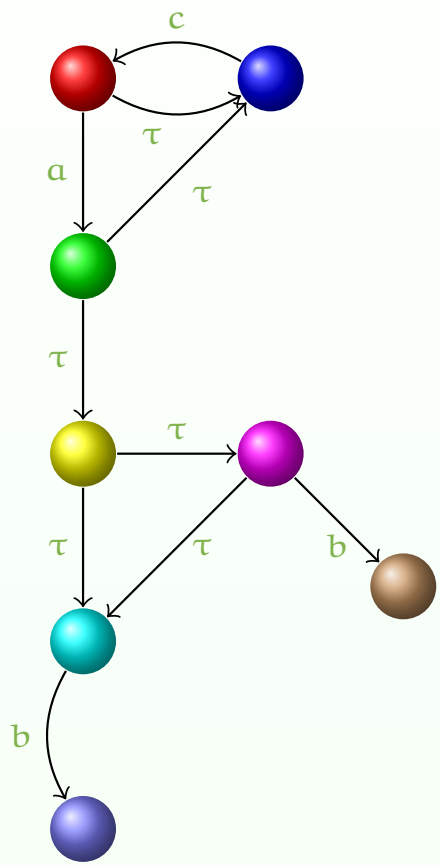
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$
- $Stop\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

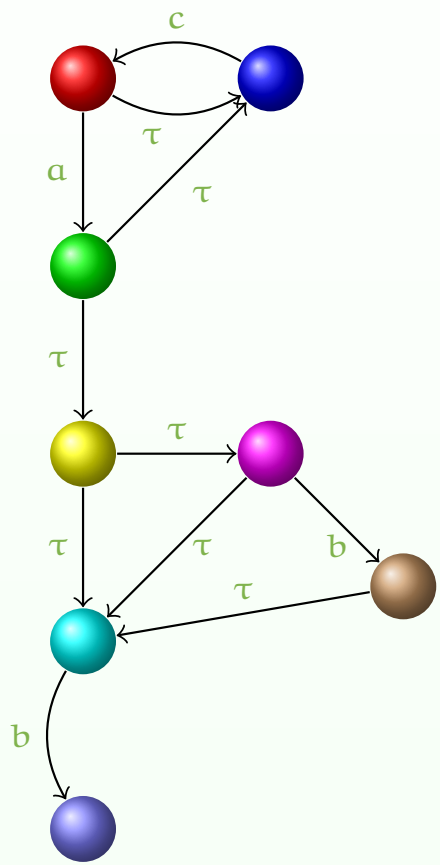
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$
- $Stop\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- P
  $u_5 \leq u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

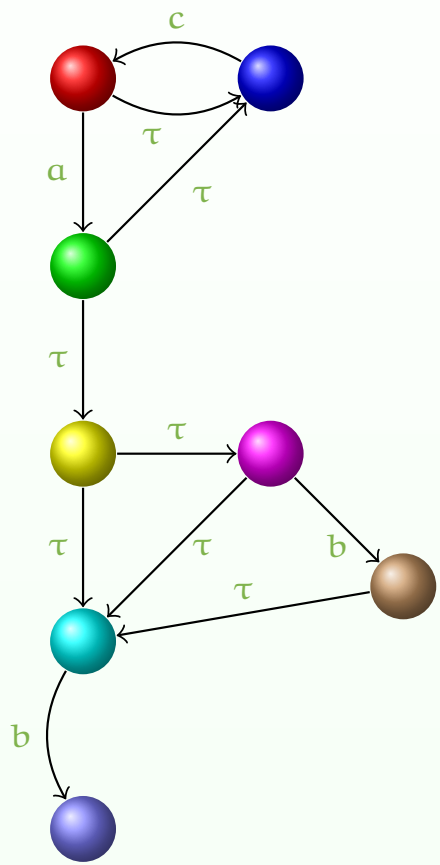
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$
- $Stop\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- P
  $u_5 \leq u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

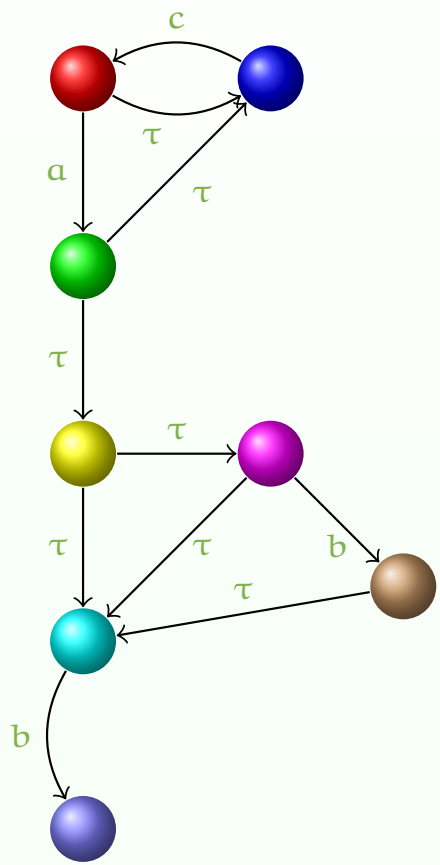
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$
- $Stop\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

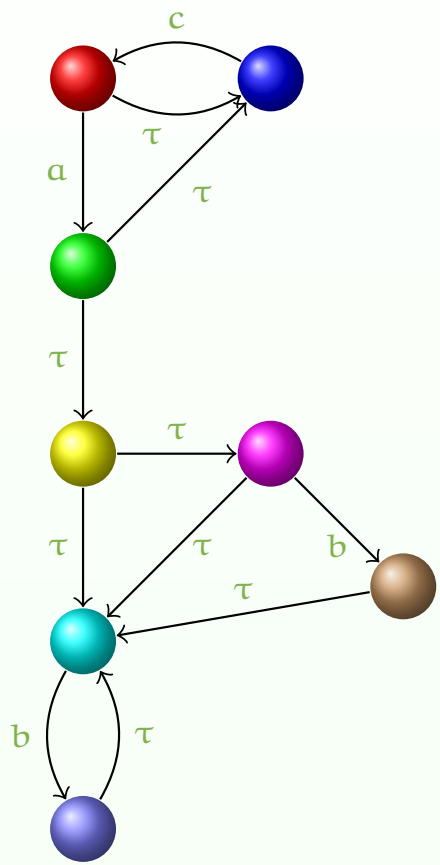
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$
- $Stop\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq u_3 \wedge x_1 = 0$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

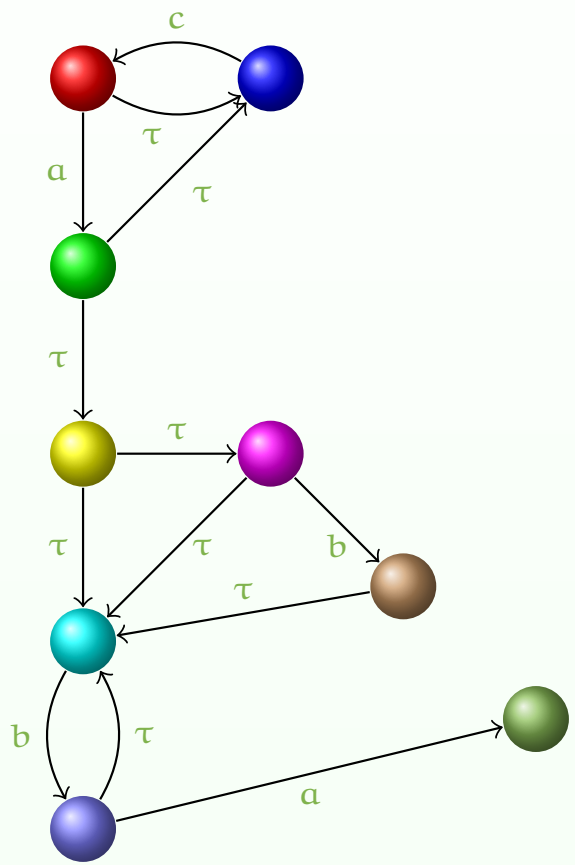
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$
- $Stop\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq u_3 \wedge x_1 = 0$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

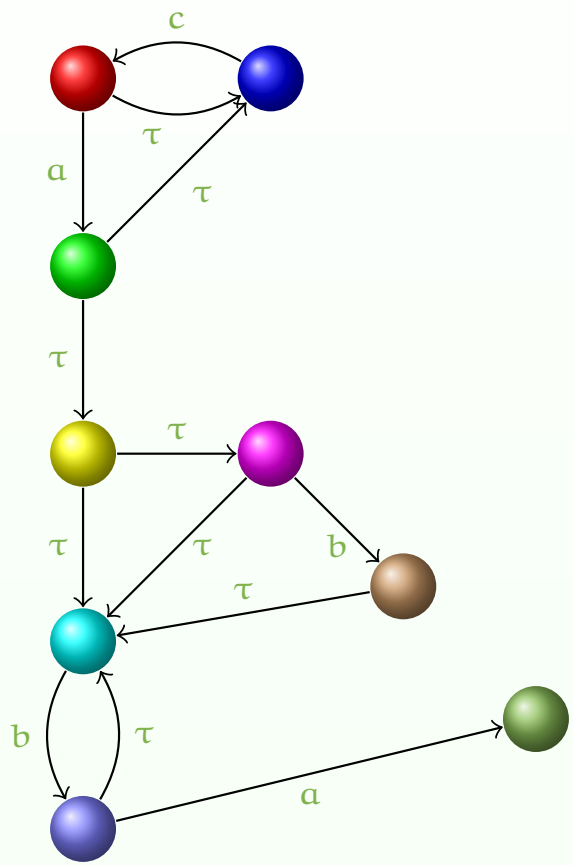
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$
- $Stop\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq u_3 \wedge x_1 = 0$ — $u_5 \leq u_3$
- $(Wait[u_5]\ ; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge u_5 \leq u_3$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

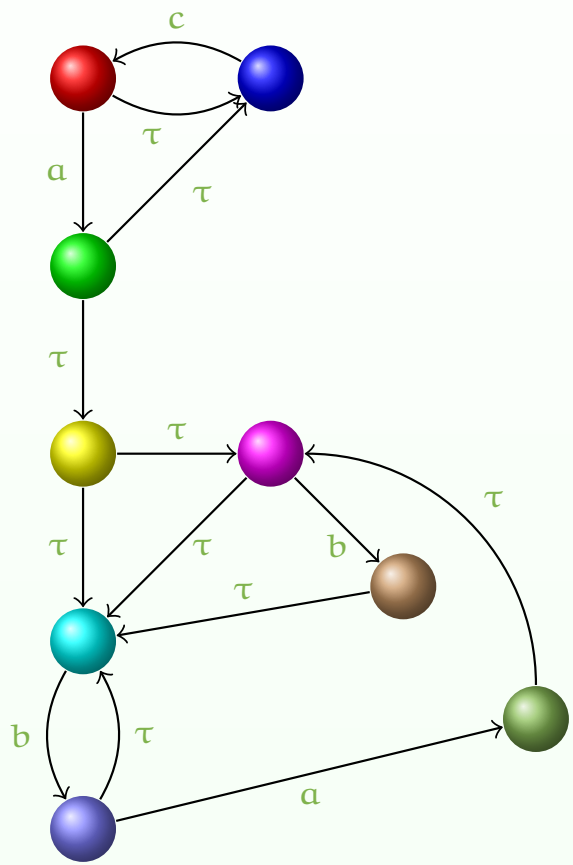
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$
- $Stop\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq u_3 \wedge x_1 = 0$ — $u_5 \leq u_3$
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge u_5 \leq u_3 \wedge x_2 = 0$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

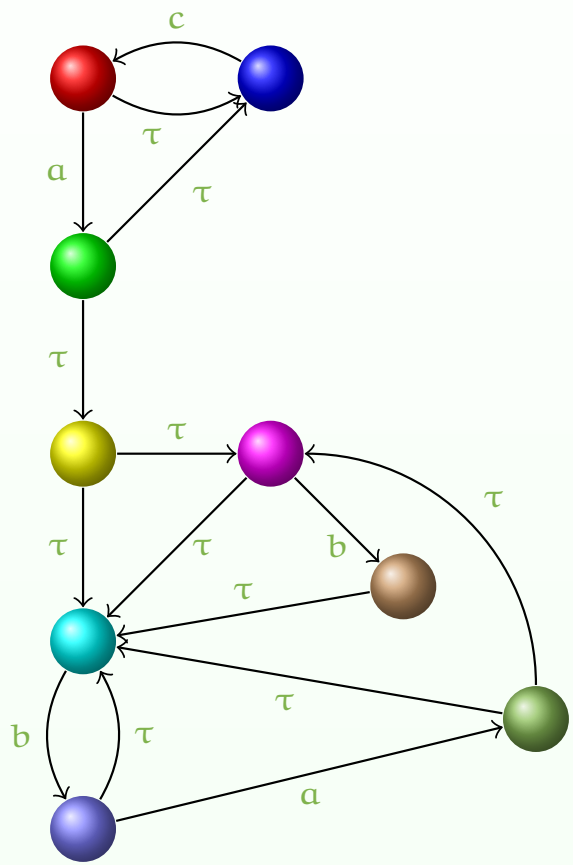
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $c \to P$
  $u_5 \leq u_3$ — $u_5 \leq u_3$
- $Stop\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ — $u_5 \leq u_3$
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq u_3 \wedge x_1 = 0$ — $u_5 \leq u_3$
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge u_5 \leq u_3 \wedge x_2 = 0$ — $u_5 \leq u_3$



An Example (2/2)

$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ — true
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \le x_1 \le u_3 \wedge x_2 = 0$ — true
- $c \to P$
  true — true
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \le x_1 \le u_3$ — $u_5 \le u_3$
- $(b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \le x_1 \le u_3$ — $u_5 \le u_3$
- $c \to P$
  $u_5 \le u_3$ — $u_5 \le u_3$
- $Stop\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \le x_1 \le u_3$ — $u_5 \le u_3$
- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \le u_3 \wedge x_1 = 0$ — $u_5 \le u_3$
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \le x_1 \le u_3 \wedge u_5 \le u_3 \wedge x_2 = 0$ — $u_5 \le u_3$



Outline

Expressiveness of PSTCSP

- Timed CSP equivalent to closed timed ϵ-automata [Ouaknine and Worrell, 2003]
  - Stricly less expressive than Timed ϵ-Automata
  - Incomparable with Timed Automata

- Corollary: Parametric Stateful Timed CSP is as expressive as Parametric Closed Timed ϵ-automata
  - Incomparable with standard PTAs

- Expressive enough
  - Most well-known protocols can be easily modeled using PSTCSP

Decidability of Membership for PSTCSP

- Membership question:
  Is a parameter valuation consistent with a process?

- Decidable (and easy)
  1. Instantiate the process with the parameter valuation
  2. Apply techniques for (non-parametric) Stateful Timed CSP [Sun et al., 2009a]

Definition of Emptiness in PSTCSP

- Emptiness question:
  "Does there exist a parameter valuation consistent with a process?"
- Unclear definition of consistency in ((P)ST)CSP
  - For Timed Automata: acceptance of at least a timed word
  - But no notion of acceptance in CSP
- Suggestion for the definition of consistency
  - "Does there exist a timed word such that a process derives to another given process?"
  - More specifically: halting problem
    "Does there exist a timed word such that a process derives to Stop?"

Undecidability of Emptiness for PSTCSP

Theorem

The problem of deciding whether a PSTCSP model is empty is undecidable.

Two possible proofs:

- Encoding of a 2-counter machine
- Equivalence to a subset of PTAs, for which the emptiness problem is also undecidable

Undecidability of Parameter Synthesis

Corollary

Parameter synthesis is undecidable in general for PSTCSP.

Alternative Definition of Emptiness

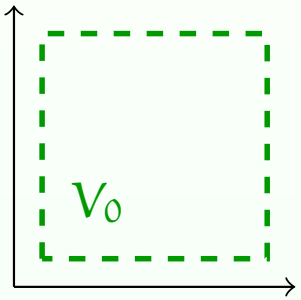
- Alternative definition of acceptance
  - "A configuration is accepting if and only if the process doesn't contain `within` or `deadline`"
  - Then, the model is non-empty if there is at least an accepting timed word
- More interesting
  - But problem still open
  - Work in progress

Outline

Algorithms for Parameter Synthesis

- Parameter synthesis: guarantees that the system will behave well w.r.t. a given property

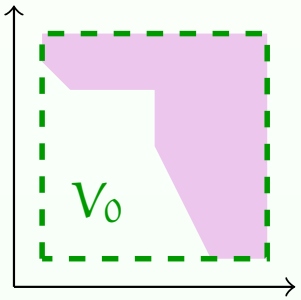
- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"



Algorithms for Parameter Synthesis

- Parameter synthesis: guarantees that the system will behave well w.r.t. a given property

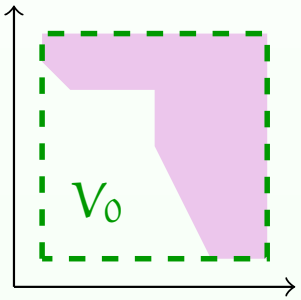
- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"



Algorithms for Parameter Synthesis

- Parameter synthesis: guarantees that the system will behave well w.r.t. a given property

- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"



- Design of semi-algorithms
  - Termination not guaranteed because undecidable problem
  - Nevertheless, some methods terminate "more often" than others

Computation of the Reachability Graph

- Idea: compute all possible configurations
  - Algorithm *reachAll*
- No guarantee to terminate!

Synthesis w.r.t. a Predicate

- Input: a process, and a predicate φ on variables
- Problem
  - Find the set of *all* parameter valuations such that we *can* reach *at least one* configuration satisfying φ
- Idea
  - On-the-fly computation of the reachable configurations (e.g., using depth first search algorithm)
  - When a configuration satisfies φ, store the corresponding constraint on the parameters, and locally stop the exploration
  - Return the union of all constraints

Synthesis w.r.t. a Predicate: Principle

V_0, P, C_0

Input:
P, φ

Output:
$K = \texttt{false}$

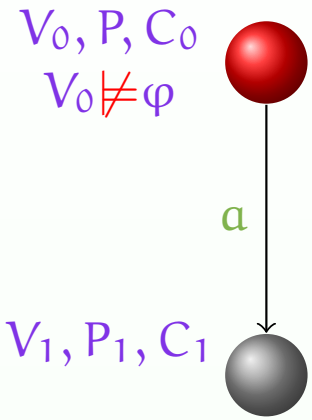
Synthesis w.r.t. a Predicate: Principle

V_0, P, C_0
$V_0 \not\models \varphi$

Input:
P, φ

Output:
$K = \texttt{false}$

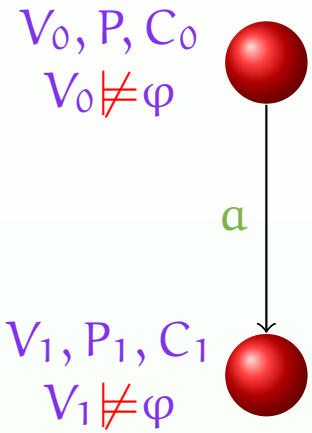
Synthesis w.r.t. a Predicate: Principle



Input:
P, φ

Output:
$K = \texttt{false}$

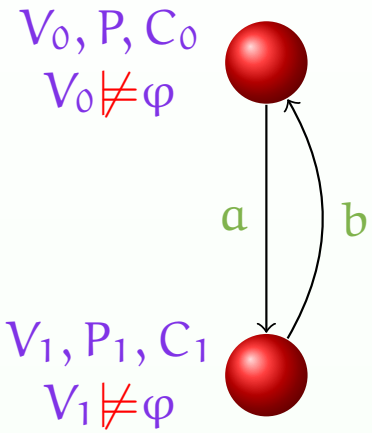
Synthesis w.r.t. a Predicate: Principle



Input:
P, φ

Output:
$K = \texttt{false}$

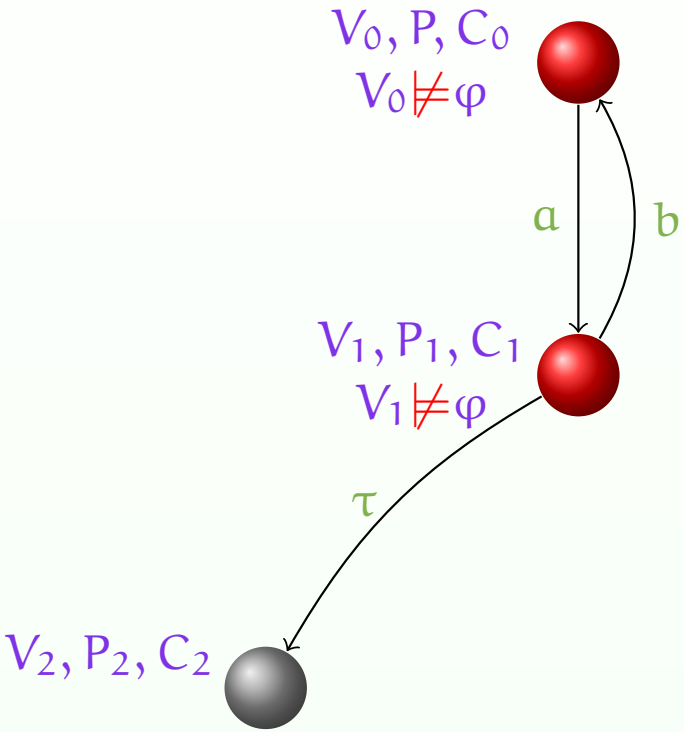
Synthesis w.r.t. a Predicate: Principle



Input:
P, φ

Output:
$K = \texttt{false}$

Synthesis w.r.t. a Predicate: Principle



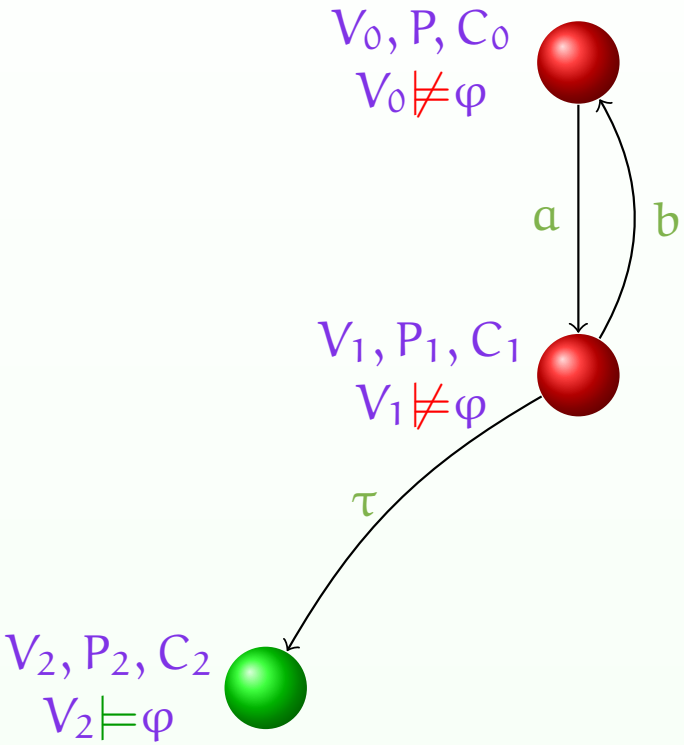
Input:

P, φ

Output:

$K = \texttt{false}$

Synthesis w.r.t. a Predicate: Principle



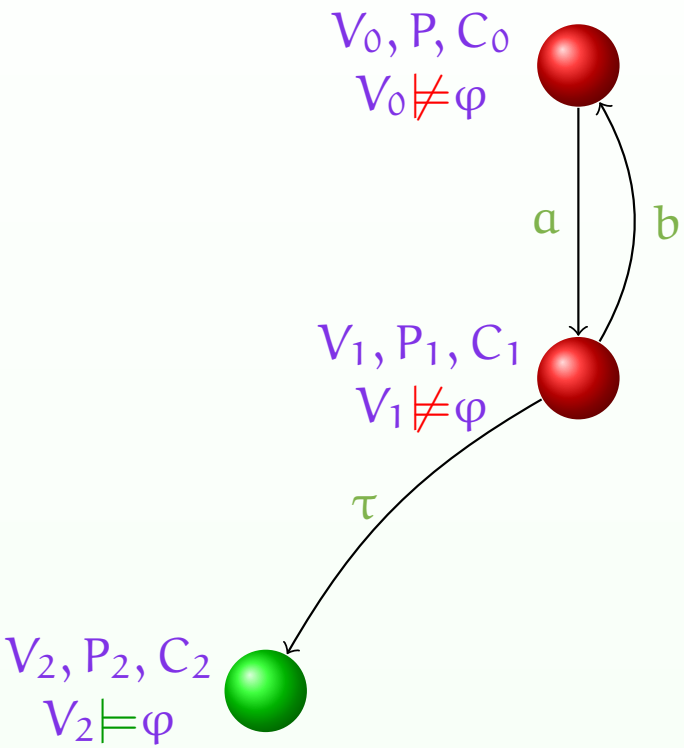
Input:

P, φ

Output:

$K = \texttt{false}$

Synthesis w.r.t. a Predicate: Principle



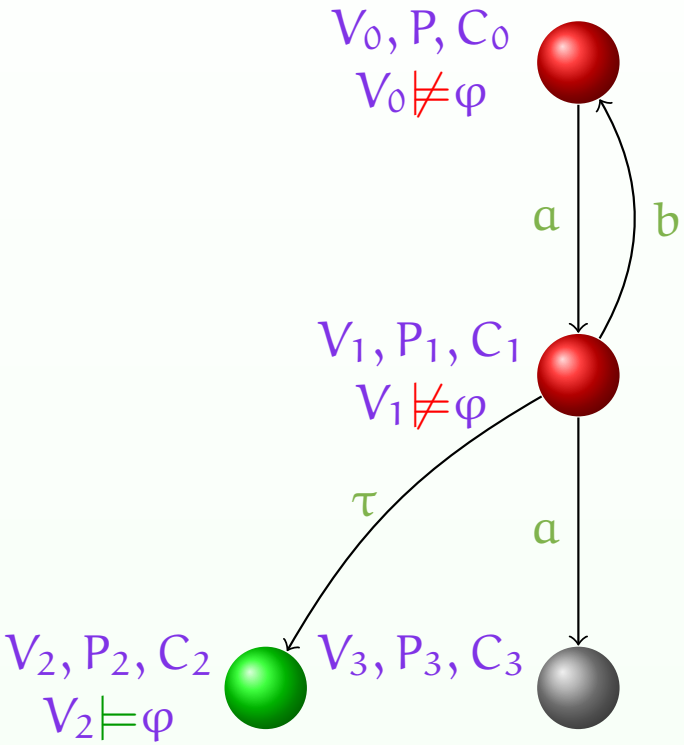
Input:

P, φ

Output:

$K = (\exists X : C_2)$

Synthesis w.r.t. a Predicate: Principle



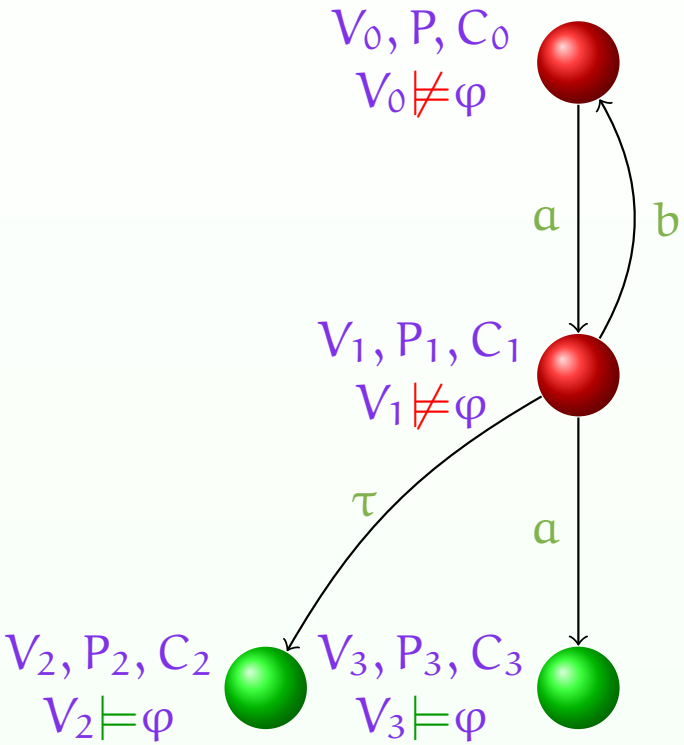
Input:

P, φ

Output:

$K = (\exists X : C_2)$

Synthesis w.r.t. a Predicate: Principle



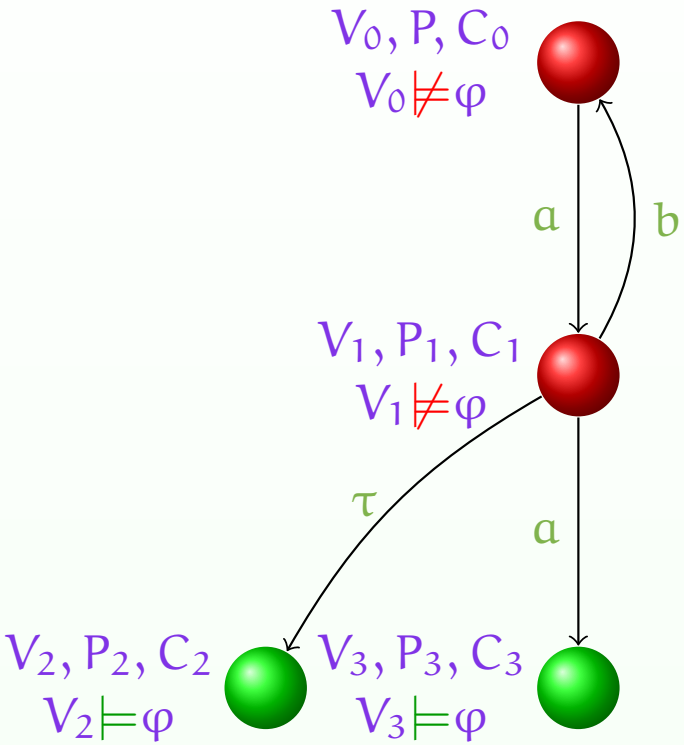
Input:

P, φ

Output:

$K = (\exists X : C_2)$

Synthesis w.r.t. a Predicate: Principle



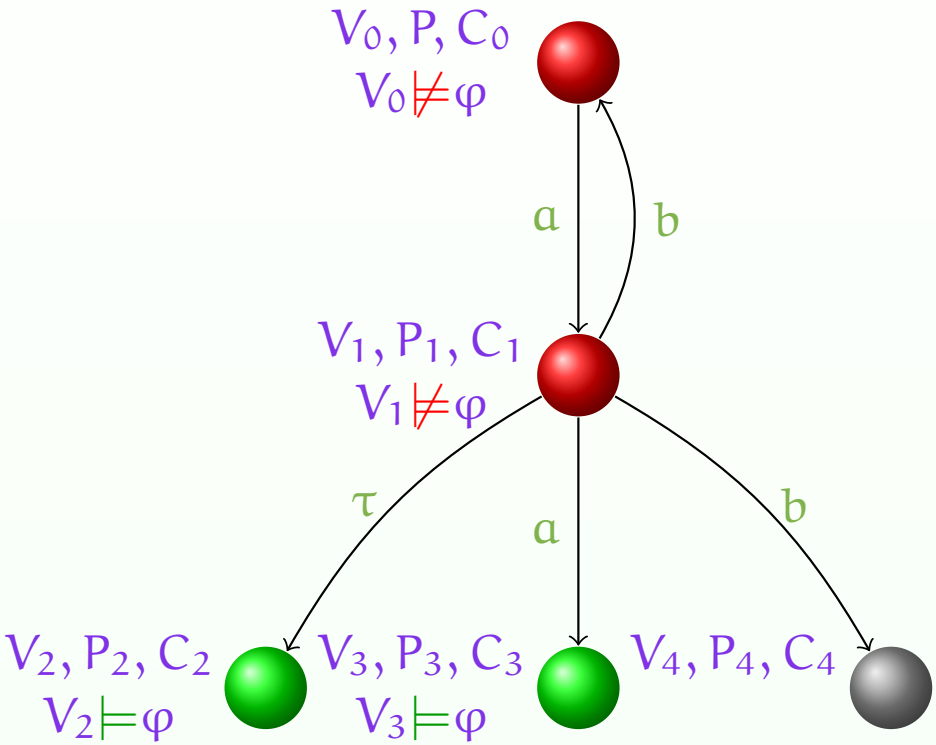
Input:

P, φ

Output:

$K = (\exists X : C_2) \vee (\exists X : C_3)$

Synthesis w.r.t. a Predicate: Principle



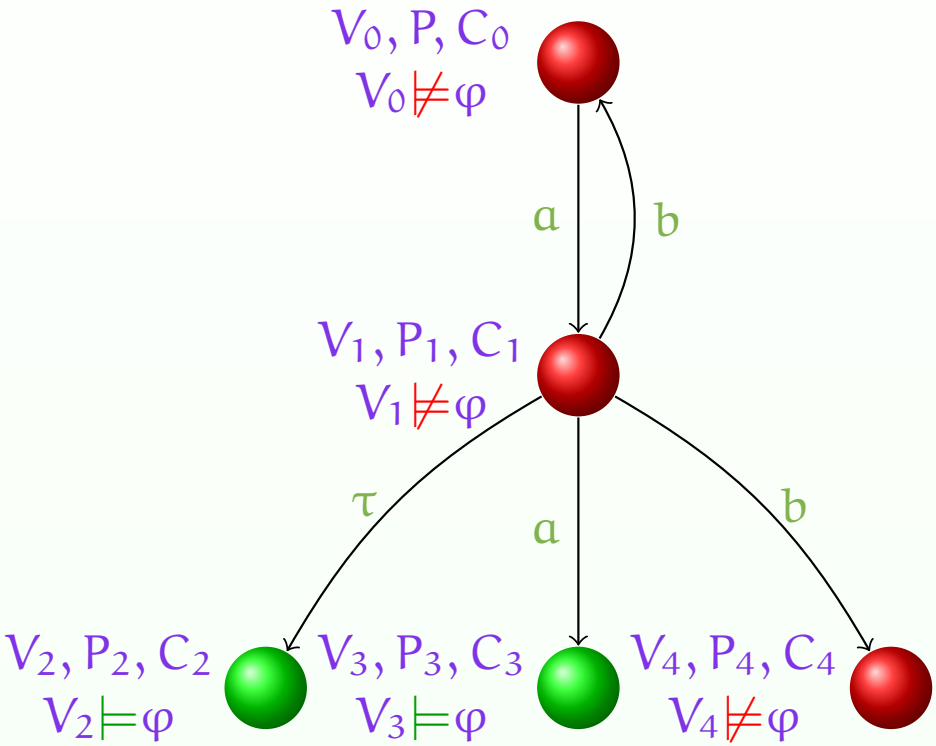
Input:

P, φ

Output:

$K = (\exists X : C_2) \vee (\exists X : C_3)$

Synthesis w.r.t. a Predicate: Principle



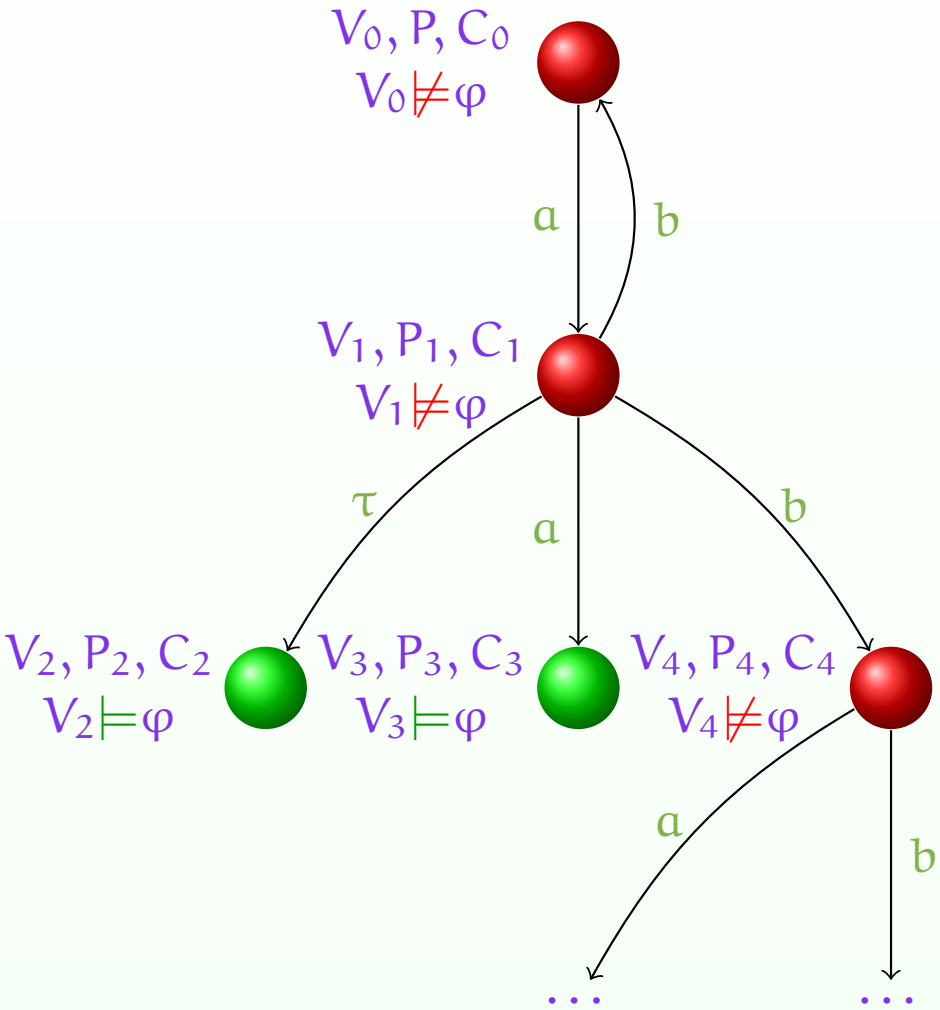
Input:

P, φ

Output:

$K = (\exists X : C_2)$
$\vee\ (\exists X : C_3)$

Synthesis w.r.t. a Predicate: Principle



Input:

P, φ

Output:

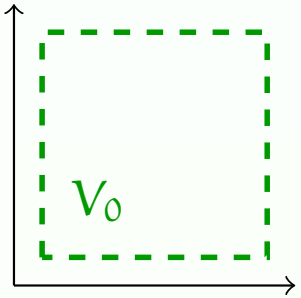
$$K = (\exists X : C_2) \vee (\exists X : C_3) \vee \ldots$$

Synthesis w.r.t. a Predicate: Properties

- Termination not guaranteed
  - But reachability graph not explored in full
- Main interest
  - Used for bad properties, i.e., for bug detection
  - In this case, the "good" constraint is the negation of the constraint computed w.r.t. φ

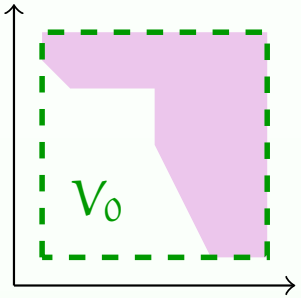
Inverse Method for PSTCSP

- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"



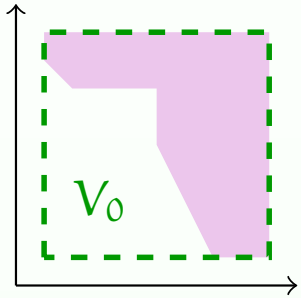
Inverse Method for PSTCSP

- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"

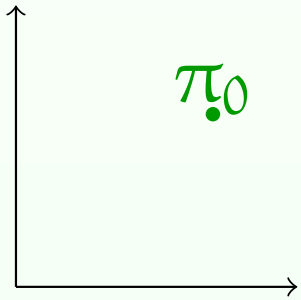


Inverse Method for PSTCSP

- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"

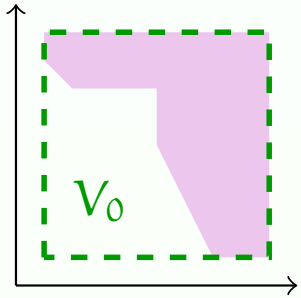


- The inverse problem
  - "Given a reference parameter valuation π_0, find other valuations around π_0 of same behavior"

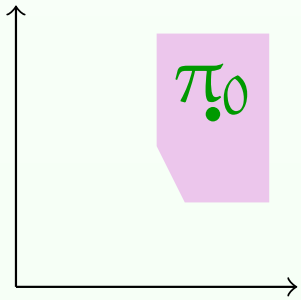


Inverse Method for PSTCSP

- The good parameters problem
  - "Given a bounded parameter domain V_0, find a set of parameter valuations of good behavior in V_0"



- The inverse problem
  - "Given a reference parameter valuation π_0, find other valuations around π_0 of same behavior"



Inverse Method for PSTCSP: Main Idea

- Idea of *IM*
  - Initially defined for Parametric Timed Automata [André et al., 2009]
  - On-the-fly computation of the reachable configurations (e.g., using depth first search algorithm)
  - When the constraint associated to a configuration does not satisfy π_0, negate an inequality and add it to all configurations, so that this configuration is removed
  - Return the intersection of the constraints associated to all configurations

Inverse Method for PSTCSP: Example

Input

$\mathbf{P} \triangleq (\mathbf{a} \rightarrow \mathbf{Wait}[\mathbf{u_5}]; \mathbf{b} \rightarrow \mathbf{Stop})\ \mathbf{interrupt}[\mathbf{u_3}]\ \mathbf{c} \rightarrow \mathbf{P}$

$\pi_0 : \{\mathbf{u_3} = \mathbf{3} \wedge \mathbf{u_5} = \mathbf{5}\}$

Output

$K = \texttt{true}$

Inverse Method for PSTCSP: Example

Input

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

Inverse Method for PSTCSP: Example

Input

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

Inverse Method for PSTCSP: Example

Input

$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\mathbf{\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}}$

Output

$K = \texttt{true}$

$(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$

$x_1 = 0$

$\pi_0 \models \texttt{true}$

Inverse Method for PSTCSP: Example

Input

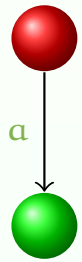
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]\ ; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3$ $\quad \texttt{true}$



Inverse Method for PSTCSP: Example

Input

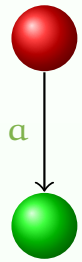
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \texttt{true}$



Inverse Method for PSTCSP: Example

Input

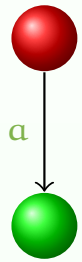
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \pi_0 \models \texttt{true}$



Inverse Method for PSTCSP: Example

Input

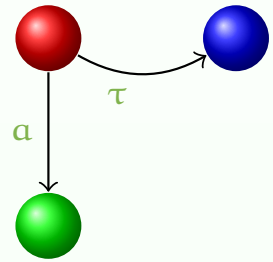
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \pi_0 \models \texttt{true}$
- $c \rightarrow P$
  $x_1 \geq 0 \wedge x_1 = u_3$ $\quad \texttt{true}$



Inverse Method for PSTCSP: Example

Input

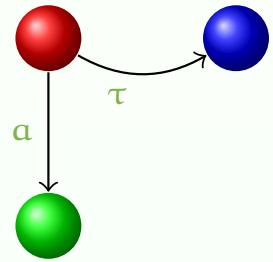
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \pi_0 \models \texttt{true}$
- $c \rightarrow P$
  $\texttt{true}$ $\quad \texttt{true}$



Inverse Method for PSTCSP: Example

Input

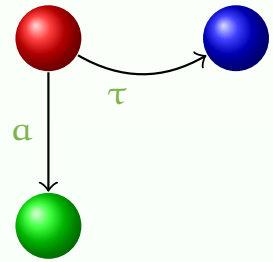
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \pi_0 \models \texttt{true}$
- $c \rightarrow P$
  $\texttt{true}$ $\quad \pi_0 \models \texttt{true}$



Inverse Method for PSTCSP: Example

Input

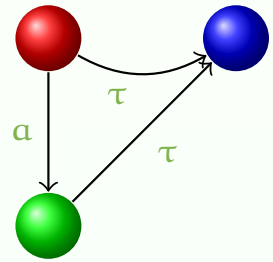
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \pi_0 \models \texttt{true}$
- $c \rightarrow P$
  $\texttt{true}$ $\quad \pi_0 \models \texttt{true}$



Inverse Method for PSTCSP: Example

Input

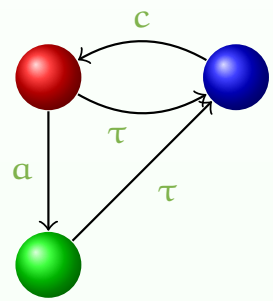
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \pi_0 \models \texttt{true}$
- $c \rightarrow P$
  $\texttt{true}$ $\quad \pi_0 \models \texttt{true}$



Inverse Method for PSTCSP: Example

Input

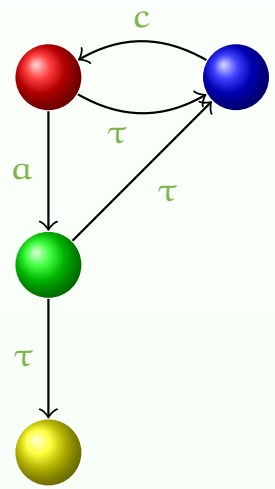
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \pi_0 \models \texttt{true}$
- $c \rightarrow P$
  $\texttt{true}$ $\quad \pi_0 \models \texttt{true}$
- $(Skip; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $u_5 \leq x_1 \leq u_3$ $\quad u_5 \leq u_3$



Inverse Method for PSTCSP: Example

Input

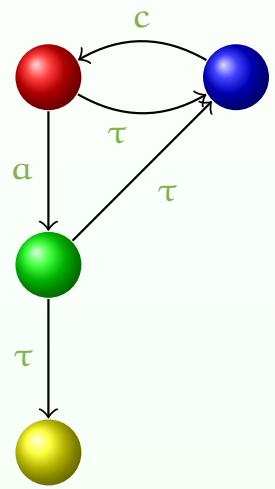
$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = \texttt{true}$

- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \pi_0 \models \texttt{true}$
- $c \to P$
  $\texttt{true}$ $\quad \pi_0 \models \texttt{true}$
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ $\quad \pi_0 \not\models u_5 \leq u_3$



Inverse Method for PSTCSP: Example

Input

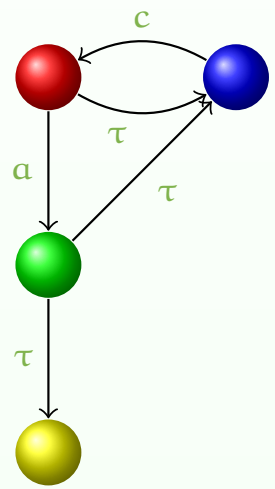
$\mathbf{P \triangleq (a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]\ c \to P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = u_5 > u_3$

- $(a \to Wait[u_5]; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $x_1 = 0$ $\quad \pi_0 \models \texttt{true}$
- $(Wait[u_5]_{x_2}; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \pi_0 \models \texttt{true}$
- $c \to P$
  $\texttt{true}$ $\quad \pi_0 \models \texttt{true}$
- $(Skip; b \to Stop)\ interrupt[u_3]_{x_1}\ c \to P$
  $u_5 \leq x_1 \leq u_3$ $\quad \pi_0 \not\models u_5 \leq u_3$



Inverse Method for PSTCSP: Example

Input

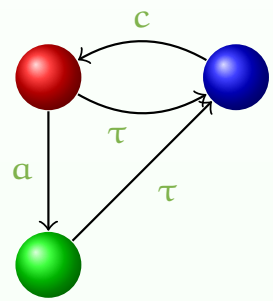
$\mathbf{P \triangleq (a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]\ c \rightarrow P}$

$\pi_0 : \{u_3 = 3 \wedge u_5 = 5\}$

Output

$K = u_5 > u_3$

- $(a \rightarrow Wait[u_5]; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $x_1 = 0$ $\quad \pi_0 \models \mathtt{true}$
- $(Wait[u_5]_{x_2}; b \rightarrow Stop)\ interrupt[u_3]_{x_1}\ c \rightarrow P$
  $0 \leq x_1 \leq u_3 \wedge x_2 = 0$ $\quad \pi_0 \models \mathtt{true}$
- $c \rightarrow P$
  $\mathtt{true}$ $\quad \pi_0 \models \mathtt{true}$



Inverse Method for PSTCSP: Properties

- Preserves properties on traces (LTL formulae)
  - From the equality of trace sets
- Termination
  - Better termination than other methods
  - Not guaranteed for PTAs (proved)
  - Not guaranteed for PSTCSP
    (...but no counter-example found)
- Completeness
  - Given π, if a process under π behaves like under π_0, then π belongs to the constraint synthesized by the method
  - Not guaranteed for PTAs (proved)
  - Not guaranteed for PSTCSP
    (...but no counter-example found)

Synthesis w.r.t. Refinement

- Input: a process P, and a specification P'
- Problem
  - Find *a* set of parameter valuations such that P refines P'
- Idea
  - On-the-fly computation of the reachable configurations (e.g., using depth first search algorithm)
  - Group τ transitions
  - If a transition a in P does not exist in P', negate an inequality of the corresponding constraint on the parameters, store this negated inequality, so that this configuration is removed
  - Return the intersection of the inequalities

Outline

PAT

- Process Analysis Toolkit [Sun et al., 2009b]
  - Self-contained framework implemented in C#
  - Composing, simulating and automatic verification of concurrent, real-time systems (and other domains)
  - User friendly interfaces, featured model editor and animated simulator
  - Model checking: deadlock-freeness, divergence-freeness, reachability, LTL properties with fairness assumptions, refinement checking, etc.

- Try it!

http://www.comp.nus.edu.sg/~pat/

Implementation of the Algorithms

- Computation of the reachability graph
  - ☺ Interesting for small examples
    - ⇝ Manual synthesis of parameters from the graph
  - ☹ Often leads to an infinite state space
    - ⇝ Does not terminate (no synthesis possible)

- Synthesis using the inverse method
  - ☺ Partial exploration of the state space only
  - ☺ Always terminate in practice

Encoding of a configuration

- Encoding
  - Process (ID)
  - Value for variables
  - List of clocks
  - Constraint: definition of a normal form
- Example
  - $(Wait[u_3]_{x_3} || Wait[u_5]_{x_3} || Wait[u_5]_{x_2}, x_3 \leq x_2)$
  - Encoding:
    - Process: $Wait[u_3] || Wait[u_5] || Wait[u_5]$
    - List of clocks: $\{x_3, x_3, x_2\}$
    - Constraint: $x_3 \leq x_2$
- Justification for the list of clocks
  - Distinguishes between
    $(Wait[u_3]_{x_3} || Wait[u_5]_{x_3} || Wait[u_5]_{x_2}, x_3 \leq x_2)$ and
    $(Wait[u_3]_{x_2} || Wait[u_5]_{x_3} || Wait[u_5]_{x_2}, x_3 \leq x_2)$

Encoding of a configuration: optimization

- Actual equivalence between
  $(Wait[u_3]_{x_1} \| Wait[u_5]_{x_2}, x_1 \leq x_2)$ and
  $(Wait[u_3]_{x_2} \| Wait[u_5]_{x_1}, x_2 \leq x_1)$

Encoding of a configuration: optimization

- Actual equivalence between
  $(Wait[u_3]_{x_1} || Wait[u_5]_{x_2}, x_1 \leq x_2)$ and
  $(Wait[u_3]_{x_2} || Wait[u_5]_{x_1}, x_2 \leq x_1)$
- Idea $\rightsquigarrow$ rename clocks
  - Example: $(Wait[u_3]_{x_3} || Wait[u_5]_{x_3} || Wait[u_5]_{x_2}, x_3 \leq x_2)$
    New encoding:
    - Process: $Wait[u_3] || Wait[u_5] || Wait[u_5]$
    - List of clocks: $\{x_1, x_1, x_2\}$
    - Constraint: $x_1 \leq x_2$
- This methods is time consuming
  - Numerous string and list sorting
  - But often lead to efficient state space reduction

Case studies

Case study	$\|U\|$	*reachAll* $\|S\|$	*reachAll* $\|T\|$	*reachAll* $\|X\|$	*reachAll* t	*reachAll+* $\|S\|$	*reachAll+* $\|T\|$	*reachAll+* $\|X\|$	*reachAll+* t	*IM* $\|S\|$	*IM* $\|X\|$	*IM* t	*IM+* $\|S\|$	*IM+* $\|X\|$	*IM+* t
Bridge	4	-	-	-	M.O.	-	-	-	M.O.	2.8k	2	253	2.8k	2	455
$Fischer_4$	2	-	-	-	M.O.	-	-	-	M.O.	11k	4	41.9	2k	4	8.65
$Fischer_5$	2	-	-	-	M.O.	-	-	-	M.O.	133k	5	1176	13k	5	84.5
$Fischer_6$	2	-	-	-	M.O.	-	-	-	M.O.	-	-	M.O.	86k	6	1144
Jobshop	8	14k	20k	2	21.0	12k	17k	2	18.1	1112	2	17.1	877	2	22.8
RCS_5	4	5.6k	7.2k	4	10.5	5.6k	7.2k	4	9.54	5.6k	4	7.83	5.6k	4	16.7
RCS_6	4	34k	43k	4	91.7	34k	43k	4	54.5	34k	4	60.4	34k	4	91.3
TrAHV	6	7.2k	13k	6	14.2	7.2k	13k	6	15.8	227	6	0.555	227	6	0.655

- *reachAll*: computation of the reachability graph
- *IM*: inverse method
- *reachAll+* (resp. *IM+*): version with optimized encoding

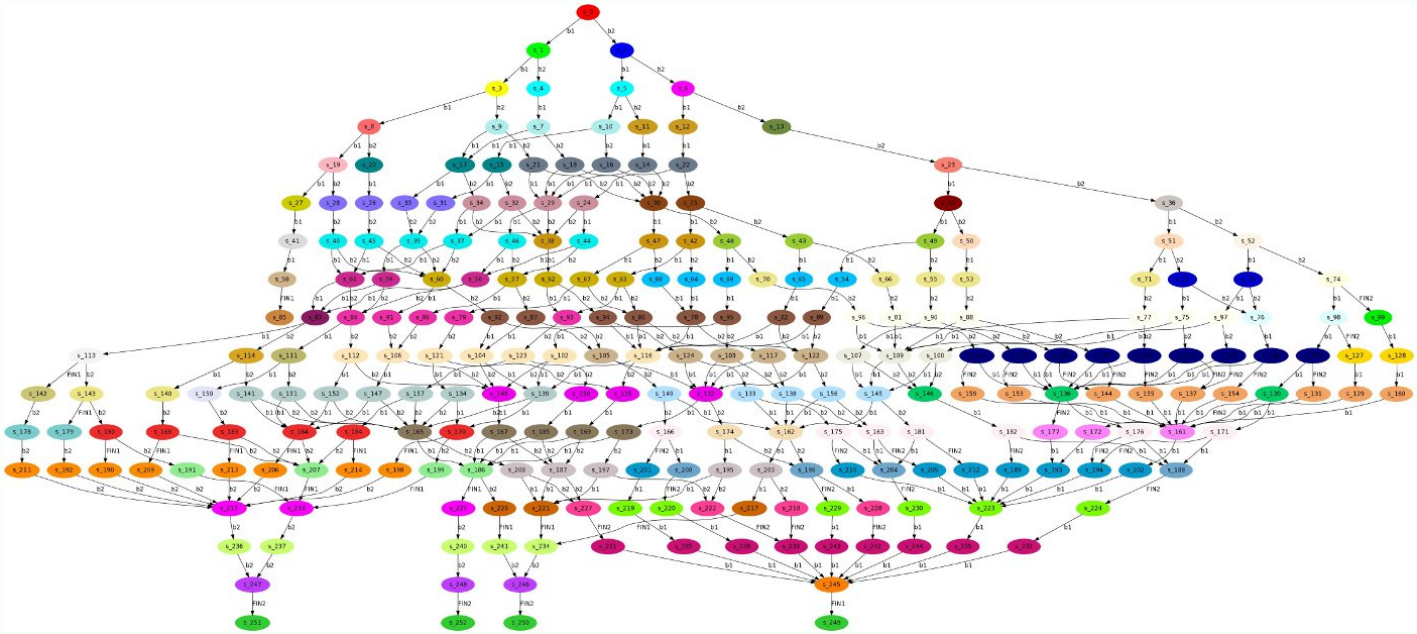
Outline

State Space Explosion

- Biggest issue in verification of real-time systems:
  state space explosion

State Space Explosion

- Biggest issue in verification of real-time systems:
  state space explosion



Non-Parametric Merging Technique

- Merging in Timed Automata (TAs) [David, 2005]

Definition (Mergeable states)

Two states (q, C) and (q', C') are mergeable if $q = q'$ and $C \cup C'$ is convex. (With q, q' the discrete part, and C, C' the associated constraint.)

Non-Parametric Merging Technique

- Merging in Timed Automata (TAs) [David, 2005]

Definition (Mergeable states)

Two states (q, C) and (q', C') are mergeable if $q = q'$ and $C \cup C'$ is convex. (With q, q' the discrete part, and C, C' the associated constraint.)

Most optimization techniques designed for TAs do not apply to Parametric Timed Automata (PTAs)

This one does! [André et al., 2012]

Merging for PTAs

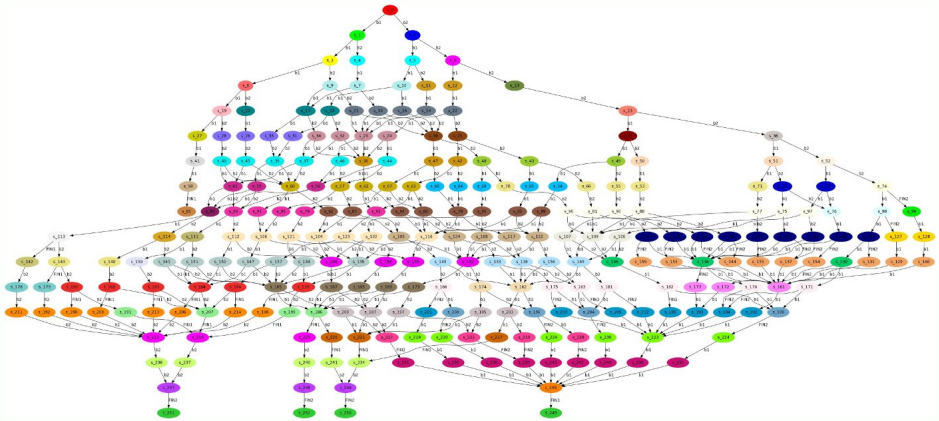
- Merging implemented in TAs using simple operations on DBMs
  - ☹ DBMs cannot be used for PTAs
- Checking union convexity for constraints is costly
- Tentative algorithm
  1. Compute the convex hull H of C and C'
  2. Compute $H \setminus C$
  3. If $(H \setminus C) \subseteq C'$ then return `true`
  4. Otherwise return `false`
- Implemented in IMITATOR [André, 2010]

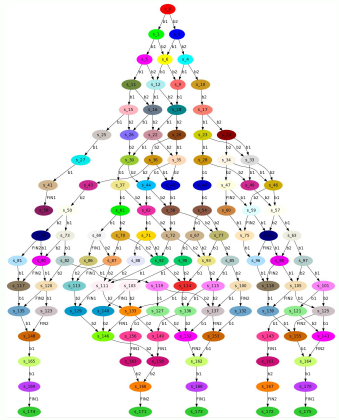
Merging for PTAs: the Inverse Method Case

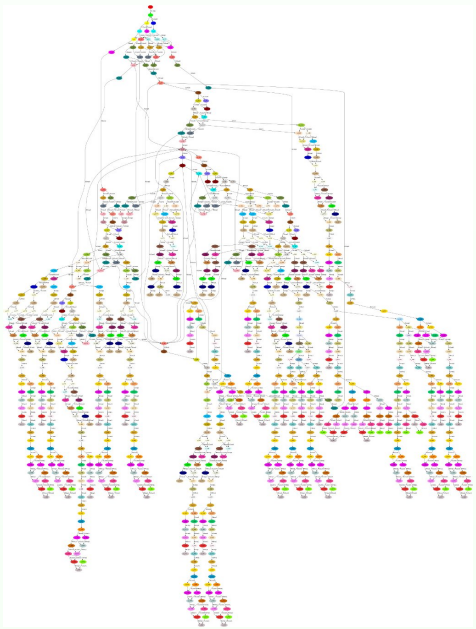
- Applying this merging technique in *IM* for PTAs
  - ☺ Reduces the state space
  - ☺ Reduces the computation time
  - ☺ Improves the resulting constraint

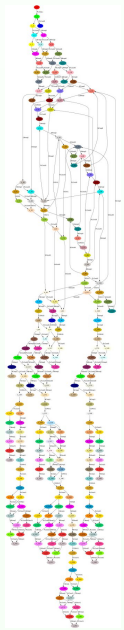
Example	X	P	*IM*				*IM* $_{merge}$				Comp.
			t	States	Trans.	M	t	States	Trans.	M	
AndOr	4	12	0.112	16	17	1,262	0.101	13	14	1,187	$=$
Flip-Flop	5	12	0.183	14	13	1,692	0.227	14	13	1,762	$=$
Latch	8	13	1.18	18	68	3,686	0.621	12	40	2,662	$\subsetneq$
BRP	7	6	4.29	428	474	25,483	7.015	426	473	25,845	$=$
WLAN	2	8	220.157	7,038	11,052	733,044	286.141	6,020	9,538	1,408,702	$=$
$SPSMALL_1$	10	26	1.578	31	35	5,098	1.642	31	35	5,442	$=$
$SPSMALL_2$	28	62	-	-	-	overflow	593	397	499	180,888	-
SIMOP	8	7	18.959	1,108	1,404	43,333	5.179	239	347	14,371	$\subsetneq$
CSMA/CD	3	3	0.801	240	383	6,580	0.947	240	383	7,049	$=$
Jobshop	3	8	1.865	253	387	10,658	1.147	118	179	5,221	$\subsetneq$
Mutex 3	3	2	0.802	307	1,060	14,598	0.671	241	811	11,934	$=$
Mutex 4	4	2	22.373	4,769	19,873	373,900	22.03	3,287	13,459	260,962	$=$

Some Nice Graphics









Outline

Conclusion

- Parametric Stateful Timed CSP
  - Extension of the process algebra (Timed) CSP
  - Powerful and intuitive language for specification of hierarchical concurrent systems
- Algorithms for parameter synthesis
  - Full reachability method: does not often terminate in practice
  - Inverse method: allows efficient parameter synthesis
    - Gives a criterion of robustness to the system
- Implementation of two optimizations
  - Normal form for states with clock renaming
  - State merging technique (for PTAs)

Future Work

- Theoretical questions
  - Equivalence of timed constructs
  - Termination and completeness of the inverse method for PSTCSP
- Application of the merging technique to PSTCSP
  - The theory applies in a straightforward manner
  - Add implementation in PAT
- More algorithms
  - Synthesis w.r.t. a predicate
  - Synthesis w.r.t. refinement (parametric refinement checking)
  - Synthesis w.r.t. LTL, CTL, TCTL properties, etc.

References I

Alur, R. and Dill, D. (1994).
A theory of timed automata.
TCS, 126(2):183–235.

André, É. (2010).
IMITATOR II: A tool for solving the good parameters problem in timed automata.
In *INFINITY'10*, volume 39 of *EPTCS*, pages 91–99.

André, É., Chatain, T., Encrenaz, E., and Fribourg, L. (2009).
An inverse method for parametric timed automata.
International Journal of Foundations of Computer Science, 20(5):819–836.

André, É., Fribourg, L., and Soulat, R. (2012).
Enhancing the inverse method with state merging.
In Goodloe, A. and Person, S., editors, *NFM'12*, volume 7226 of *LNCS*, pages 100–105. Springer.
To appear.

David, A. (2005).
Merging DBMs efficiently.
In *17th Nordic Workshop on Programming Theory*, pages 54–56. DIKU, University of Copenhagen.

References II


Hoare, C. (1978).
Communicating sequential processes.
Commun. ACM, 21:666–677.


Merlin, P. M. (1974).
A study of the recoverability of computing systems.
PhD thesis, University of California, Irvine.


Ouaknine, J. and Worrell, J. (2003).
Timed CSP = closed timed ϵ-automata.
Nordic J. of Computing, 10:99–133.


Schneider, S. (2000).
Concurrent and Real-time Systems.
John Wiley and Sons.


Sun, J., Liu, Y., Dong, J., and Zhang, X. (2009a).
Verifying Stateful Timed CSP Using Implicit Clocks and Zone Abstraction.
In *ICFEM'09*, volume 5885 of *LNCS*, pages 581–600. Springer.


Sun, J., Liu, Y., Dong, J. S., and Pang, J. (2009b).
PAT: Towards flexible verification under fairness.
In *CAV'09*, volume 5643 of *LNCS*. Springer.