# Relational Invariants

## for Verification of Parameterized Timed Systems

(Ongoing Work)

**Hossein Hojjat** [1]

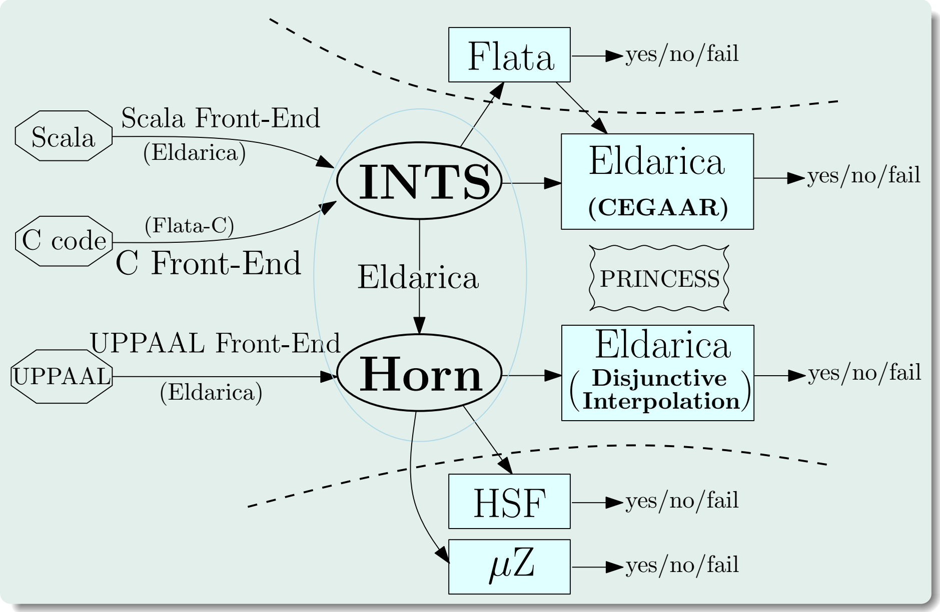
Philipp Rümmer [2] Pavle Subotic [2] Viktor Kuncak [1] Wang Yi [2]

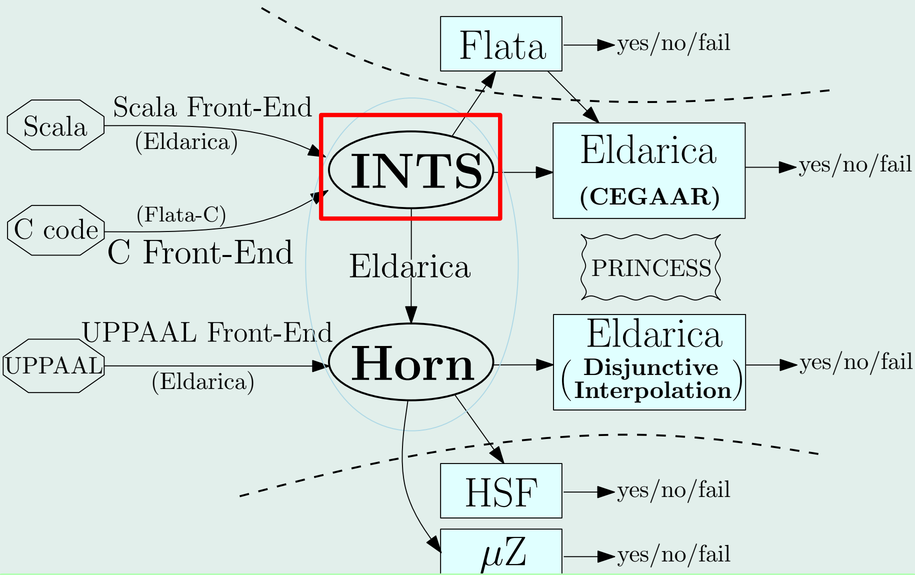
[1]École Polytechnique Fédérale de Lausanne

[2]Uppsala University

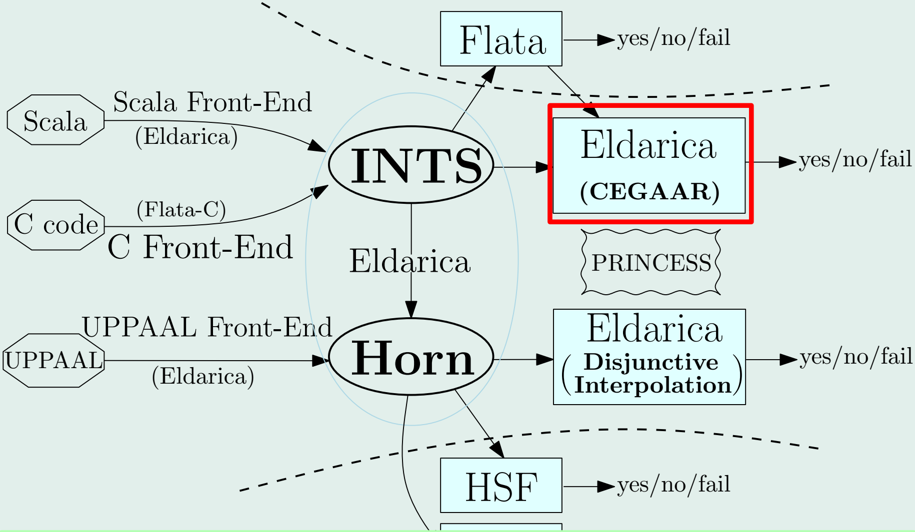
Final COST Action Meeting, Madrid

October 18, 2013

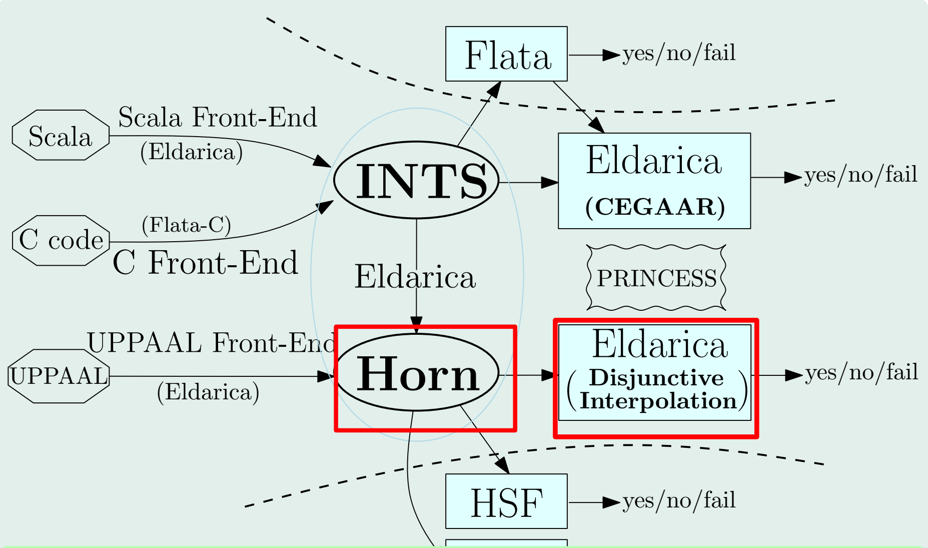

Flata
yes/no/fail
Scala Front-End
Scala
(Eldarica)
INTS
Eldarica
(CEGAAR)
yes/no/fail
(Flata-C)
C code
C Front-End
Eldarica
PRINCESS
UPPAAL Front-End
UPPAAL
(Eldarica)
Horn
Eldarica
(Disjunctive Interpolation)
yes/no/fail
HSF
yes/no/fail
μZ
yes/no/fail



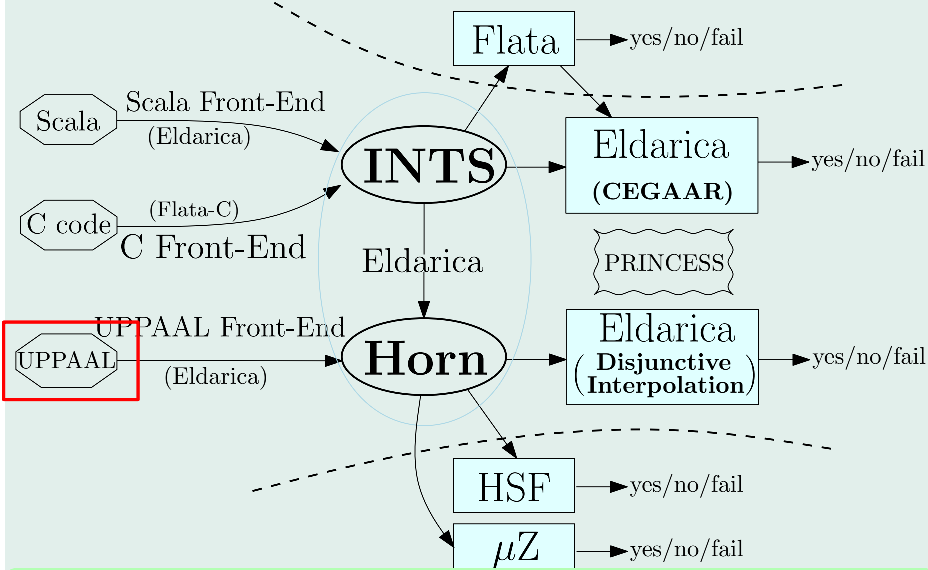

- Numerical Transition Systems (FM'12)
- Control Flow Graphs where edges are annotated by Presburger arithmetic formulas




- **C**ounter**E**xample-**G**uided **A**ccelerated **A**bstraction **R**efinement - CEGAAR (ATVA'12)
- Computes inductive interpolants from Craig interpolants and transitive closures of loops




- Disjunctive Interpolants for Horn-Clause Verification (CAV'13)
- Classifying and Solving Horn Clauses for Verification (VSTTE'13)
- Relation between different fragments of Horn clauses and Craig interpolation to refine abstractions



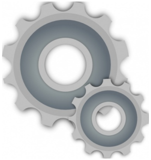

- The engine supports inter-procedural analysis
- Next mission:
  Verification of (parameterized) concurrent timed systems

- Using **Horn clauses** as an intermediate language is promising for modeling and verifying software
    - Sergey Grebenshchikov, Nuno P. Lopes, Corneliu Popeea, Andrey Rybalchenko: **“Synthesizing Software Verifiers from Proof Rules”**. PLDI 2012

Code


Safety Description

$$\forall \bar{v}.\ \Phi^0(\bar{v}) \wedge R_1^0(\bar{v}) \wedge \cdots \wedge R_n^0(\bar{v}) \rightarrow R_0^0(\bar{v})$$
$$\forall \bar{v}.\ \Phi^1(\bar{v}) \wedge R_1^1(\bar{v}) \wedge \cdots \wedge R_n^1(\bar{v}) \rightarrow R_0^1(\bar{v})$$
$$\vdots$$
$$\forall \bar{v}.\ \Phi^m(\bar{v}) \wedge R_1^m(\bar{v}) \wedge \cdots \wedge R_n^m(\bar{v}) \rightarrow R_0^m(\bar{v})$$
$$\forall \bar{v}.\ \Phi^i(\bar{v}) \wedge R_1^i(\bar{v}) \wedge \cdots \wedge R_n^i(\bar{v}) \rightarrow \mathit{false}$$



Horn Clause Solver

# Horn clauses

## Context

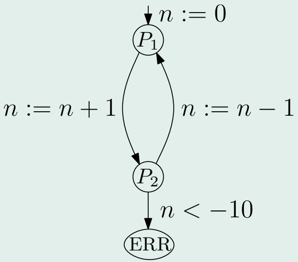
- $\mathcal{R}$: set of relation symbols with fixed arity
- $\mathcal{X}$: set of first-order variables
- $\mathcal{L}$: constraint language e.g. Presburger arithmetic

A **Horn clause** is a formula

$$C \wedge B_1 \wedge \cdots \wedge B_n \rightarrow H$$

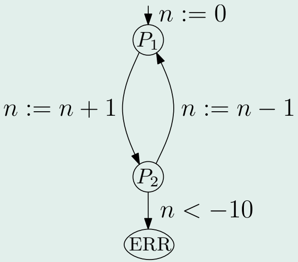
- $C$: constraint over $\mathcal{L}$ and $\mathcal{X}$ not containing symbols from $\mathcal{R}$
- $B_i$: application of $r \in \mathcal{R}$ to first-order terms $t_0, \cdots, t_n$ over $\mathcal{L}$, $\mathcal{X}$: $\quad r(t_0, \cdots, t_n)$
- $H$: *false*, or application of a relation symbol to first-order terms similar to $B_i$

# Simple Counter - Hoare Style Proof




- How to prove that ERR is unreachable?

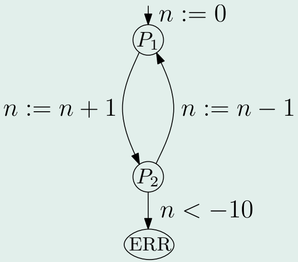
# Simple Counter - Hoare Style Proof




- How to prove that ERR is unreachable?
- We need invariants $P_1(n)$ and $P_2(n)$
- These invariants have to satisfy conditions:

$$\begin{array}{rcl} (n = 0) & \rightarrow & P_1(n) \\ P_1(n) \wedge (n' = n + 1) & \rightarrow & P_2(n') \\ P_2(n) \wedge (n' = n - 1) & \rightarrow & P_1(n') \\ P_2(n) \wedge (n < -10) & \rightarrow & \mathit{false} \end{array}$$

# Simple Counter - Hoare Style Proof

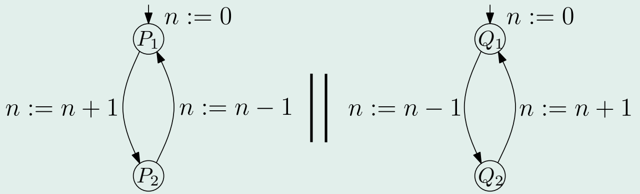



- How to prove that ERR is unreachable?
- We need invariants $P_1(n)$ and $P_2(n)$
- These invariants have to satisfy conditions:

$$\begin{array}{rcl} (n = 0) & \rightarrow & P_1(n) \\ P_1(n) \wedge (n' = n + 1) & \rightarrow & P_2(n') \\ P_2(n) \wedge (n' = n - 1) & \rightarrow & P_1(n') \\ P_2(n) \wedge (n < -10) & \rightarrow & \mathit{false} \end{array}$$

- Solvable: $P_1(n) \equiv (n \geq 0)$ and $P_2(n) \equiv (n \geq 1)$

# Concurrent Counters




## Left Thread

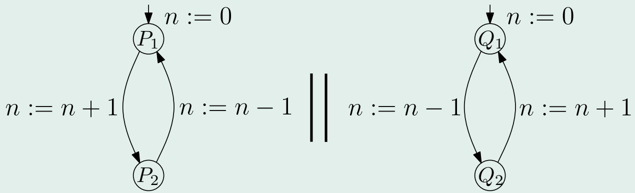
$$
\begin{array}{rcl}
n = 0 & \to & P_1(n) \\
P_1(n) \wedge n' = n + 1 & \to & P_2(n') \\
P_2(n) \wedge n' = n - 1 & \to & P_1(n')
\end{array}
$$

## Right Thread

$$
\begin{array}{rcl}
n = 0 & \to & Q_1(n) \\
Q_1(n) \wedge n' = n - 1 & \to & Q_2(n') \\
Q_2(n) \wedge n' = n + 1 & \to & Q_1(n')
\end{array}
$$

$$Q_2(n) \wedge P_2(n) \wedge (n = 0) \to \mathit{false}$$

# Concurrent Counters




## Left Thread

$$
\begin{array}{rcl}
n = 0 & \rightarrow & P_1(n) \\
P_1(n) \wedge n' = n + 1 & \rightarrow & P_2(n') \\
P_2(n) \wedge n' = n - 1 & \rightarrow & P_1(n')
\end{array}
$$

## Right Thread

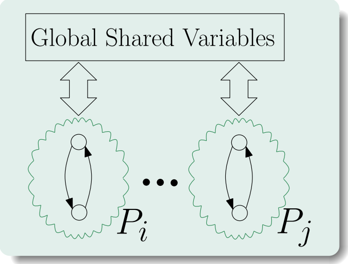
$$
\begin{array}{rcl}
n = 0 & \rightarrow & Q_1(n) \\
Q_1(n) \wedge n' = n - 1 & \rightarrow & Q_2(n') \\
Q_2(n) \wedge n' = n + 1 & \rightarrow & Q_1(n')
\end{array}
$$

$$Q_2(n) \wedge P_2(n) \wedge (n = 0) \rightarrow \mathit{false}$$

- **Unsound:** proves to be correct although the real system does not have the property

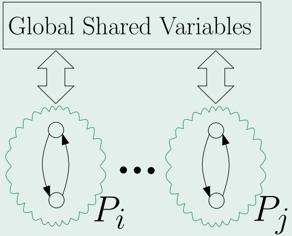
# Concurrency

**Interference** with process $P_i$ are the interleaved updates to global variables from another process $P_j$ ($j \neq i$)



# Concurrency

**Interference** with process $P_i$ are the interleaved updates to global variables from another process $P_j$ $(j \neq i)$

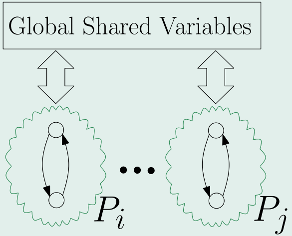



Two classical proof methods to capture interference:

1. **Owicki-Gries:** A transition by $P_j$ should not violate the local invariant of $P_i$
2. **Rely-Guarantee:** Model all the interferences caused by other processes to $P_i$ using an environment $E_i$

# Concurrency

**Interference** with process $P_i$ are the interleaved updates to global variables from another process $P_j$ $(j \neq i)$

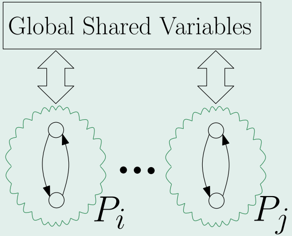



Two classical proof methods to capture interference:

1. **Owicki-Gries:** A transition by $P_j$ should not violate the local invariant of $P_i$
2. **Rely-Guarantee:** Model all the interferences caused by other processes to $P_i$ using an environment $E_i$

# Concurrency

**Interference** with process $P_i$ are the interleaved updates to global variables from another process $P_j$ ($j \neq i$)

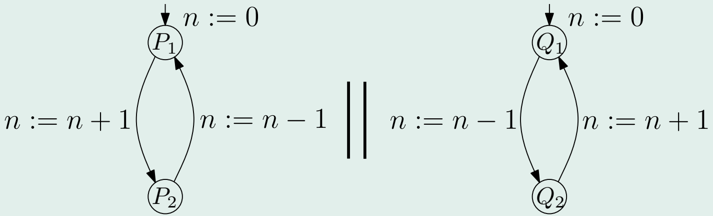



Two classical proof methods to capture interference:

1. **Owicki-Gries:** A transition by $P_j$ should not violate the local invariant of $P_i$
2. **Rely-Guarantee:** Model all the interferences caused by other processes to $P_i$ using an environment $E_i$

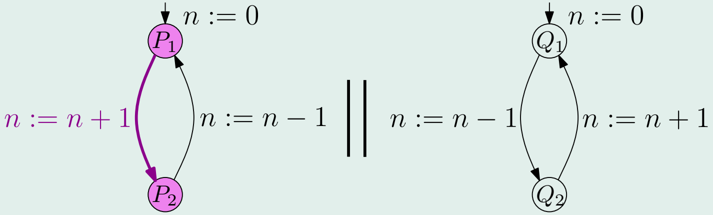
- Completeness in Owicki-Gries can be achieved by
  - Adding auxiliary history variables
  - Sharing the local state among the processes

# Owicki-Gries Interference-Free Conditions



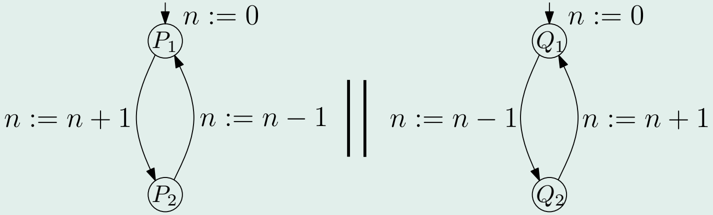

$$
\begin{array}{l}
P_1(n,1) \wedge Q_1(n,1) \wedge n' = n + 1 \rightarrow Q_1(n',2) \\
P_1(n,2) \wedge Q_2(n,1) \wedge n' = n + 1 \rightarrow Q_2(n',2) \\
P_2(n,1) \wedge Q_1(n,2) \wedge n' = n - 1 \rightarrow Q_1(n',1) \\
P_2(n,2) \wedge Q_2(n,2) \wedge n' = n - 1 \rightarrow Q_2(n',1) \\
Q_1(n,1) \wedge P_1(n,1) \wedge n' = n - 1 \rightarrow P_1(n',2) \\
Q_1(n,2) \wedge P_2(n,1) \wedge n' = n - 1 \rightarrow P_2(n',2) \\
Q_2(n,1) \wedge P_1(n,2) \wedge n' = n + 1 \rightarrow P_1(n',1) \\
Q_2(n,2) \wedge P_2(n,2) \wedge n' = n + 1 \rightarrow P_2(n',1)
\end{array}
$$

# Owicki-Gries Interference-Free Conditions




$$
\begin{aligned}
&P_1(n,1) \wedge Q_1(n,1) \wedge n' = n+1 \rightarrow Q_1(n',2)\\
&P_1(n,2) \wedge Q_2(n,1) \wedge n' = n+1 \rightarrow Q_2(n',2)\\
&P_2(n,1) \wedge Q_1(n,2) \wedge n' = n-1 \rightarrow Q_1(n',1)\\
&P_2(n,2) \wedge Q_2(n,2) \wedge n' = n-1 \rightarrow Q_2(n',1)\\
&Q_1(n,1) \wedge P_1(n,1) \wedge n' = n-1 \rightarrow P_1(n',2)\\
&Q_1(n,2) \wedge P_2(n,1) \wedge n' = n-1 \rightarrow P_2(n',2)\\
&Q_2(n,1) \wedge P_1(n,2) \wedge n' = n+1 \rightarrow P_1(n',1)\\
&Q_2(n,2) \wedge P_2(n,2) \wedge n' = n+1 \rightarrow P_2(n',1)
\end{aligned}
$$

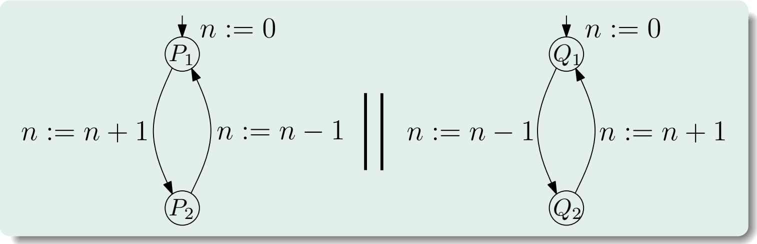
# Monolithic Encoding




- Uses only one relation symbol to model the system: $\mathbf{R}(id, n, t_1, t_2)$
- Invariant covering the whole system
- Simpler and creates more elegant solutions

$$\begin{array}{rcl}
(n = 0) \land (t_1 = 1) \land (t_2 = 1) & \rightarrow & \mathbf{R}(id, n, t_1, t_2) \\
\mathbf{R}(1, n, 1, t_2) \land (n' = n + 1) & \rightarrow & \mathbf{R}(1, n', 2, t_2) \\
\mathbf{R}(1, n, 2, t_2) \land (n' = n - 1) & \rightarrow & \mathbf{R}(1, n', 1, t_2) \\
\mathbf{R}(2, n, t_1, 1) \land (n' = n - 1) & \rightarrow & \mathbf{R}(2, n', t_1, 2) \\
\mathbf{R}(2, n, t_1, 2) \land (n' = n + 1) & \rightarrow & \mathbf{R}(2, n', t_1, 1)
\end{array}$$

# Monolithic Encoding

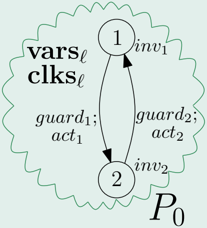



## Interference-Free Conditions

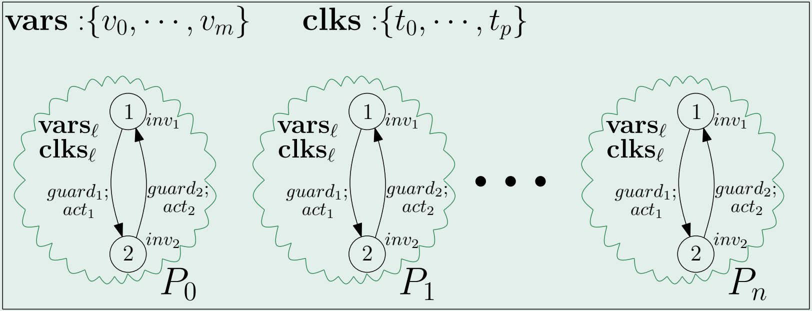
$$
\begin{aligned}
\mathbf{R}(1, n, 1, t2) \wedge \mathbf{R}(2, n, 1, t_2) \wedge (n' = n + 1) &\quad\rightarrow\quad \mathbf{R}(2, n', 2, t_2)\\
\mathbf{R}(1, n, 2, t_2) \wedge \mathbf{R}(2, n, 2, t_2) \wedge (n' = n - 1) &\quad\rightarrow\quad \mathbf{R}(2, n', 1, t_2)\\
\mathbf{R}(2, n, t_1, 1) \wedge \mathbf{R}(1, n, t_1, 1) \wedge (n' = n - 1) &\quad\rightarrow\quad \mathbf{R}(1, n', t_1, 2)\\
\mathbf{R}(2, n, t_1, 2) \wedge \mathbf{R}(1, n, t_1, 2) \wedge (n' = n + 1) &\quad\rightarrow\quad \mathbf{R}(1, n', t_1, 1)
\end{aligned}
$$

# Timed Process

$\mathbf{vars}:\{v_0, \cdots, v_m\}$ $\mathbf{clks}:\{t_0, \cdots, t_p\}$

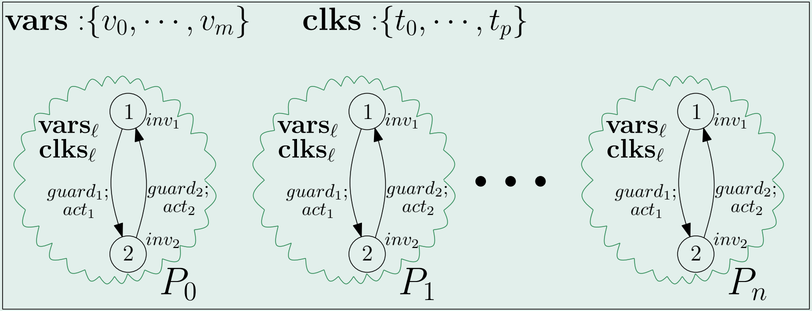


# Parameterized Timed System



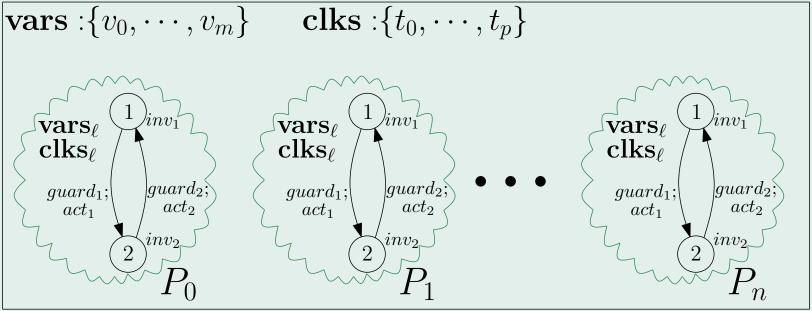

- A parameterized system consists of an arbitrary number of processes

# Parameterized Timed System




- A parameterized system consists of an arbitrary number of processes
- Verification of parameterized systems is beyond the reach of traditional finite-state model checkers

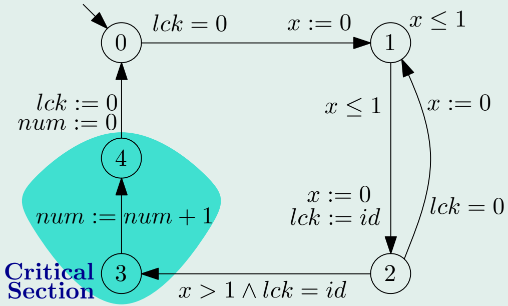
# Parameterized Timed System




- A parameterized system consists of an arbitrary number of processes
- Verification of parameterized systems is beyond the reach of traditional finite-state model checkers
- We use the approach of solving Horn clauses to prove safety

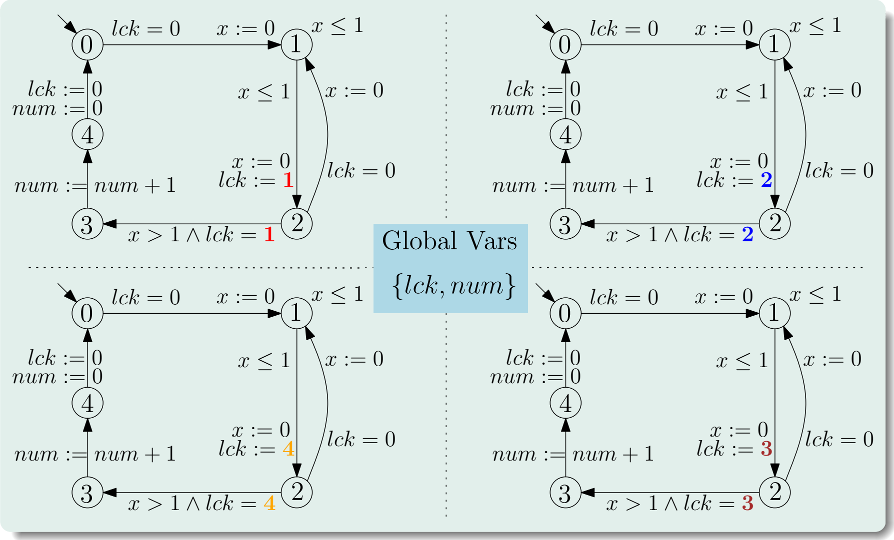
# Fischer's Mutual Exclusion Protocol

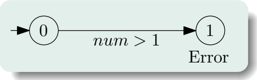
- Global Variables: $\{lck, num\}$
- Local Variable: $id \neq 0$ which is unique
- Local Clock: $x$



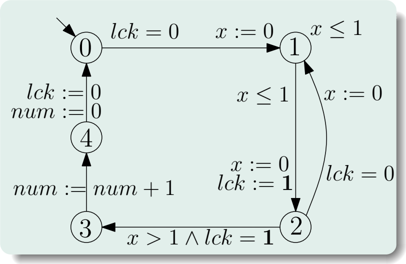

After waiting 1 time unit only one process has the right for entering CS

# A Safety Property for Fischer's Protocol

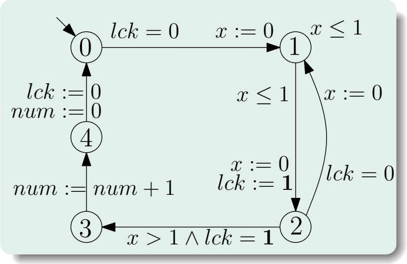


# Horn Clauses for Fischer's Protocol

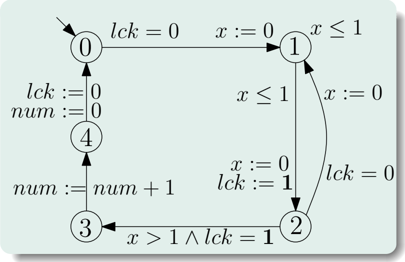


# Horn Clauses for Fischer's Protocol




$$\mathbf{P}(\underset{\text{global clock}}{c}, \underbrace{num, lck}_{\text{global vars}}, id, \underset{\text{local clock}}{x}, \underset{\text{position}}{t})$$

# Horn Clauses for Fischer's Protocol




$$\mathbf{P}(c, \underbrace{num, lck}_{\text{global vars}}, id, x, t)$$

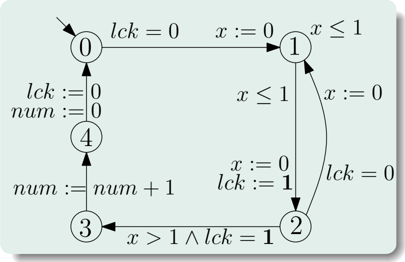
global clock

local clock

position

- Time is measured relative to a global clock $c$

# Horn Clauses for Fischer's Protocol



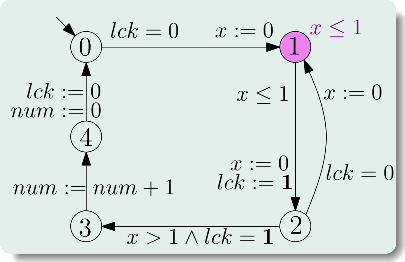

$$\mathbf{P}(\underset{\text{global clock}}{c}, \underbrace{num, lck}_{\text{global vars}}, id, \underset{\text{local clock}}{x}, \underset{\text{position}}{t})$$

- Time is measured relative to a global clock $c$

## Initialization Clause

$$(num = 0) \wedge (lck = 0) \wedge (id \neq 0) \wedge (x = c) \wedge (t = 0) \longrightarrow \mathbf{P}(c, num, lck, id, x, t)$$

# Horn Clauses for Fischer's Protocol

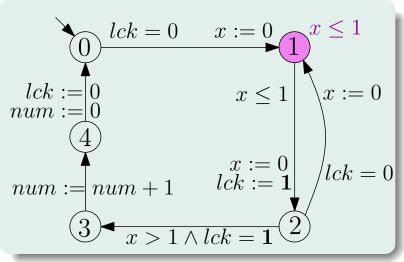



$$\mathbf{P}(\underbrace{c}_{\text{global clock}}, \underbrace{num, lck}_{\text{global vars}}, id, \underbrace{x}_{\text{local clock}}, \underbrace{t}_{\text{position}})$$

## Time Elapse

- $\mathbf{P}(c, num, lck, id, x, t) \wedge (c' \geq c) \wedge (t \neq 1) \longrightarrow \mathbf{P}(c', num, lck, id, x, t)$

# Horn Clauses for Fischer's Protocol




$$\mathbf{P}(c, \underbrace{num, lck}_{\text{global vars}}, id, x, t)$$

global clock

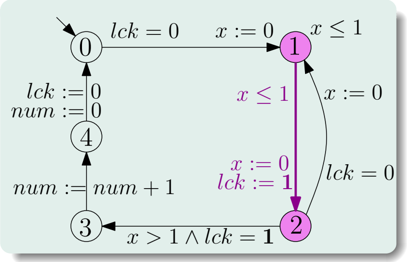
local clock

position

## Time Elapse

- $\mathbf{P}(c, num, lck, id, x, t) \wedge (c' \geq c) \wedge (t \neq 1) \longrightarrow \mathbf{P}(c', num, lck, id, x, t)$
- $\mathbf{P}(c, num, lck, id, x, t) \wedge (c' \geq c) \wedge (t = 1) \wedge (c' - x \leq 1) \longrightarrow \mathbf{P}(c', num, lck, id, x, t)$

# Horn Clauses for Fischer's Protocol




$$\mathbf{P}(c, \underbrace{num, lck}_{\text{global vars}}, id, x, t)$$

global clock

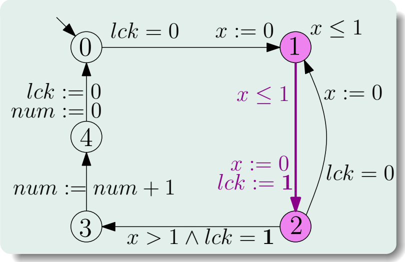
local clock

position

## Local Transition

- We associate one clause to each transition
- Transition from 1 to 2

# Horn Clauses for Fischer's Protocol



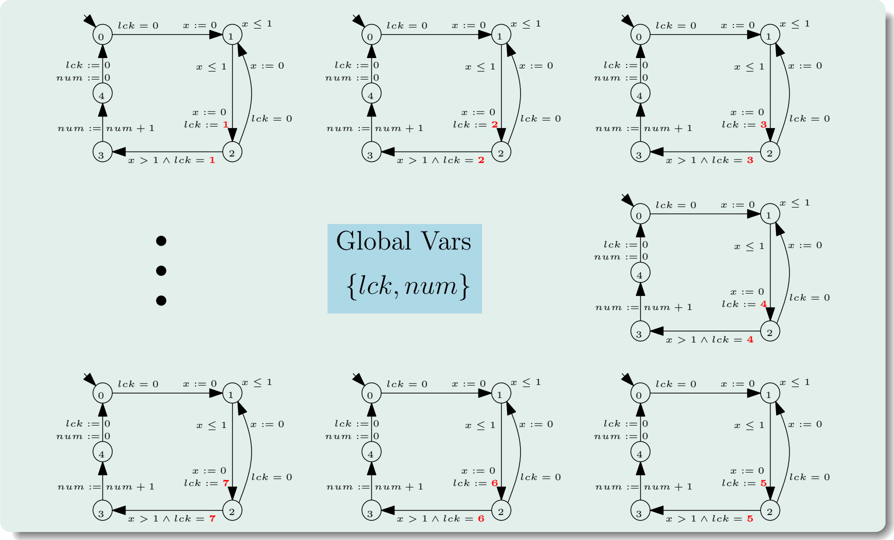

$$\mathbf{P}(c, \underbrace{num, lck}_{\text{global vars}}, id, x, t)$$

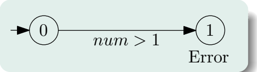
global clock  local clock  position

## Local Transition

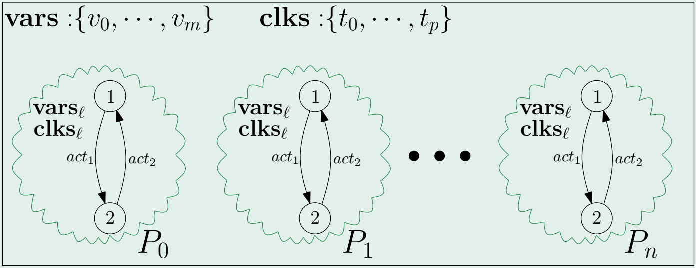
- We associate one clause to each transition
- Transition from 1 to 2
  - $\mathbf{P}(c, num, lck, id, x, 1) \wedge (c - x \leq 1) \wedge (x' = c) \wedge (lck' = id) \longrightarrow \mathbf{P}(c, num, lck', id, x', 2)$

# Parameterized Fischer's Protocol



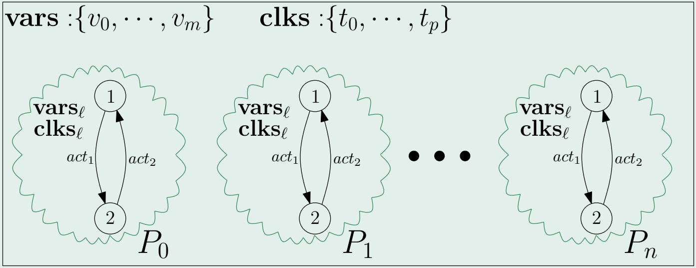
# Invariant for Parameterized System




$\mathbf{P}(id, global, local)$

- It is impossible to promote the local state to global scope in a parameterized system

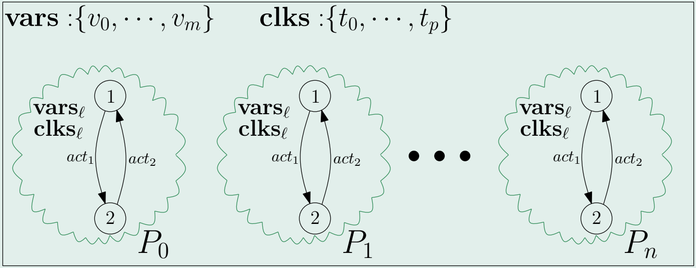
# Invariant for Parameterized System




$$\mathbf{P}(id, global, local)$$

- It is impossible to promote the local state to global scope in a parameterized system
- The relation symbol **P** is not able to talk about different distinct processes

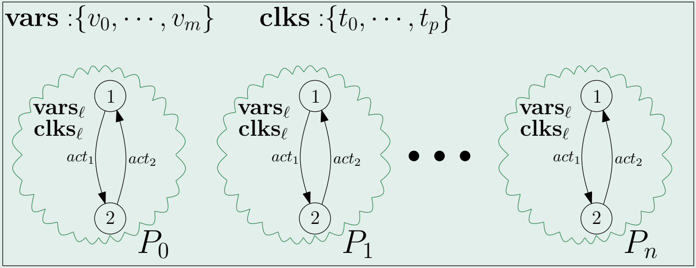
# Invariant for Parameterized System




$\mathbf{P}(id, global, local)$

- It is impossible to promote the local state to global scope in a parameterized system
- The relation symbol **P** is not able to talk about different distinct processes
  - Mutual Exclusion: $P_i$ and $P_j$ ($i \neq j$) cannot be in some particular control state at the same time

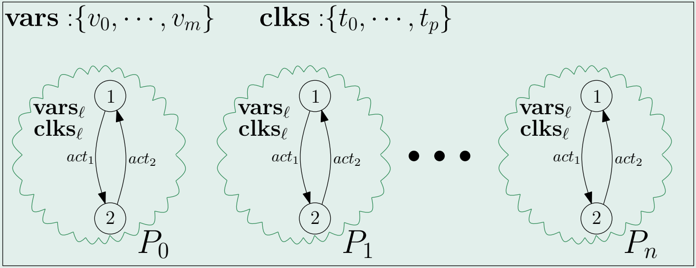
# Relational Invariants




$$\mathbf{P_k}(global, id_1, local_1, \cdots, id_k, local_k)$$

- The relational invariant $\mathbf{P_k}$ can talk about the global state and $k$ pairs of (pairwise distinct) process identifiers and local states

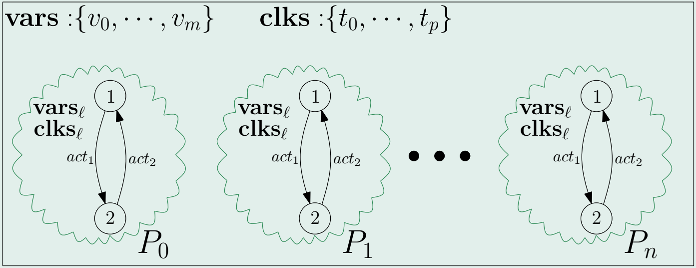
# Relational Invariants




$$\mathbf{P_k}(global, id_1, local_1, \cdots, id_k, local_k)$$

- The relational invariant $\mathbf{P_k}$ can talk about the global state and $k$ pairs of (pairwise distinct) process identifiers and local states
- $\mathbf{P_k}$ can express which combinations of states of $k$ processes can occur simultaneously
  - possible to encode properties such as mutual exclusion

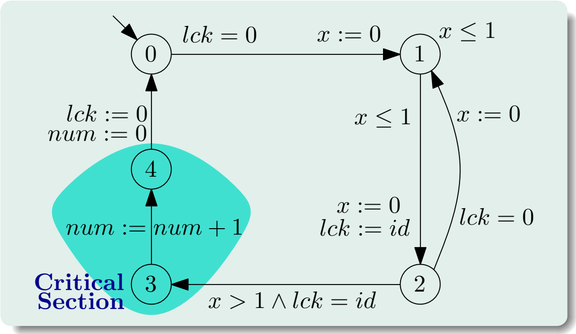
# Relational Invariants




$$\mathbf{P_k}(global, id_1, local_1, \cdots, id_k, local_k)$$

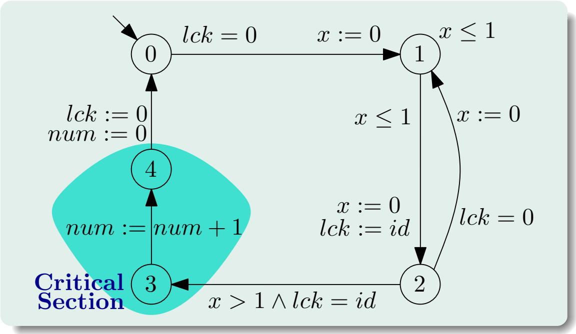
- The relational invariant $\mathbf{P_k}$ can talk about the global state and $k$ pairs of (pairwise distinct) process identifiers and local states
- $\mathbf{P_k}$ can express which combinations of states of $k$ processes can occur simultaneously
  - possible to encode properties such as mutual exclusion
- For $k = 1$, relational invariants reduce to Owicki-Gries style reasoning

# Parameterized Fischer's Protocol



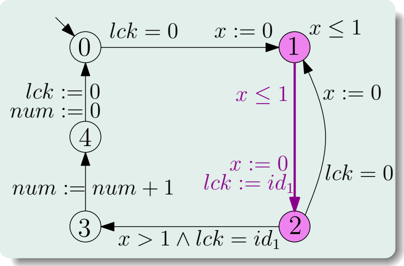

- 1-invariant is not strong to verify the parameterized Fischer protocol
  - $\mathbf{P}(c, num, lck, \boxed{id, x, t})$

# Parameterized Fischer's Protocol




- 1-invariant is not strong to verify the parameterized Fischer protocol
  - $\mathbf{P}(c, num, lck, \boxed{id, x, t})$
- We use 2-invariant for this purpose
  - $\mathbf{P}(c, num, lck, \boxed{id_1, x_1, t_1}, \boxed{id_2, x_2, t_2})$

# Horn Clauses for Parameterized Fischer's Protocol



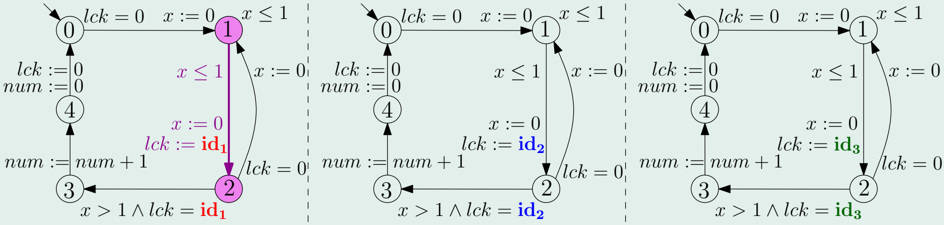

$$\mathbf{P}(c, \underbrace{num, lck}_{\text{global vars}}, \boxed{id_1, x_1, t_1}, \boxed{id_2, x_2, t_2})$$

## Local Transition

- Transition from 1 to 2

$$\begin{aligned} &\mathbf{P}(c, lck, num, id_1, x_1, 1, id_2, x_2, t_2) \\ &\wedge (id_1 \neq 0) \wedge (id_2 \neq 0) \wedge (id_1 \neq id_2) \\ &\wedge (c - x_1 \leq 1) \wedge (x_1' = c) \wedge (lck' = id_1) \\ \longrightarrow &\mathbf{P}(c, lck', num, id_1, x_1', 2, id_2, x_2, t_2) \end{aligned}$$

# Interference Freedom




$$
\begin{aligned}
&\mathbf{P}(c, lck, num, id_3, x_3, t_3, id_2, x_2, t_2)\\
\wedge\ &\mathbf{P}(c, lck, num, id_1, x_1, 1, id_2, x_2, t_2)\\
\wedge\ &\mathbf{P}(c, lck, num, id_1, x_1, 1, id_3, x_3, t_3)\\
\wedge\ &(id_1 \neq 0) \wedge (id_2 \neq 0) \wedge (id_3 \neq 0)\\
\wedge\ &(id_1 \neq id_2) \wedge (id_2 \neq id_3) \wedge (id_1 \neq id_3)\\
\wedge\ &(c - x_1 \leq 1) \wedge (x_1' = c) \wedge (lck' = id_1)\\
\longrightarrow &\mathbf{P}(c, lck', num, id_3, x_3, t_3, id_2, x_2, t_2)
\end{aligned}
$$

# Eldarica Framework (`http://lara.epfl.ch/w/eldarica`)

- Predicate abstraction with interpolation-based counterexample-driven refinement
  - Disjunctive interpolation (CAV'13) as refinement algorithm
- For checking the feasibility of paths and constructing abstractions, Eldarica employs the provers Z3 and Princess
- Eldarica can solve Horn clauses over Presburger arithmetic as one of its input languages
- Interface to UPPAAL benchmarks
  - Finite + unbounded/infinite sets of processes
- Verified a number of (timed/untimed) benchmarks
  - Fischer Protocol
  - Train Gate Controller
  - Synchronization Barriers

# Related Work


- K. L. McMillan and A. Rybalchenko:
  **"Solving constrained Horn clauses using interpolation"**,
  Technical Report MSR-TR-2013-6, 2013.
- A. Sánchez, S. Sankaranarayanan, C. Sánchez, and E. Chang:
  **"Invariant Generation for Parametrized Systems using Self-Reflection"**, SAS, 2012.
- A. Roychoudhury, K.N. Kumar, C. R. Ramakrishnan, I. V. Ramakrishnan, S.A. Smolka:
  **"Verification of Parameterized Systems Using Logic Program Transformations"**, TACAS 2000.
- P. Rümmer, H. Hojjat, V. Kuncak:
  **"Disjunctive Interpolants for Horn-Clause Verification"**, CAV 2013.

# Conclusions

- We introduce **relational invariants** to take the relationship between multiple processes into account
- Relational invariant allows us to verify a larger class of concurrent systems
- Relational invariants show promising results in practice

# Horn Clauses for Fischer's Protocol

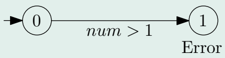
## Backup Slide

$$\mathbf{P}(c, \underbrace{num, lck}_{\text{global vars}}, id, x, t)$$

global clock — local clock — position

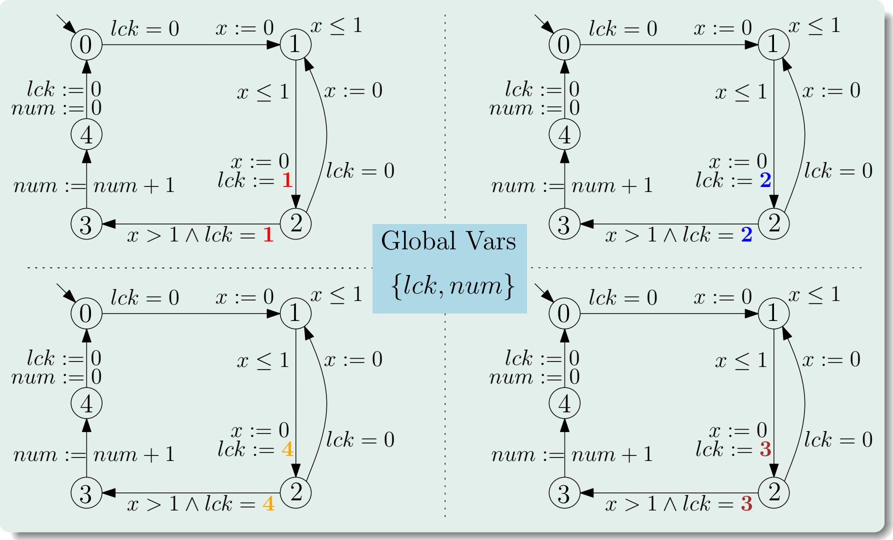
### Assertion

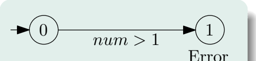
$$\mathbf{P}(c, num, lck, 1, x, t_1) \land \mathbf{P}(c, num, lck, 2, x, t_2) \land$$
$$\mathbf{P}(c, num, lck, 3, x, t_3) \land \mathbf{P}(c, num, lck, 4, x, t_4) \land$$
$$\mathbf{Observer}(c, num, lck, 1) \longrightarrow \mathit{false}$$



# Local & Global Variables

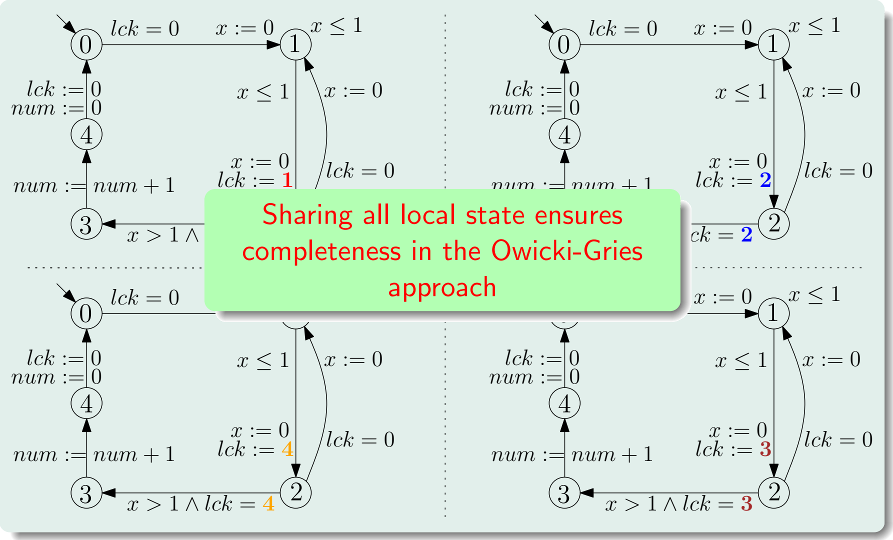
## Backup Slide



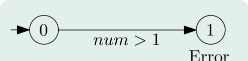



# Local & Global Variables

## Backup Slide

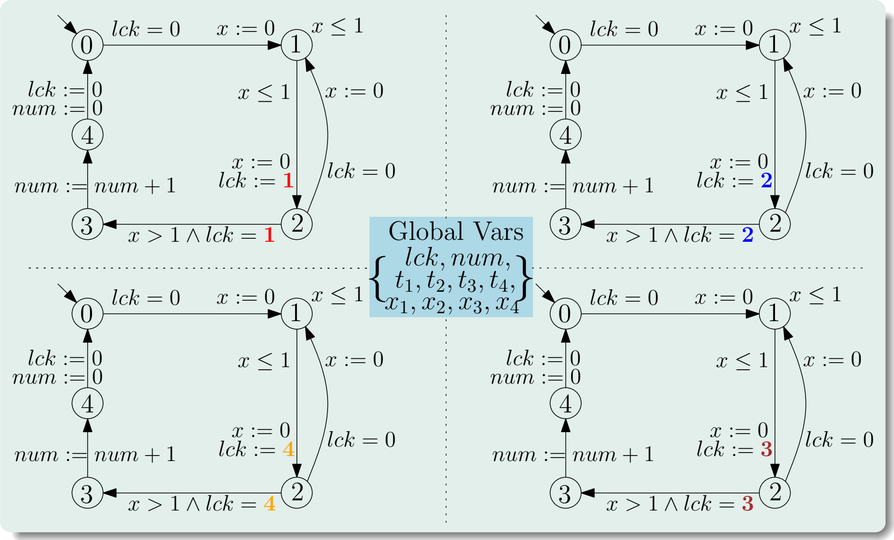


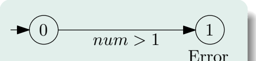

Sharing all local state ensures completeness in the Owicki-Gries approach



# Local & Global Variables

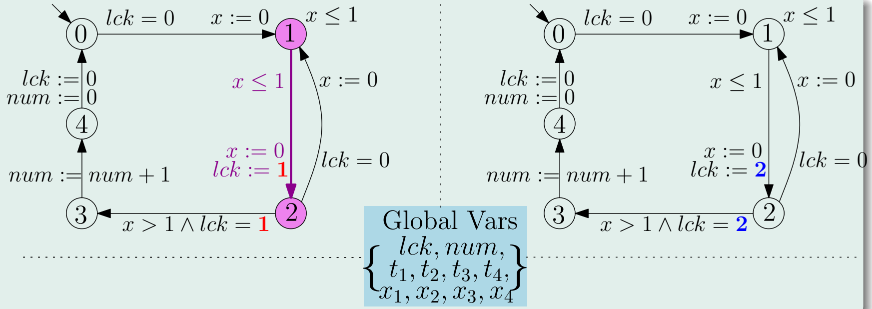
## Backup Slide



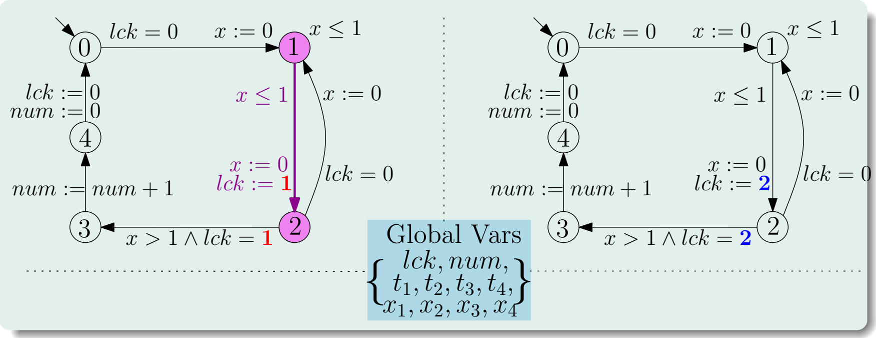
# Interference Freedom

## Backup Slide



# Interference Freedom

## Backup Slide




$$
\begin{aligned}
& \mathbf{P}(c, num, lck, \mathbf{1}, x_1, x_2, x_3, x_4, t_1, t_2, t_3, t_4) \\
\wedge\ & \mathbf{P}(c, num, lck, \mathbf{2}, x_1, x_2, x_3, x_4, t_1, t_2, t_3, t_4) \\
\wedge\ & (c - x_1 \leq 1) \wedge (x_1' = c) \wedge (lck' = 1) \\
\wedge\ & (t_1 = 1) \wedge (t_1' = 2) \\
\longrightarrow\ & \mathbf{P}(c, num, lck', \mathbf{2}, x_1', x_2, x_3, x_4, t_1', t_2, t_3, t_4)
\end{aligned}
$$