Relational Invariants
for Verification of Parameterized Timed Systems

(Ongoing Work)

Hossein Hojjat
Philipp Rümmer
Pavle Subotic
Viktor Kuncak
Wang Yi

1École Polytechnique Fédérale de Lausanne
2Uppsala University

Final COST Action Meeting, Madrid
October 18, 2013
Numerical Transition Systems (FM’12)

Control Flow Graphs where edges are annotated by Presburger arithmetic formulas
CounterExample-Guided Accelerated Abstraction Refinement - 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


\[
\forall \bar{v}. \Phi_0(\bar{v}) \land R_0^1(\bar{v}) \land \cdots \land R_0^n(\bar{v}) \rightarrow R_0^0(\bar{v}) \\
\forall \bar{v}. \Phi_1(\bar{v}) \land R_1^1(\bar{v}) \land \cdots \land R_1^n(\bar{v}) \rightarrow R_1^0(\bar{v}) \\
\vdots \\
\forall \bar{v}. \Phi_m(\bar{v}) \land R_m^1(\bar{v}) \land \cdots \land R_m^n(\bar{v}) \rightarrow R_m^0(\bar{v}) \\
\forall \bar{v}. \Phi_i(\bar{v}) \land R_i^1(\bar{v}) \land \cdots \land R_i^n(\bar{v}) \rightarrow false
\]
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

\[
C \land B_1 \land \cdots \land B_n \rightarrow H
\]

- \( C \): 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 \( \mathcal{L}, \mathcal{X} \): \( r(t_0, \cdots, t_n) \)
- \( H \): false, or application of a relation symbol to first-order terms similar to \( B_i \)
How to prove that ERR is unreachable?
How to prove that ERR is unreachable?

We need invariants $P_1(n)$ and $P_2(n)$

These invariants have to satisfy conditions:

$$(n = 0) \quad \rightarrow \quad P_1(n)$$
$$P_1(n) \land (n' = n + 1) \quad \rightarrow \quad P_2(n')$$
$$P_2(n) \land (n' = n - 1) \quad \rightarrow \quad P_1(n')$$
$$P_2(n) \land (n < -10) \quad \rightarrow \quad false$$
How to prove that ERR is unreachable?

We need invariants \( P_1(n) \) and \( P_2(n) \)

These invariants have to satisfy conditions:

\[
\begin{align*}
(n = 0) & \quad \rightarrow \quad P_1(n) \\
P_1(n) \land (n' = n + 1) & \quad \rightarrow \quad P_2(n') \\
P_2(n) \land (n' = n - 1) & \quad \rightarrow \quad P_1(n') \\
P_2(n) \land (n < -10) & \quad \rightarrow \quad false
\end{align*}
\]

Solvable: \( P_1(n) \equiv (n \geq 0) \) and \( P_2(n) \equiv (n \geq 1) \)
Concurrent Counters

\[
\begin{align*}
n &:= 0 \\
P_1(n) &\Rightarrow n := n + 1 \\
P_2(n) &\Rightarrow n := n - 1 \\
Q_1(n) &\Rightarrow n := n - 1 \\
Q_2(n) &\Rightarrow n := n + 1
\end{align*}
\]

Left Thread

\[
\begin{align*}
n &= 0 &\Rightarrow P_1(n) \\
P_1(n) \land n' &= n + 1 &\Rightarrow P_2(n') \\
P_2(n) \land n' &= n - 1 &\Rightarrow P_1(n')
\end{align*}
\]

Right Thread

\[
\begin{align*}
n &= 0 &\Rightarrow Q_1(n) \\
Q_1(n) \land n' &= n - 1 &\Rightarrow Q_2(n') \\
Q_2(n) \land n' &= n + 1 &\Rightarrow Q_1(n')
\end{align*}
\]

\[
Q_2(n) \land P_2(n) \land (n = 0) \Rightarrow false
\]
Concurrent Counters

\[
\begin{align*}
&n := 0 \\
&P_1(n) \land n' = n + 1 \rightarrow P_2(n') \\
&P_2(n) \land n' = n - 1 \rightarrow P_1(n') \\
\end{align*}
\]

\[
\begin{align*}
&n := n + 1 \\
&P_2(n) \land n' = n - 1 \rightarrow P_1(n') \\
\end{align*}
\]

\[
\begin{align*}
&n := n - 1 \\
&Q_1(n) \land n' = n - 1 \rightarrow Q_2(n') \\
&Q_2(n) \land n' = n + 1 \rightarrow Q_1(n') \\
\end{align*}
\]

\[
Q_2(n) \land P_2(n) \land (n = 0) \rightarrow false
\]

**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 $E_i$. 
Concurrent Programming

**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

\[ n := 0 \]

\[ n := n + 1 \]

\[ n := n - 1 \]

\[ n := n - 1 \]

\[ n := n + 1 \]

\[ P_1(n, 1) \land Q_1(n, 1) \land n' = n + 1 \rightarrow Q_1(n', 2) \]

\[ P_1(n, 2) \land Q_2(n, 1) \land n' = n + 1 \rightarrow Q_2(n', 2) \]

\[ P_2(n, 1) \land Q_1(n, 2) \land n' = n - 1 \rightarrow Q_1(n', 1) \]

\[ P_2(n, 2) \land Q_2(n, 2) \land n' = n - 1 \rightarrow Q_2(n', 1) \]

\[ Q_1(n, 1) \land P_1(n, 1) \land n' = n - 1 \rightarrow P_1(n', 2) \]

\[ Q_1(n, 2) \land P_2(n, 1) \land n' = n - 1 \rightarrow P_2(n', 2) \]

\[ Q_2(n, 1) \land P_1(n, 2) \land n' = n + 1 \rightarrow P_1(n', 1) \]

\[ Q_2(n, 2) \land P_2(n, 2) \land n' = n + 1 \rightarrow P_2(n', 1) \]
Owicki-Gries Interference-Free Conditions

\[ P_1(n, 1) \land Q_1(n, 1) \land n' = n + 1 \rightarrow Q_1(n', 2) \]
\[ P_1(n, 2) \land Q_2(n, 1) \land n' = n + 1 \rightarrow Q_2(n', 2) \]
\[ P_2(n, 1) \land Q_1(n, 2) \land n' = n - 1 \rightarrow Q_1(n', 1) \]
\[ P_2(n, 2) \land Q_2(n, 2) \land n' = n - 1 \rightarrow Q_2(n', 1) \]
\[ Q_1(n, 1) \land P_1(n, 1) \land n' = n - 1 \rightarrow P_1(n', 2) \]
\[ Q_1(n, 2) \land P_2(n, 1) \land n' = n - 1 \rightarrow P_2(n', 2) \]
\[ Q_2(n, 1) \land P_1(n, 2) \land n' = n + 1 \rightarrow P_1(n', 1) \]
\[ Q_2(n, 2) \land P_2(n, 2) \land n' = n + 1 \rightarrow P_2(n', 1) \]
Monolithic Encoding

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

$(n = 0) \land (t_1 = 1) \land (t_2 = 1) \rightarrow R(id, n, t_1, t_2)$
$R(1, n, 1, t_2) \land (n' = n + 1) \rightarrow R(1, n', 2, t_2)$
$R(1, n, 2, t_2) \land (n' = n - 1) \rightarrow R(1, n', 1, t_2)$
$R(2, n, t_1, 1) \land (n' = n - 1) \rightarrow R(2, n', t_1, 2)$
$R(2, n, t_1, 2) \land (n' = n + 1) \rightarrow R(2, n', t_1, 1)$
Monolithic Encoding

Interference-Free Conditions

\[
\begin{align*}
\mathbf{R}(1, n, 1, t_2) \land \mathbf{R}(2, n, 1, t_2) \land (n' = n + 1) & \Rightarrow \mathbf{R}(2, n', 2, t_2) \\
\mathbf{R}(1, n, 2, t_2) \land \mathbf{R}(2, n, 2, t_2) \land (n' = n - 1) & \Rightarrow \mathbf{R}(2, n', 1, t_2) \\
\mathbf{R}(2, n, t_1, 1) \land \mathbf{R}(1, n, t_1, 1) \land (n' = n - 1) & \Rightarrow \mathbf{R}(1, n', t_1, 2) \\
\mathbf{R}(2, n, t_1, 2) \land \mathbf{R}(1, n, t_1, 2) \land (n' = n + 1) & \Rightarrow \mathbf{R}(1, n', t_1, 1)
\end{align*}
\]
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.
A parameterized system consists of an arbitrary number of processes
A parameterized system consists of an arbitrary number of processes

Verification of parameterized systems is beyond the reach of traditional finite-state model checkers
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

Global Vars
\{lck, num\}

0
\(lck = 0\) \(x := 0\) \(x \leq 1\)

1
\(x \leq 1\) \(x := 0\)

2
\(lck := 1\) \(lck = 0\)

3
\(num := num + 1\) \(lck := 1\)

0
\(lck = 0\) \(x := 0\) \(x \leq 1\)

1
\(x \leq 1\) \(x := 0\)

2
\(lck := 2\) \(lck = 0\)

3
\(num := num + 1\) \(lck := 2\)

0
\(lck = 0\) \(x := 0\) \(x \leq 1\)

1
\(x \leq 1\) \(x := 0\)

2
\(lck := 3\) \(lck = 0\)

3
\(num := num + 1\) \(lck := 3\)

0
\(num > 1\) \text{Error}

1

2

3

4

x \geq 1 \land lck = 1

num > 1 \text{Error}
Horn Clauses for Fischer’s Protocol

\begin{align*}
\text{P}(c, \text{num}, \text{lck}, \text{id}, x, t) \\
\text{global clock} \\
\text{local clock} \\
\text{position} \\
\text{global vars} \\
\text{11} \\
\text{0} \\
\text{4} \\
\text{3} \\
\text{2} \\
\text{x} \leq 1 \\
x := 0 \\
x := 0 \\
x := 0 \\
x := 0 \\
\text{lck} := 1 \\
\text{lck} := 1 \\
\text{lck} := 0 \\
\text{lck} := 0 \\
\text{num} := 0 \\
\text{num} := 0 \\
\text{num} := \text{num} + 1 \\
x > 1 \land \text{lck} = 1 \\
x \leq 1 \\
x \leq 1 \\
x := 0 \\
x := 0 \\
x := 0 \\
x := 0 \\
\text{lck} := 1 \\
\text{lck} := 0 \\
\text{num} := 0 \\
\text{num} := 0 \\
\text{num} := \text{num} + 1 \\
x > 1 \land \text{lck} = 1 \\
x \leq 1
\end{align*}
Horn Clauses for Fischer’s Protocol

\[ P(c, num, lck, id, x, t) \]

- **Global clock**: \( lck = 0 \) \( x := 0 \) \( x \leq 1 \)
- **Local clock**: \( lck := 0 \) \( num := 0 \) \( num := num + 1 \) \( x := 0 \) \( lck := 1 \) \( lck = 0 \)
- **Position**: \( x > 1 \land lck = 1 \)
Horn Clauses for Fischer’s Protocol

\[ P(c, \text{num}, \text{lck}, \text{id}, x, t) \]

- **Global vars**:
  - \( lck = 0 \)
  - \( x := 0 \)
  - \( x \leq 1 \)
  - \( x := 0 \)
  - \( lck := 1 \)
  - \( lck = 0 \)
  - \( num := num + 1 \)

- **Local clock**: \( x > 1 \land lck = 1 \)

- **Position**: \( x \leq 1 \)

- **Global clock**: \( c \)

- **Time is measured relative to a global clock \( c \)**
Horn Clauses for Fischer’s Protocol

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

Initialization Clause

$$(num = 0) \land (lck = 0) \land (id \neq 0) \land (x = c) \land (t = 0) \rightarrow P(c, num, lck, id, x, t)$$
Horn Clauses for Fischer’s Protocol

\[ P(c, \text{num}, lck, id, x, t) \]

\[ \land \quad (c' \geq c) \land (t \neq 1) \quad \rightarrow \quad P(c', \text{num}, lck, id, x, t) \]
Horn Clauses for Fischer’s Protocol

Time Elapse

- \( P(c, num, lck, id, x, t) \land (c' \geq c) \land (t \neq 1) \longrightarrow P(c', num, lck, id, x, t) \)
- \( P(c, num, lck, id, x, t) \land (c' \geq c) \land (t = 1) \land (c' - x \leq 1) \longrightarrow P(c', num, lck, id, x, t) \)
Horn Clauses for Fischer’s Protocol

Local Transition

- We associate one clause to each transition
- Transition from 1 to 2
Horn Clauses for Fischer’s Protocol

Local Transition

- We associate one clause to each transition
- Transition from 1 to 2
  - \( P(c, \text{num}, lck, id, x, 1) \land (c - x \leq 1) \land (x' = c) \land (lck' = id) \rightarrow P(c, \text{num}, lck', id, x', 2) \)
Parameterized Fischer’s Protocol

Global Vars

\{lck, num\}

Error

\(num > 1\)
It is impossible to promote the local state to global scope in a parameterized system
Invariant for Parameterized System

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

- 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.

vars : $\{v_0, \cdots, v_m\}$

clks : $\{t_0, \cdots, t_p\}$

$P(id, \text{global, local})$
The relational invariant $P_k$ can talk about the global state and $k$ pairs of (pairwise distinct) process identifiers and local states.
The relational invariant $P_k$ can talk about the global state and $k$ pairs of (pairwise distinct) process identifiers and local states.

$P_k$ can express which combinations of states of $k$ processes can occur simultaneously.

$\triangleright$ possible to encode properties such as mutual exclusion
The relational invariant $P_k$ can talk about the global state and $k$ pairs of (pairwise distinct) process identifiers and local states.

$P_k$ 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

\[ P(c, num, lck, \text{id}, x, t) \]
Parameterized Fischer’s Protocol

1-invariant is not strong to verify the parameterized Fischer protocol

\[ P(c, num, lck, id, x, t) \]

We use 2-invariant for this purpose

\[ P(c, num, lck, id_1, x_1, t_1, id_2, x_2, t_2) \]
Horn Clauses for Parameterized Fischer’s Protocol

Local Transition

- Transition from 1 to 2

\[
\begin{align*}
\mathbf{P}(c, lck, num, id_1, x_1, 1, id_2, x_2, t_2) \\
\land (id_1 \neq 0) \land (id_2 \neq 0) \land (id_1 \neq id_2) \\
\land (c - x_1 \leq 1) \land (x_1' = c) \land (lck' = id_1) \\
\rightarrow \mathbf{P}(c, lck', num, id_1, x_1', 2, id_2, x_2, t_2)
\end{align*}
\]
\( P(c, lck, num, id_3, x_3, t_3, id_2, x_2, t_2) \)
\( \land P(c, lck, num, id_1, x_1, 1, id_2, x_2, t_2) \)
\( \land P(c, lck, num, id_1, x_1, 1, id_3, x_3, t_3) \)
\( \land (id_1 \neq 0) \land (id_2 \neq 0) \land (id_3 \neq 0) \)
\( \land (id_1 \neq id_2) \land (id_2 \neq id_3) \land (id_1 \neq id_3) \)
\( \land (c - x_1 \leq 1) \land (x'_1 = c) \land (lck' = id_1) \)
\( \longrightarrow P(c, lck', num, id_3, x_3, t_3, id_2, x_2, t_2) \)
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


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

Assertion

\[ P(c, num, lck, 1, x, t_1) \land P(c, num, lck, 2, x, t_2) \land P(c, num, lck, 3, x, t_3) \land P(c, num, lck, 4, x, t_4) \land \text{Observer}(c, num, lck, 1) \rightarrow false \]
Local & Global Variables

Backup Slide

```
0  lck = 0  x := 0  x ≤ 1

1  x ≤ 1  x := 0

2  lck := 0  lck = 0

3  num := num + 1  lck := 1

4  x > 1 ∧ lck = 1

num := num + 1
```

Global Vars

\{lck, num\}

```
0  lck = 0  x := 0  x ≤ 1

1  x ≤ 1  x := 0

2  lck := 0  lck = 0

3  x > 1 ∧ lck = 2
```

```
0  lck = 0  x := 0  x ≤ 1

1  x ≤ 1  x := 0

2  lck := 0  lck = 0

3  x > 1 ∧ lck = 3
```

```
0  lck = 0  x := 0  x ≤ 1

1  x ≤ 1  x := 0
```

```
0  num > 1  num ≠ 1
```

Error
Local & Global Variables

Sharing all local state ensures completeness in the Owicki-Gries approach.
**Local & Global Variables**

**Backup Slide**

```
0 → lck = 0, x := 0, x ≤ 1 → 1

lck := 0
num := 0

num := num + 1
lck := 1

x := 0
lck = 0
num := 0

x > 1 ∧ lck = 1

0 → lck = 0, x := 0, x ≤ 1 → 1

lck := 0
num := 0

num := num + 1
lck := 2

x := 0
lck = 0
num := 0

x > 1 ∧ lck = 2

Global Vars
{lck, num, t1, t2, t3, t4, x1, x2, x3, x4} {x > 1 ∧ lck = 4
```

```
0 → lck = 0, x := 0, x ≤ 1 → 1

lck := 0
num := 0

num := num + 1
lck := 3

x := 0
lck = 0
num := 0

x > 1 ∧ lck = 3
```

```
0 → lck = 0, x := 0, x ≤ 1 → 1

lck := 0
num := 0

num := num + 1
lck := 4

x := 0
lck = 0
num := 0

x > 1 ∧ lck = 4
```

```
0 → num > 1 → Error
```

Interference Freedom

Backup Slide

Global Vars
\{lck, num, t_1, t_2, t_3, t_4, x_1, x_2, x_3, x_4\}
Interference Freedom

Backup Slide

\[
\begin{align*}
\mathcal{P}(c, \text{num}, lck, 1, x_1, x_2, x_3, x_4, t_1, t_2, t_3, t_4) \\
\wedge \mathcal{P}(c, \text{num}, lck, 2, x_1, x_2, x_3, x_4, t_1, t_2, t_3, t_4) \\
\wedge (c - x_1 \leq 1) \wedge (x_1' = c) \wedge (lck' = 1) \\
\wedge (t_1 = 1) \wedge (t_1' = 2) \\
\rightarrow \mathcal{P}(c, \text{num}, lck', 2, x_1', x_2, x_3, x_4, t_1', t_2, t_3, t_4)
\end{align*}
\]