VTS: Conformance testing
Symbolic model-based test selection

Thierry Jéron

IRISA / INRIA Rennes, France
jeron@irisa.fr
http://www.irisa.fr/prive/jeron/
Brief state of the art

- Conformance testing theory for finite state models e.g., FSM [Lee-Yann 96], ioLTS [Tretmans 96].
- On-line/on-the-fly test generation algorithms and tools e.g., TorX [Belinfante et al. 99], TGV [Jard-Jéron 04].

Successfully used on industrial size case studies, but may suffer from state explosion problems.

For large/infinite state models, solutions based on

- symbolic execution and constraint solving:
  Agatha [Gaston 06], BZ-TT [Legeard 02], Gatel [Marre-Arnoud 00], combination with random exploration: [Godefroid 05].
- abstractions: predicate abstraction [Ball 05],
  finite state generation + concretization [Calamé et al. 05].

Generate instantiated test cases i.e. finite paths
Motivating example

Test for behaviors where ![ok(p)](p \neq y - x \lor p \leq 10, ?ok(p)) is sent with ![p > 10 \land x > 0](p > 10 \land x > 0) while ![x](x) is positive.
What we need

- a model to specify reactive systems
- a model to express testing objectives
- a theory for reasoning about testing
- an algorithm to compute test cases with:
  - backward propagation of symbolic constraints
  - fix-point computation to deal with loops
  - approximation to ensure convergence
Contribution

- Conformance testing based on the ioco testing theory.
- Adapted to infinite state models: ioSTS.
- Focus on models with data variables: guards, assignments.
- Selection of test programs based on approximate analysis.
- Implemented in the STG tool.
1. The ioSTS model

2. Conformance testing theory

3. Test selection using approximate analysis

4. Test execution

5. Conclusion and perspectives
1. The ioSTS model
2. Conformance testing theory
3. Test selection using approximate analysis
4. Test execution
5. Conclusion and perspectives
IOSTS syntax

**Definition**

\[ \mathcal{M} = (V, \Theta, \Sigma, T) \]

- \( V = V_i \cup V_x \): partitioned set of (internal / external) variables
- \( \Theta \subseteq D_{V_i} \): initial condition with unique solution in \( D_{V_i} \).
- \( \Sigma = \Sigma? \cup \Sigma! \): finite alphabet of actions with communication parameters of type \( \text{sig}(a) \).
- \( T \): finite set of symbolic transitions \( t = (a, \bar{p}, G, A) \)
  - \( a \in \Sigma \): action
  - \( \bar{p} \): tuple of communication parameters local to \( t \);
  - \( G \subseteq D_V \times D_{\text{sig}(a)} \): guard.
  - \( A : D_V \times D_{\text{sig}(a)} \rightarrow D_{V_i} \): assignment.

**Assumption**

Guards are expressed in a theory in which satisfiability is decidable;
Running example: a simple lift-controller

Parameter: \( h \): integer,
Variables: \( c, g \): integer, \( pc \): \{Wait, Move, End\}
Inputs: Target?; Outputs: Up!, Down!, Stop!, Break!
Communication parameters: \( p \);
The semantics of an ioSTS $\mathcal{M} = (V, \Theta, \Sigma, T)$ is an ioLTS $\llbracket \mathcal{M} \rrbracket = (Q, Q_0, \Lambda, \rightarrow)$ where:

- $Q = \mathcal{D}_V$: (infinite) set of states;
- $Q^0 = \{ \vec{\nu} = \langle \vec{\nu}_i, \vec{\nu}_x \rangle \mid \vec{\nu}_i \in \Theta \land \vec{\nu}_x \in \mathcal{D}_V \}$: set of initial states;
- $\Lambda = \{ \langle a, \vec{\pi} \rangle \mid a \in \Sigma \land \vec{\pi} \in \mathcal{D}_{\text{sig}(a)} \}$: set of valued actions partitioned into $\Lambda = \Lambda? \cup \Lambda!$;
- $\rightarrow$: transition relation defined by the rule:

\[
\begin{align*}
(a, \vec{\rho}, G, A) & \in T & \vec{\nu} = \langle \vec{\nu}_i, \vec{\nu}_x \rangle & \in \mathcal{D}_V & \vec{\pi} & \in \mathcal{D}_{\text{sig}(a)} \\
\vec{\nu}' & = \langle \vec{\nu}'_i, \vec{\nu}'_x \rangle & \in \mathcal{D}_V & G(\vec{\nu}, \vec{\pi}) & \vec{\nu}'_i & = A(\vec{\nu}, \vec{\pi}) \\
\hline
\vec{\nu} \xrightarrow{\langle a, \vec{\pi} \rangle} & \vec{\nu}'
\end{align*}
\]
Runs, Traces

**Run:** \( \text{Runs}(\mathcal{M}) \)

\[
\langle pc = \text{Wait}, g = 0, c = 0 \rangle \xrightarrow{T\text{arget}?^{(3)}} \langle \text{Move}, 3, 0 \rangle \xrightarrow{U\text{p}!^{(0)}} \ldots
\]

\[
\langle \text{Move}, 3, 0 \rangle \xrightarrow{U\text{p}!^{(1)}} \langle \text{Move}, 3, 2 \rangle \xrightarrow{U\text{p}!^{(2)}} \langle \text{Move}, 3, 0 \rangle \xrightarrow{S\text{top}!^{(3)}} \langle \text{Wait}, 3, 0 \rangle
\]

**Traces:** \( \text{Traces}(\mathcal{M}) \): projection of runs on valued actions

\[T\text{arget}?^{(3)}.U\text{p}!^{(0)}.U\text{p}!^{(1)}.U\text{p}!^{(2)}.S\text{top}!^{(3)}\]

\[\rightarrow \text{Accepted runs, accepted traces in } F \subseteq Q\]

\[\text{Runs}_F(\mathcal{M}), \text{Traces}_F(\mathcal{M}).\]
Deterministic ioSTS

Restriction to deterministic ioSTS, where an ioSTS $\mathcal{M} = (V, \Theta, \Sigma, T)$ is deterministic if for any action $a \in \Sigma$, and any pair of transitions $t_1 = (a, \vec{p}, G_1, A_1)$ and $t_2 = (a, \vec{p}, G_2, A_2)$ carrying the same action, the conjunction of the guards $G_1 \land G_2$ is unsatisfiable.

Determinization of ioSTS is not always possible.
Deterministic ioSTS form a strict subclass of ioSTS.
→ Determinization heuristic terminates for a subclass of bounded lookahead ioSTS.
Observability for testing

The tester controls / observes:

- Inputs / Outputs
- Quiescence: state \( q \) is quiescent if no output is fireable in \( q \).

Suspension of \( \mathcal{M} = (V, \Theta, \Sigma, T) \):

\[
\Delta(\mathcal{M}) = (V, \Theta, \Sigma^\delta, T_\delta)
\]

where:

- \( \Sigma^\delta = \Sigma^\delta_! \cup \Sigma^? \) with \( \Sigma^\delta_! = \Sigma_! \cup \{\delta\} \),
- \( T_\delta = T \cup \{\langle \delta, G_\delta, Id_V \rangle\} \) with

\[
G_\delta = \neg \left( \bigvee_{(a, \bar{p}, G, A) \in T, a \in \Sigma_!} \exists \bar{\pi} \in \mathcal{D}_{\text{sig}(a)} : G(\bar{\nu}, \bar{\pi}) \right)
\]

Observable behavior for testing: \( STraces(\mathcal{M}) \triangleq Traces(\Delta(\mathcal{M})) \)
Suspension automaton: example

\[
c = g = 0 \rightarrow \text{Wait} \\
\]

\[
c = g \wedge p = c \\
\text{Stop!}(p) \\
\]

\[
0 \leq p \leq h \\
\text{Target?}(p) \\
\]

\[
c < g \wedge p = c \\
\text{Up!}(p) \\
\]

\[
c > g \wedge p = c \\
\text{Down!}(p) \\
\]

\[
c := c + 1 \\
\]

\[
c := c - 1 \\
\]

\[
\delta! \\
\]

\[
\delta! \\
\]

\[
\delta! \\
\]

\[
\delta! \\
\]

\[
\delta! \\
\]

\[
\delta! \\
\]

\[
\delta! \\
\]

\[
\delta! \\
\]
Outline

1. The ioSTS model
2. Conformance testing theory
3. Test selection using approximate analysis
4. Test execution
5. Conclusion and perspectives
## Testing framework

### Specification

Deterministic ioSTS $S = (V^S, \Theta^S, \Sigma, T^S)$, with $\Sigma = \Sigma! \cup \Sigma?$ and $V^S_x = \emptyset$ (only internal variables).

$[S] = S = (Q, Q^0, \Lambda, \rightarrow)$ with $\Lambda = \Lambda! \cup \Lambda\text{?}$.

### Implementation

unknown $\Lambda\text{?}$-complete ioLTS $I = (Q_I, Q^0_I, \Lambda! \cup \Lambda\text{?}, \rightarrow_I)$.

### Test case

ioSTS $TC = (V^{TC}, \Theta^{TC}, \Sigma^{TC}, T^{TC})$, with $\Sigma^{TC}_? = \Sigma!$, $\Sigma^{TC}_! = \Sigma?$ + variable Verdict with $D_{\text{verdict}} = \{\text{none, fail, pass, inconc}\}$
deterministic, $\Sigma^{TC}_?$-complete in all states where Verdict = none.

$[TC] = TC = (Q^{TC}, q^0_{TC}, \Lambda^{TC}, \rightarrow_{TC})$

Fail = (Verdict = fail), Pass = (Verdict = pass), Inconc = (Verdict = inconc)
Conformance relation

**Definition (Tretmans 96)**

\[ I \ ioco \ S \triangleq \forall \sigma \in \text{Straces}(S), \]
\[ \text{Out}(\Delta(I) \text{ after } \sigma) \subseteq \text{Out}(\Delta(S) \text{ after } \sigma) \]

i.e., after a suspension trace of \( S \), outputs (and quiescences) allowed by \( I \) are allowed by \( S \).

**Alternative characterization**

\[ I \ ioco \ S \iff \text{STraces}(I) \cap [\text{STraces}(S) \cdot \Lambda_\delta \setminus \text{STraces}(S)] = \emptyset \]

\( \text{STraces}(S) \cdot \Lambda_\delta \setminus \text{STraces}(S) \): minimal non-conformant traces
Examples

**Specification S**

**Implementation choice**

**Forbidden output**

**Implementation of a partial specification**

**Forbidden quiescence**
Canonical tester

Build an observer that recognizes \( STraces(S) \cdot \Lambda_1^\delta \setminus STraces(S) \)

**Canonical Tester of \( S = (V^S, \Theta^S, \Sigma, T^S) \)**

\[ Can(S) = (V^{Can}, \Theta^{Can}, \Sigma^{Can}, T^{Can}) \] such that:

- \( V^{Can} = V^S \cup \{\text{Verdict}\} \) where \( D_{\text{Verdict}} = \{\text{none, fail}\} \)
- \( \Theta^{Can} = \Theta^S \land \text{Verdict} = \text{none} \)
- \( \Sigma^?_{Can} = \Sigma^?_1 \) and \( \Sigma^!_{Can} = \Sigma^? \) (alphabet is mirrored / \( \Delta(S) \))
- \( T^{Can} = T^{\Delta(S)} + \) transitions defined by the rules:

\[ a \in \Sigma^?_1 = \Sigma^?_{Can} \quad G_a = \bigwedge_{(a, \bar{p}, G, A) \in T^{\Delta(S)}} \neg G \]

\[ [a(\bar{p}) : G_a(\bar{V}, \bar{p}) \text{ ? Verdict}' := \text{fail}] \in T^{Can} \]

\[ Traces_{\text{Fail}}(Can(S)) = STraces(S) \cdot \Lambda_1^\delta \setminus STraces(S) \]
Canonical tester of the lift specification

\[ c = g = 0 \]
\[ 0 \leq p \leq h \]
\[ Target!(p) \]
\[ g := p \]
\[ 0 \leq p \leq h \]
\[ Target!(p) \]
\[ c < g \land p = c \]
\[ Up?(p) \]
\[ c := c + 1 \]
\[ c > g \land p = c \]
\[ Down?(p) \]
\[ c := c - 1 \]
\[ c \neq g \lor p \neq c, Stop?(p) \]
\[ c \geq g \lor p \neq c, Up?(p) \]
\[ c \leq g \lor p \neq c, Down?(p) \]
Modeling test execution

Test execution of $\mathcal{T}C$ on $I$

modelled by the parallel composition of
$\Delta(I)$ and $[\mathcal{T}C] = \mathcal{T}C = (Q^{\mathcal{T}C}, q_0^{\mathcal{T}C}, \Lambda? \cup \Lambda! \cup \{\delta\}):$
$\Delta(I) \parallel \mathcal{T}C = (Q^I \times Q^{\mathcal{T}C}, Q_0^I \times \{q_0^{\mathcal{T}C}\}, \Lambda! \cup \{\delta\} \cup \Lambda?, \rightarrow_{\Delta(I) \parallel \mathcal{T}C})$

where $\rightarrow_{\Delta(I) \parallel \mathcal{T}C}$, is defined by the rule:

$$\alpha \in \Lambda! \cup \{\delta\} \cup \Lambda? \quad q_1 \xrightarrow{\alpha} \Delta(I) \quad q_2 \quad q'_1 \xrightarrow{\alpha} \mathcal{T}C \quad q'_2$$

$$\quad (q_1, q'_1) \xrightarrow{\alpha} \Delta(I) \parallel \mathcal{T}C \quad (q_2, q'_2)$$

$\text{Traces}(\Delta(I) \parallel \mathcal{T}C) = \text{STraces}(I) \cap \text{Traces}(\mathcal{T}C) = \text{STraces}(I) \cap \text{Traces}(\mathcal{T}C).$
Test failure

For $P \in \{\text{Fail, Pass, Inconc}\}$,

$$\text{Traces}_{Q^I \times P}(\Delta(I) \parallel TC) = S\text{Traces}(I) \cap \text{Traces}_P(TC).$$

**Test execution failure**

$$TC \ mayfail \ I \triangleq \text{Traces}_{Q^I \times \text{Fail}}(\Delta(I) \parallel TC) \neq \emptyset$$

$$\iff \ S\text{Traces}(I) \cap \text{Traces}_\text{Fail}(TC) \neq \emptyset$$

Similar definitions for *maypass*, *mayinconc*.

Due to choices of the implementation, a test case may fail, pass and inconc on the same implementation.
Test case properties

### Soundness, Exhaustiveness, Completeness

A set of test cases $TS$ is

- **Sound** $\triangleq \forall I : (I \ ioco\ S \implies \forall TC \in TS : \neg(TC\ mayfail\ I))$, i.e., only non-conformant $I$ may be rejected by a $TC \in TS$.

- **Exhaustive** $\triangleq \forall I : (\neg(I \ ioco\ S) \implies \exists TC \in TS : TC\ mayfail\ I)$, i.e., any non-conformant $I$ may be rejected by a $TC \in TS$.

- **Complete** = Sound and Exhaustive

Using $TC\ mayfail\ I \iff STraces(I) \cap Traces_{\text{Fail}}(TC) \neq \emptyset$:

$I \ ioco\ S \iff STraces(I) \cap Traces_{\text{Fail}}(\text{Can}(S)) = \emptyset$

- $TS$ sound $\iff \bigcup_{TC \in TS} Traces_{\text{Fail}}(TC) \subseteq Traces_{\text{Fail}}(\text{Can}(S))$

- $TS$ exhaustive $\iff \bigcup_{TC \in TS} Traces_{\text{Fail}}(TC) \supseteq Traces_{\text{Fail}}(\text{Can}(S))$
Outline

1. The ioSTS model
2. Conformance testing theory
3. Test selection using approximate analysis
4. Test execution
5. Conclusion and perspectives
Guide test selection by **Test Purpose**: abstract description of behaviors to be tested.

- Test Purpose specified by observer of $\text{Can}(S)$: ioSTS $\mathcal{TP}$.
- Compute the behaviors of $\text{Can}(S)$ accepted by $\mathcal{TP}$.
- Problem similar to computing feasible behaviors to a goal.
- Exact computation is not possible  
  $\Rightarrow$ compute over-approximation.
Selection principle
Test purpose

Test selection is guided by a non-intrusive observer:

**Test Purpose**

deterministic ioSTS $\mathcal{TP} = (V^{TP}, \Theta^{TP}, \Sigma^{\delta}, T^{TP})$ such that:

- $V_{x}^{TP} = V_{i}^{S}$: $\mathcal{TP}$ is allowed to observe the internal state of $S$;
- $V_{i}^{TP} \cap V_{i}^{S} = \emptyset$ with $pc^{TP} \in V_{i}^{TP}$ and $\text{accept} \in D_{pc^{TP}}$. 
  Accept $\triangleq (pc^{TP} = \text{accept})$.
- $\mathcal{TP}$ is complete except in accept:
  $\forall a \in \Sigma^{\delta}, pc^{TP} \neq \text{accept} \Rightarrow \bigvee_{(a, \bar{p}, G, A) \in T^{TP}} G = \text{true}$.

**Note:** most coverage criteria can be described by a set of Test Purposes.
A Test Purpose for the lift-controller

\[ l = 0 \Rightarrow S_1 \xrightarrow{\Sigma \delta \setminus \text{Stop}(p)} S_2 \xrightarrow{\Sigma \delta \setminus \text{Stop}(p)} \text{Accept} \]

\[ l := p \]

\[ 2p = l \land 3p \leq h \Rightarrow \text{Stop}(p) \]

\[ 2p \neq l \lor 3p > h \Rightarrow \text{Stop}(p) \]

\[ \text{Sink} \]

\[ 2p = l \land 3p \leq h \Rightarrow \text{Stop}(p), l := p \]
Synchronous Product

Used to identify accepting runs.

\[ \mathcal{P} = \text{Can}(S) \times \mathcal{TP} = (V^P, \Theta^P, \Sigma^{\text{Can}}, T^P) \text{ where:} \]

- \( V^P = V^P_i \cup V^P_x \), with \( V^P_i = V^{\text{Can}}_i \cup V^{\text{TP}}_i \) and \( V^P_x = \emptyset \);
- \( \Theta^P(\langle \vec{v}^{\text{Can}}, \vec{v}^{\text{TP}} \rangle) = \Theta^{\text{Can}}(\vec{v}^{\text{Can}}) \land \Theta^{\text{TP}}(\vec{v}^{\text{TP}}) \);
- \( T^P \) is defined by the following inference rule:
  \[
  \begin{align*}
  [a(\vec{p}) : G^c(\vec{v}^c, \vec{p}) ? (\vec{v}^c_i)' := A^c(\vec{v}^c, \vec{p})] & \in T^{\text{Can}} \\
  [a(\vec{p}) : G^t(\vec{v}^t, \vec{p}) ? (\vec{v}^t_i)' := A^t(\vec{v}^t, \vec{p})] & \in T^{\text{TP}}
  \end{align*}
  \]
  
  \[
  [a(\vec{p}) : G^c(\vec{v}^c, \vec{p}) \land G^t(\vec{v}^t, \vec{p}) ? (\vec{v}^c_i)' := A^c(\vec{v}^c, \vec{p}), (\vec{v}^t_i)' := A^t(\vec{v}^t, \vec{p})] & \in T^P
  \]

\( \mathcal{P}' \): ioSTS obtained by adding \( \text{Verdict} := \text{pass} \) to transitions with \( pc' := \text{accept} \).
Synchronous product $\text{Can}(S) \times TP$ for the lift-controller

T. Jéron  
VTS: Conformance testing Symbolic model-based test selection
Properties of $\mathcal{P}' = \text{Can}(S) \times T\mathcal{P}$

\[
\text{Traces}(\mathcal{P}') \subseteq \text{Traces}(\text{Can}(S)) \\
\text{Traces}_{\text{Fail}}(\mathcal{P}') = \text{Traces}(\mathcal{P}') \cap \text{Traces}_{\text{Fail}}(\text{Can}(S)).
\]

$\mathcal{P}'$ detects every non-conformance along its traces. It is thus a sound test case.

\[
\text{Traces}_{\text{Pass}}(\mathcal{P}') = \text{Traces}_{\text{Accept}}(\mathcal{P}) \subseteq \text{STraces}(S) \cap \text{Traces}_{\text{Accept}}(T\mathcal{P})
\]

(equality if $T\mathcal{P}$ does not observe variables of $S$).
Over-approximation

Let $\text{pre}(A)(X)(\vec{v}, \vec{p}) = \exists \vec{v}' : X(\vec{v}') \land \vec{v}' = A(\vec{v}, \vec{p}) = X(A(\vec{v}, \vec{p}))$
i.e., precondition of $X$ by an assignment $A$
and $\text{pre}^\alpha(A)(X)(\vec{v}, \vec{p}) \supseteq \text{pre}(A)(X)(\vec{v}, \vec{p})$ an over-apparoximation

Let $\text{coreach}(\text{Pass}) = \text{lfp}(\lambda X. \text{Pass} \cup \text{pre}(X))$
where $\text{pre}(X) = \{ q \mid \exists q' \in X, \exists \alpha \in \Lambda : q \xrightarrow{\alpha} q' \}$ is the set of states from which $X$ can be reached in one transition.

If $\text{coreach}^\alpha$ is an over-approximation of $\text{coreach}(\text{Pass})$, then

- $\text{pre}^\alpha(A)(\text{coreach}^\alpha)$ is a necessary condition to stay in $\text{coreach}(\text{Pass})$
- $\neg \text{pre}^\alpha(A)(\text{coreach}^\alpha)$ is a sufficient condition to leave $\text{coreach}(\text{Pass})$.

Used to reinforce the guards and compute a test case from $\mathcal{P}'$. 
Test selection using approximation
Selected test case

The test case for $S$ and $\mathcal{T}\mathcal{P}$ is $\mathcal{TC} = (V^{P'}, \Theta^{P'}, \Sigma^{Can}, T^{TC})$ where $T^{TC}$ is defined from $\mathcal{P}'$ by the three rules:

- **(Select output):**
  
  \[
  (a, \bar{p}, G, A) \in T^{P'} \quad a \in \Sigma^{Can} \\
  G' = \text{pre}^\alpha(A)(\text{coreach}^\alpha) \\
  (a, \bar{p}, G \land G', A) \in T^{TC}
  \]

- **(Fail):**
  
  \[
  (a, \bar{p}, G, A) \in T^{P'} \quad a \in \Sigma^{Can} \\
  A_{\text{Verdict}} = \text{Verdict'} := \text{fail} \\
  (a, \bar{p}, G, A) \in T^{TC}
  \]

- **(Split):**
  
  \[
  (a, \bar{p}, G, A) \in T^{P'} \quad a \in \Sigma^{Can} \\
  A_{\text{Verdict}} \neq \text{Verdict'} := \text{fail} \\
  G' = \text{pre}^\alpha(A)(\text{coreach}^\alpha) \\
  (a, \bar{p}, G \land G', A), (a, \bar{p}, G \land \lnot G', A') \in T^{TC}
  \]

  where $A'$ is defined by
  \[
  \{ \\
  A'_{\text{Verdict}} = \text{Verdict'} := \text{inconc}, \\
  A'_v = A_v \text{ for } v \neq \text{Verdict},
  \}
  \]
Approximate analysis

\[ c = g = 0 \quad l = 0 \]

\[ WS_1 \quad MS_1 \quad WS_2 \quad MS_2 \quad WSink \]

\[ c < g \land p = c \quad Up?(p) \quad c := c + 1 \]

\[ c = g \land p = c \quad Stop?(p) \quad l := p \]

\[ c > g \land p = c \quad Down?(p) \quad c := c - 1 \]

\[ 0 \leq p \leq h \]

\[ Target!(p) \quad g := p \]

\[ c = g \land p = c \land \neg l \land \neg (2p > h) \quad Stop?(p) \quad \text{verdict} := \text{Pass} \]
Approximate analysis

\[ h \geq 0 \]

\[ 3g \leq 2h \]

\[ 3c \leq 2h \]

\[ 3g \leq h \wedge 2g = l \]

\[ \text{true} \]

\[ c = g = 0 \]

\[ l = 0 \]

\[ c = g = 0 \]

\[ l = 0 \]

\[ WS_1 \]

\[ MS_1 \]

\[ WS_2 \]

\[ MS_2 \]

\[ WS_{\text{Sink}} \]

\[ \delta ? \]

\[ \delta ? \]

\[ \delta ? \]

\[ \delta ? \]

\[ \text{Wait} \]

\[ \text{Accept} \]

\[ \text{false} \]

\[ \text{false} \]

\[ \text{false} \]

\[ \text{verdict} := \text{Pass} \]

\[ \text{Stop}(p) \]

\[ \text{Stop}(p) \]

\[ \text{Stop}(p) \]

\[ \text{Stop}(p) \]

\[ \text{T. Jéron} \]

\[ \text{VTS: Conformance testing Symbolic model-based test selection} \]
Approximate analysis

Approximate analysis

\[ WS_1 \xrightarrow{\delta?} MS_1 \xrightarrow{\delta?} WS_2 \xrightarrow{\delta?} MS_2 \xrightarrow{\delta?} WS_1 \]

\[ c = g = 0 \]
\[ l = 0 \]

\[ 0 \leq p \leq h \]
\[ \land \ 3p \leq 2h \]
\[ Target!(p) \]
\[ g := p \]

\[ c < g \land p = c \]
\[ Up?(p) \]
\[ c := c + 1 \]

\[ c > g \land p = c \]
\[ Down?(p) \]
\[ c := c - 1 \]

\[ \]
Approximate analysis

Reachability Analysis

T. Jéron VTS: Conformance testing Symbolic model-based test selection
Test case for the lift controller

\[ c = g = 0 \quad l = 0 \]

\[ c < g \land p = c \quad \text{Up}(p) \quad c := c + 1 \]

\[ 0 \leq p \leq h \land 3p \leq 2h \quad \text{Target}!(p) \quad g := p \]

\[ c = g \land p = c \land 3p \leq 2h \quad \text{Stop}(p) \quad l := p \]

\[ c = g \land p = c \land 2p = l \land 3p \leq h \quad \text{Target}!(p) \quad g := p \]

\[ c > g \land p = c \land \text{Down}(p) \quad c := c - 1 \]

\[ \text{verdict} := \text{Pass} \]

\[ \text{verdict} := \text{Inconc} \]

T. Jérône | VTS: Conformance testing Symbolic model-based test selection
Properties of selected test cases

It can be shown that the (infinite) set of test cases that can be selected is:

**Sound**: comes from soundness of $Can(S)$.
No fail verdict added by subsequent transformations.

**Exhaustive**: for any non-conformant implementation $I$, choose a minimal non-conformant trace $\sigma!.a$,
choose $!b$ such that $\sigma!.b \in STraces(S)$.
Build $TP$ recognizing $\sigma!.b$.
The selected $TC$ fails on $\sigma!.a$. 
Consequences of over-approximation

For two abstractions $\alpha_1$ and $\alpha_2$
(e.g. $\alpha_1$: control vs $\alpha_2$: polyhedra)
\[
pre_{\alpha_1}(A)(\text{coreach}_{\alpha_1}) \supseteq pre_{\alpha_2}(A)(\text{coreach}_{\alpha_2}) 
\implies Traces(\mathcal{T}C_1) \supseteq Traces(\mathcal{T}C_2)
\]

Less precise approximation $\implies$

- More infeasible traces to Accept
- More fail verdicts (all sound)

Limit cases:
- exact analysis: best guiding to Accept
- no analysis: no guiding to Accept
Outline

1. The ioSTS model
2. Conformance testing theory
3. Test selection using approximate analysis
4. Test execution
5. Conclusion and perspectives
Start from the unique initial state.
In each state $\vec{v}$, repeat until a verdict is set, choose either:

Output: Using constraint solving, choose, $\vec{\pi}$ s.t. $G(\vec{v}, \vec{\pi})$
for $(a, \vec{p}, G, A)$, $a \in \Sigma!$.
If no solution, receive an input or observe quiescence.
Send $a(\vec{\pi})$ to $I$.
Move to state $\vec{v}' := A(\vec{v}, \vec{\pi})$.

Input: Receive $a(\vec{\pi})$ from $I$ (or observe quiescence $\delta$).
For each $(a, \vec{p}, G, A)$, $a \in \Sigma_?$, check $G(\vec{v}, \vec{\pi})$ until one of them is true ($TC$ is input-complete)
Move to state $\vec{v}' := A(\vec{v}, \vec{\pi})$. 

T. Jéron

VTS: Conformance testing Symbolic model-based test selection
The lift-controller example

IUT

fix $h = 10$

Target(4):
- Up(0)
- Up(1)
- Up(2)
- Up(3)
- Stop(4)

Target (2):
- Down(4)
- Down(3)
- Stop($c=2$)

Tester

Solve: $0 \leq p \leq h \land 3p \leq 2h \rightarrow p=4$

Check: $c < g$

Check: $c = g \land l:=4$

Solve: $0 \leq p \leq h \land 3p \leq h \land 2p = 1 \rightarrow p=2$

Check: $c > g$

Check: $c = g \land 2c = 1 \land 3c \leq h$

Pass
The lift-controller example

IUT

fix \( h = 10 \)

Target(4):
- Up(0)
- Up(1)
- Up(2)
- Up(3)
- Up(4)
- Stop(5)

\( \delta \)

Tester

Solve: \( 0 \leq p \leq h \land 3p \leq 2h \rightarrow p=5 \)

Check: \( c < g \)

\[ \vdots \]

Check: \( c = g \quad \text{l:=5} \)

Solve: \( 0 \leq p \leq h \land 3p \leq h \land 2p = 1 \)

\( \rightarrow \) No solution

Observe Quiescence

Inconc

T. Jéron

VTS: Conformance testing Symbolic model-based test selection
Outline

1. The ioSTS model
2. Conformance testing theory
3. Test selection using approximate analysis
4. Test execution
5. Conclusion and perspectives
Conclusion

- Test selection algorithm for infinite state (non-deterministic) models of reactive systems
- Using approximate analysis
- Test execution using constraint solving
- Implemented in STG using Nbac (AI) and Lucky (CS)
- Used for conformance testing but a similar approach can be used to eliminate infeasible paths for white box software testing [Denmat 08].
Perspectives

- Tool improvement: simplification of guards, utility of conditions in guards, improved analysis on other domains.
- Similar approach for infinite state heterogeneous models
  - Timed models + data
  - Recursive programs modeled as pushdown systems: [Constant et al. 07]
- Coverage based selection
  - AI + dynamic partitioning as a basis for coverage criteria
  - More semantic based coverage criteria.