Model-based conformance test generation for timed systems

Thierry JÉRON
Joint work with Nathalie Bertrand, Amélie Stainer, Moez Krichen

INRIA Rennes - Bretagne Atlantique, France
Thierry.Jeron@inria.fr
http://www.irisa.fr/prive/jeron/

August, 2015
Conformance testing of reactive systems

Checking that a black-box implementation (IUT) of a reactive system behaves correctly wrt. its specification S, through test experiments.

- **black box**: unknown code, but known interfaces
- the specification is the reference (oracle)

Application domains

Embedded systems in automotive, aerospace, medical devices, etc
Telecommunication systems, Information systems, Web services, etc
Why (and how) formalizing conformance testing?

**Industrial practice:**
manual design of test suites from informal specifications
⇒ high cost, low quality, difficult maintenance, ...

⇒ automatization of test synthesis from formal specifications can be profit earning

→ **formalizing testing/test generation:** model-based testing
  - formal models for specifications, test cases, implementations,
  - formalize the conformance relation, test execution, verdicts
  - design test generation algorithms
  - ensure properties of test cases
Model-based test generation from timed systems

Motivations

- Testing reactive systems with **timing constraints**
  
  *e.g.* real-time systems.

Timed Automata (TA) [AD94]

- A standard model for RT systems
- Well studied theory
  
  *(e.g. reachability pb decidable using Region/Zone Automata)*
- Verification tools: UPPAAL, Chronos, IF...

Conformance theory for TAs

- TA model adapted for testing: TAIO
- Conformance relation: `tioco` [KT09] / `rtioco` [LMN04]
  
  Extends `ioco` for untimed models (IOLTS) to TAIOs
Challenges for MBT with tioco

Determinization
may be necessary to foresee allowed actions after observable traces.
but not all TAs can be determinized

→ Two approaches to test generation:
  ▶ On-line testing (e.g. UPPAAL-TRON): test gen. during execution;
      Allowed actions after one trace: no determinization.
  ▶ Off-line testing: separate test generation and test execution;
      Most often restricted to deterministic/determinizable classes of TAs.
      Exception: [KT09] based on approximate determinization.

Test selection
not all behaviours can be tested (infinite runs/dense time),
thus it is necessary to select some finite behaviors to test.
Different approaches: random, coverage criteria, test purposes.
Our approach

Off-line test generation from TAIOs in the *tioco* testing theory

- **General model** of non-deterministic TAIOs:
  - input/output/internal actions, invariants (urgency)

- **Off-line** test case generation [BJSK11, BJSK12]
  - Approximate determinization of TAIOs [BSJK11, BSJK15].
  - Selection by expressive test purposes,
  - using symbolic reachability analysis,
  - producing TAIOs test cases.
Outline

1. Timed Automata with inputs and outputs (TAIOs)
2. The tioco testing theory
3. Off-line test case selection
1 Timed Automata with inputs and outputs (TAIOs)

2 The $\text{tioco}$ testing theory

3 Off-line test case selection
Timed automata with inputs and outputs (TAIOs)

Automata + clocks + inputs /outputs/internal to describe testing artifacts (specif., implem., test cases), extended for test purposes.

TAIO $\mathcal{A} = (L, \ell_0, \Sigma?, \Sigma!, \Sigma\tau, X, M, I, E)$.

Guard/invariant: conj. of $x \sim c$, $c \in [0, M] \cap \mathbb{N}$, $\sim \in \{<, \leq, =, \geq, >\}$

Resources $(X, M) = (\{x\}, 2)$, $\rightarrow$ region abstraction, determinization
Semantics of TAIOs: Runs, Traces

- **state** = (location, valuation of X),

- **Runs**: from state to state by discrete trans./time elapse

  \[ \rho_1 = (\ell_0, 0) \xrightarrow{1} (\ell_0, 1) \xrightarrow{\text{x=1, } \tau} (\ell_1, 1) \xrightarrow{5} (\ell_1, 1.5) \xrightarrow{1<x<2, a?, \{x\}} (\ell_2, 0) \]

  \[ \rho_2 = (\ell_0, 0) \xrightarrow{1} (\ell_0, 1) \xrightarrow{\text{x=1, } \tau, \{x\}} (\ell_5, 0) \xrightarrow{5} (\ell_5, 5) \xrightarrow{\text{x<1, a?, \{x\}}} (\ell_6, 0) \]

- **Traces**: \( \sigma_1 = \sigma_2 = (1.5).a? : \) proj. on observ. delays, actions

- **After**: \( \mathcal{A} \text{ after } (1.5).a? = \{(\ell_2, 0), (\ell_6, 0)\} \) (non-determinism)

- **Out**: \( \text{out}(\mathcal{A} \text{ after } (1.5).a?) = \text{out}((\{(\ell_2, 0), (\ell_6, 0)\}) = \{b\} \cup [0, \infty) \)
Some characteristics of TAIOs

A TAIO $\mathcal{A}$ is said

- **deterministic** (DTAIO): no $\tau$ action, no intersecting guards in any $\ell$
  Ensures that $\forall \sigma \in \text{Traces}(\mathcal{A}), \mathcal{A}$ after $\sigma$ is a singleton.

- **complete**: in any location, all delays and actions are enabled
  $\forall \ell \in L, (I(\ell) = \text{true} \land \forall a \in \Sigma, \bigvee (\ell, g, a, x', \ell') \in E \ g = \text{true})$

- **input-complete in state** $(\ell, v)$: ready to receive any input
  $\forall a \in \Sigma^A, (\ell, v) \xrightarrow{a}$.

- **non-blocking**: does not prevent time to progress
  from any reachable state, there is an execution of arbitrary duration.
1 Timed Automata with inputs and outputs (TAIOs)

2 The **tioco** testing theory

3 Off-line test case selection
Conformance testing framework

**Specification**
\[ A \in TAIO \]
\((\ldots, \Sigma^A?, \Sigma^A!, \Sigma^A\tau, X^A, M^A, \ldots)\)

**Implementation**
\[ I \in TAIO \]
input-complete, non-blocking
\((\ldots, \Sigma^I?, \Sigma^I!, \Sigma^I\tau, X^I, M^I, \ldots)\)

**Test purpose**
\[ TP \in OTAIO \]
+ Accept \( \in L^{TP} \)
complete

**Test case**
\[ TC \in DTAIO \]
Verdicts:
- None
- Pass
- Fail
- Inconc
\((\ldots, \Sigma^I!, \Sigma^I?, Y,N,\ldots)\)
input-complete in None

**Test Generation**
The tioco conformance relation [KT09]

Let $\mathcal{A}$ be a TAIO, and $\mathcal{I}$ an input-complete, non-blocking TAIO, $\mathcal{I}$ tioco $\mathcal{A}$ if after traces of $\mathcal{A}$, outputs and delays of $\mathcal{I}$ are allowed by $\mathcal{A}$. Formally, $\forall \sigma \in \text{Traces}(\mathcal{A}), \text{out}(\mathcal{I} \text{ after } \sigma) \subseteq \text{out}(\mathcal{A} \text{ after } \sigma)$.

Alternative def.: $\text{Traces}(\mathcal{I}) \cap [\text{Traces}(\mathcal{A}).(\Sigma_1 \cup \mathbb{R}^+) \setminus \text{Traces}(\mathcal{A})] = \emptyset$.

---

**Diagram:**

<table>
<thead>
<tr>
<th>$\mathcal{A}$</th>
<th>$\mathcal{I}_1$ tioco $\mathcal{A}$</th>
<th>$\neg \mathcal{I}_2$ tioco $\mathcal{A}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>{x}</td>
<td>{x}</td>
<td>{x}</td>
</tr>
<tr>
<td></td>
<td>$?a, {x}$</td>
<td>$?a, {x}$</td>
</tr>
<tr>
<td>$x \leq 8$</td>
<td>$x \leq 5$</td>
<td>$x \leq 5$</td>
</tr>
<tr>
<td></td>
<td>$2 \leq x \leq 8, !b$</td>
<td>$4 \leq x \leq 5, !b$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$1 \leq x \leq 5, !b$</td>
</tr>
</tbody>
</table>

\[
\text{out}(\mathcal{A} \text{ after } ?a.1) = [0, 7] \\
\text{out}(\mathcal{A} \text{ after } ?a.2) = \{b\} \cup [0, 6] \\
\text{out}(\mathcal{I}_2 \text{ after } ?a.1) = \{b, c\} \cup [0, 4] \\
\text{out}(\mathcal{I}_1 \text{ after } ?a.2) = [0, 3] \\
\]
Test purposes

Formalize practice for selecting behaviors of specifications for testing.

A **Test purpose** for $A$ is a pair $(\mathcal{TP}, \text{Accept})$ where

1. $\mathcal{TP} = (L^{TP}, \ell_0^{TP}, \Sigma^A, \Sigma^I, \Sigma^T, X^A, X^{TP}, M^{TP}, I^{TP}, E^{TP})$ is a non-intrusive OTAIO: complete, observing $\Sigma^A$ and $X^A$,
   + proper clocks $X^{TP}$ enhancing precision

2. $\text{Accept} \subseteq L^{TP}$: accepting trap locations.
Test cases

Test case for $\mathcal{A} : (\mathcal{T}C, \textbf{Verdicts})$ where

- $\mathcal{T}C = (L^{\mathcal{T}C}, \ell_0^{\mathcal{T}C}, \Sigma_?^{\mathcal{T}C} = \Sigma_?^A, \Sigma_?^{\mathcal{T}C} = \Sigma_?^A, Y, N, I^{\mathcal{T}C}, E^{\mathcal{T}C})$ is a DTAIO
- **Verdicts**: partition of $S^{\mathcal{T}C} = \text{None} \sqcup \text{Pass} \sqcup \text{Fail} \sqcup \text{Inconc}$
- $\mathcal{T}C$ is input-complete in \text{None} states + $\forall \ell$, $I^{\mathcal{T}C}(\ell) = \text{true}$.

Test suite $\mathcal{T}S = \text{set of test cases}$.

```
\begin{align*}
\ell''_0 \quad & 1 < y < 2, a!, \{y\} \\
\ell''_1 \quad & y = 0, b?, \{y\} \\
\ell''_2 \quad & y = 0, b?, \{y\} \\
\ell''_3 \quad & y = 0, b?, \{y\} \\
\ell''_4 \quad & y = 0, b?, \{y\} \\
\text{Accept}_1 \quad & \\
\text{Accept}_2 \quad & \\
\ell_{\text{Fail}} \quad & y \geq 0, b? \\
\end{align*}
```

**Pass** = $\{\text{Accept}_1, \text{Accept}_2\} \times \mathbb{R}_+$

**Inconc** = $\{\ell''_0\} \times [2, \infty) \cup \{\ell''_1\} \times (0, \infty) \cup \{\ell''_2\} \times (0, 1]$

**Fail** = $\{\ell_{\text{Fail}}\} \times \mathbb{R}_+ \cup \{\ell''_3, \ell''_4\} \times (0, \infty) \cup \{\ell''_2\} \times (1, \infty)$
Test execution and verdicts

Test execution

The execution of \( TC \) on \( I \) is modelled by the parallel composition \( I \parallel TC \) where time and (opposite) observable actions synchronize.

Ensures \( \text{Traces}(I \parallel TC) = \text{Traces}(I) \cap \text{Traces}(TC) \).

Failure by a test case

The (possible) failure of an implementation to pass a test is modelled as

\[
I \text{ fails } TC \equiv \text{Traces}(I) \cap \text{Traces(Fail}(TC)) \neq \emptyset
\]

i.e. the execution of \( I \parallel TC \) may lead \( TC \) to a Fail state.

(similar defs of passes for Pass and inconc for Inconc).

Warning: due to non-controlability, the same \( I \) may produce different verdicts for the same test case.
Expected properties of test suites

- **Soundness:** \( \forall I, \forall TC \in TS, I \text{ fails } TC \Rightarrow \neg(I \text{ tioco } A) \)
  only non-conformant implementations can be rejected by a test case

- **Exhaustiveness:** \( \forall I, \neg(I \text{ tioco } A) \Rightarrow \exists TC \in TS, I \text{ fails } TC \)
  all non-conformant implem. may be rejected by some test case

- **Strictness:** \( \forall I, \forall TC \in TS, \neg(I \parallel TC \text{ tioco } A) \Rightarrow I \text{ fails } TC \)
  non-conformant traces traversed during test execution imply rejection

- **Precision:** A test suite \( TS \) for \( A \) and \( TP \) is *precise* if \( \text{Pass} \) verdicts are delivered for traces of runs of \( A \) accepted by \( TP \).
  \[
  \text{Traces}_{\text{Pass}}(TC) = \text{Traces}(\text{Seq}(A) \uparrow^X TP \cap \text{Seq}_{\text{Accept}}^TP(\mathcal{TP}))
  \]
Let $A, B$ be two TAIOs with same input/output alphabets

$A \text{ io-refines } B$ $(B \text{ io-abstracts } A)$ if

$A \preceq B \equiv \begin{cases} \forall \sigma \in \text{Traces}(B), \quad \text{out}(A \text{ after } \sigma) \subseteq \text{out}(B \text{ after } \sigma) \\ \forall \sigma \in \text{Traces}(A), \quad \text{in}(B \text{ after } \sigma) \subseteq \text{in}(A \text{ after } \sigma). \end{cases}$
io-abstraction and \textit{tioco}

**Proposition: io-abstraction preserves conformance**

If $A \preceq B$ then $\mathcal{I} \text{ tioco } A \Rightarrow \mathcal{I} \text{ tioco } B$.

**Proof sketch:** when $\mathcal{I}$ input-complete, $\mathcal{I} \text{ tioco } A \iff \mathcal{I} \preceq A$

by transitivity: $\mathcal{I} \text{ tioco } A \land A \preceq B \Rightarrow \mathcal{I} \preceq B \iff \mathcal{I} \text{ tioco } B$

**Corollary: io-refinement preserves soundness**

If $A \preceq B$ then $\mathcal{T} \mathcal{S}$ sound for $B \Rightarrow \mathcal{T} \mathcal{S}$ sound for $A$.

**Proof sketch:** $A \preceq B \Rightarrow (\neg(\mathcal{I} \text{ tioco } B) \Rightarrow \neg(\mathcal{I} \text{ tioco } A))$

$\mathcal{T} \mathcal{S}$ sound for $B = (\forall \mathcal{I}, \mathcal{I} \text{ fails } \mathcal{T} \mathcal{C} \Rightarrow \neg(\mathcal{I} \text{ tioco } B))$

$\Rightarrow (\forall \mathcal{I}, \mathcal{I} \text{ fails } \mathcal{T} \mathcal{C} \Rightarrow \neg(\mathcal{I} \text{ tioco } A)) = \mathcal{T} \mathcal{S}$ sound for $A$. 
1. Timed Automata with inputs and outputs (TAIOs)

2. The tioco testing theory

3. Off-line test case selection
Challenges of test generation

Generating a test suite $\mathcal{T}S$ from a TAIO $\mathcal{A}$.

- **Selection** of a finite set of $\mathcal{T}C$ by **test purposes** $\mathcal{T}P$:
  → precision gained by an expressive model of $\mathcal{T}P$: OTAIOs

- **Off-line** test generation:
  - **determinization** required to foresee outputs after any trace of $\mathcal{A}$,
  - but TAs cannot be determinized in general
  → approximate determinization adapted to tioco

- **Desired** **properties** of $\mathcal{T}S$:
  → conditions to ensure soundness ?, exhaustiveness ?, strictness ?
Off-line test case selection with test purposes

\[ \mathcal{A} \in TAIO \]
\[ (L^A, \epsilon_0^A, \Sigma^A_i, \Sigma^A_T, X^A, M^A, I^A, E^A) \]

\[ TP \in OTAIO + \text{Accept} \]
\[ TP \in LP \]
\[ (\ldots, \Sigma^A_i, \Sigma^A_T, X^T_P, M^P, \ldots) \]

\[ \mathcal{P} \in TAIO + \text{Accept} \]
\[ \mathcal{P} \in LP \]
\[ (\ldots, \Sigma^A_i, \Sigma^A_T, X^A \cup X^T_P, M^P, \ldots) \]

\[ \mathcal{D}P \in DTAIO + \text{Accept} \]
\[ \mathcal{D}P \in LP \]
\[ (\ldots, \Sigma^A_i, \Sigma^A_T, Y, N, \ldots) \]

\[ TC \in DTAIO \]
\[ \text{Verdicts:} s^{TC} = \text{None} \sqcup \text{Pass} \sqcup \text{Fail} \sqcup \text{Inconc} \]
\[ (\ldots, \Sigma^A_i, \Sigma^A_T, Y, N, \ldots) \]
Timed Automata with inputs and outputs (TAIOs)

The \textit{tioco} testing theory

Off-line test case selection

\textbf{Product} $\mathcal{P} = \mathcal{A} \times \mathcal{TP}$

Synchronization on actions and observed clocks (conjunction of guards).

\begin{align*}
\mathcal{P} &= \mathcal{A} \times \mathcal{TP} \\
&= \ell_0' \xrightarrow{x \leq 1} \ell_1' \xrightarrow{x = 1, \tau, \{x\}} \ell_2' \xrightarrow{1 < x < 2, a?, \{x\}} \ell_3' \xrightarrow{x = 0, b!} \ell_4' \\
&\xrightarrow{x \leq 1} \ell_6' \xrightarrow{x = 0} \ell_7' \xrightarrow{x \leq 1} \ell_8' \xrightarrow{x = 0} \ell_9' \xrightarrow{\text{Acc}} \Sigma^A
\end{align*}

Non-intrusiveness: $\text{Traces}(\mathcal{P}) = \text{Traces}(\mathcal{A}) \Rightarrow \text{same tioco implementations.}$

Intersection: $\text{Traces}_{\text{Accept}}^\mathcal{P}(\mathcal{P}) = \text{Traces}(\text{Seq}(\mathcal{A}) \uparrow^{X_{\mathcal{TP}}} \cap \text{Seq}_{\text{Accept}}^\mathcal{TP}(\mathcal{TP}))$
Determinization

Determinization is crucial to set \textbf{Fail} verdicts, i.e. detect non-conformant traces in \( \text{Traces}(\mathcal{P}).(\Sigma ! \cup \mathbb{R}^+) \setminus \text{Traces}(\mathcal{P}) \) but TAIOs (like TAs) cannot be determinized in general (some languages of TAIOs cannot be recognized by DTAIOs).

- Restriction to determinizable classes is limited
- Approximate determinization for any TAIO, adapted to \textbf{tioco}:
  - What approximation is allowed?
    - Remember: io-abstraction preserves soundness
  - How to compute an io-abstract determinization of a TAIO?
    - fix ressources (Y,N), simulate X by Y,
    - try to be exact when possible,
    - when necessary, over-approx. outputs/delays, under-approx. inputs
  \[ \text{[BSJK11]}: \text{a game approach to determinization} \]
Approximate determinization: general scheme

\[ \text{TAIO } \mathcal{P}(X, M) \]
\[ \text{Resources } (Y, N) \]
\[ \text{Game } \mathcal{G}_\mathcal{P}(Y, N) \]
\[ \text{Strategy } \pi \]
\[ \text{DTAIO } \mathcal{D}\mathcal{P}(Y, N) \]

- If $\pi$ wins then $\text{Traces}(\mathcal{P}) = \text{Traces}(\mathcal{D}\mathcal{P})$
- Otherwise $\mathcal{P} \preceq \mathcal{D}\mathcal{P}$

**Corollary:** approximate determinization preserves soundness

If a test suite $\mathcal{T}S$ is **sound** for $\mathcal{D}\mathcal{P}$, it is **sound** for $\mathcal{P}$, thus for $\mathcal{A}$. 
Game principles

Finite turn-based safety game between **Spoiler** and **Determinizator**.

- Config. of game = state estimate \((\tau\text{-closure} + \text{subset construction} + \text{clock relations encoding} X \text{ by } Y)\).
- **Spoiler** chooses an action \(a\) and when to fire it (region \(r\) on \(Y\)).
- **Determinizator** chooses clocks \(Y' \subseteq Y\) to reset.
- Avoid unsafe states (possible strict io-abstraction).
Game principles

Finite turn-based safety game between **Spoiler** and **Determinizator**.

- Config. of game = state estimate ($\tau$-closure + subset construction + clock relations encoding $X$ by $Y$).
- **Spoiler** chooses an action $a$ and when to fire it (region $r$ on $Y$).
- **Determinizator** chooses clocks $Y' \subseteq Y$ to reset.
- Avoid unsafe states (possible strict io-abstraction).

**Properties of the game**

- Strategy of Determinizator $\rightarrow$ deterministic io-abstraction.
- **Winning** strategy of Determinizator $\rightarrow$ deterministic equivalent. (with sufficient ressources, winning strategies exist for all known determinizable classes: event-clock, int. reset, non-Zeno TAs).

**Complexity**: doubly exponential in $|X \cup Y|$, exponential in $|L^P|$. 
Game principles

Finite turn-based safety game between **Spoiler** and **Determinizator**.

- Config. of game = state estimate ($\tau$-closure + subset construction + clock relations encoding $X$ by $Y$).
- **Spoiler** chooses an action $a$ and when to fire it (region $r$ on $Y$)
- **Determinizator** chooses clocks $Y' \subseteq Y$ to reset
- Avoid unsafe states (possible strict io-abstraction).

**Properties of the game**

- Strategy of Determinizator $\rightarrow$ deterministic io-abstraction.
- **Winning** strategy of Determinizator $\rightarrow$ deterministic equivalent. (with sufficient resources, winning strategies exist for all known determinizable classes: event-clock, int. reset, non-Zeno TAs).

**Complexity**: doubly exponential in $|X \cup Y|$, exponential in $|L^P|$. 
Game principles

Finite turn-based safety game between **Spoiler** and **Determinizator**.

- Config. of game = state estimate ($\tau$-closure + subset construction + clock relations encoding $X$ by $Y$).
- **Spoiler** chooses an action $a$ and when to fire it (region $r$ on $Y$)
- **Determinizator** chooses clocks $Y' \subseteq Y$ to reset
- Avoid unsafe states (possible strict io-abstraction).

Properties of the game

- Strategy of Determinizator $\rightarrow$ deterministic io-abstraction.
- **Winning** strategy of Determinizator $\rightarrow$ deterministic equivalent. (with sufficient ressources, winning strategies exist for all known determinizable classes: event-clock, int. reset, non-Zeno TAs).

**Complexity**: doubly exponential in $|X \cup Y|$, exponential in $|L^P|$. 
The game $G_P(Y, N)$ built from $P$

$\text{Accept}^D = \{ \ell \in L^D \text{ containing a config. with location in } \text{Accept}^P \}$. 

Exact determinization $\Rightarrow \text{Traces}(D^P) = \text{Traces}(P) \land \text{Traces}_{\text{Accept}^D}(D^P) = \text{Traces}_{\text{Accept}^P}(P)$
From a strategy to a DTAIO $\mathcal{DP}$

For a strategy $\pi$ of the game, build a TAIO $\mathcal{DP}$. 
Generating $\mathcal{T}C$ from $\mathcal{DP}$: principle

Essentially consists in identifying verdicts in $\mathcal{DP}$:

- **Fail**: detect non-conformant traces in $\text{Traces}(\mathcal{DP}).(\Sigma_! \cup \mathbb{R}^+) \setminus \text{Traces}(\mathcal{DP})$, i.e.:
  - unspecified delays = violation of invariants, incorporated in Fail
    Warning: invariants in $\mathcal{DP}$ transfered to guards in $\mathcal{T}C$
  - unspecified outputs by complementation to a new location $\ell_{\text{Fail}}$

- **Pass**: captured by $\text{Accept}^{\mathcal{DP}}$ locations

- **Inconc**: states not co-reachable from Pass. Avoid them when controllable.

+ Inversion of input/output alphabets
Generating $\mathcal{T}C$ from $\mathcal{DP}$: formalization

$\mathcal{T}C = (L^\mathcal{DP} \sqcup \{\ell_{\text{Fail}}\}, \ell_0^\mathcal{DP}, \Sigma^A_1, \Sigma^A_? , Y, N, I^\mathcal{T}C = \text{true}, E^\mathcal{DP}_I \cup E_{\ell_{\text{Fail}}})$ such that:

$\begin{align*}
\text{▶ } E^\mathcal{DP}_I &= \{(\ell, g \land I^\mathcal{DP}(\ell), a, X', \ell') \mid (\ell, g, a, X', \ell') \in E^\mathcal{DP}\} \text{ and } \\
\text{▶ } E_{\ell_{\text{Fail}}} &= \{\ell, \neg \bigvee_{(\ell, g, a, X', \ell') \in E^\mathcal{DP}} g, a, X^\mathcal{T}C_{\ell_{\text{Fail}}} \mid \ell \in L^\mathcal{DP}, a \in \Sigma^A_1\}.
\end{align*}$
Generating $\mathcal{T}C$ from $\mathcal{DP}$: formalization

$$\mathcal{T}C = (L^{\mathcal{DP}} \sqcup \{\ell_{\text{Fail}}\}, \ell_0^{\mathcal{DP}}, \Sigma^A, \Sigma^?, Y, N, I^{\mathcal{T}C} = \text{true}, E^{\mathcal{DP}}_I \cup E_{\ell_{\text{Fail}}})$$ such that:

- $E^{\mathcal{DP}}_I = \{(\ell, g \land I^{\mathcal{DP}}(\ell), a, X', \ell') \mid (\ell, g, a, X', \ell') \in E^{\mathcal{DP}}\}$ and
- $E_{\ell_{\text{Fail}}} = \{(\ell, \neg \bigvee (\ell, g, a, X', \ell') \in E^{\mathcal{DP}} g, a, X_{\mathcal{T}C}^{\ell_{\text{Fail}}}, \ell_{\text{Fail}}) \mid \ell \in L^{\mathcal{DP}}, a \in \Sigma^A\}$.

**Verdicts:**

- $\text{Fail} = \{\ell_{\text{Fail}}\} \times R^Y \cup \bigcup_{\ell \in L^{\mathcal{DP}}} \{\ell\}, \neg I^{\mathcal{DP}}(\ell)$
- $\text{Pass} = \bigcup_{\ell \in \text{Accept}^{\mathcal{DP}}} (\{\ell\} \times I^{\mathcal{DP}}(\ell))$
- $\text{None} = \text{coreach}(\mathcal{DP}, \text{Pass}) \setminus \text{Pass}$
- $\text{Inconc} = S^{\mathcal{DP}} \setminus (\text{Pass} \cup \text{Fail} \cup \text{Inconc})$

$\text{coreach}(\mathcal{DP}, \text{Pass})$ computed symbolically using regions/zones.

**Complexity:** $\mathcal{O}(|L^{\mathcal{DP}}|, |Y|, N)$
Selection of $\mathcal{TC}$

\[ \text{Fail} = \{ \ell_{\text{Fail}} \} \times \mathbb{R}_+ \cup \{ \ell''_3, \ell''_4 \} \times (0, \infty) \cup \{ \ell''_2 \} \times (1, \infty) \]
Selection of $\mathcal{T}C$

$$\ell''_0 \quad \ell''_1 \quad \ell''_2 \quad \ell''_3 \quad \ell''_4$$

Pass = $\{\text{Accept}_1, \text{Accept}_2\} \times \mathbb{R}_+$

Inconc = $\ell''_0 \times [2, \infty) \cup \ell''_1 \times (0, \infty) \cup \ell''_2 \times (0, 1] \cup \ell''_{**} \times \mathbb{R}$

Fail = $\ell_{\text{Fail}} \times \mathbb{R}_+ \cup \ell''_3, \ell''_4 \times (0, \infty) \cup \ell''_2 \times (1, \infty)$

Urgency “preserved” by incorporating the negation of invariants into Fail.
Selection of $\mathcal{T}$

$$
\begin{align*}
\ell_0'' & \rightarrow \ell_1' \quad y = 0, b\,?, \{y\} \\
\ell_0'' & \rightarrow \ell_3' \quad y = 1, a\,!, \{y\} \\
\ell_0'' & \rightarrow \ell_4' \quad y = 0, b\,?, \{y\}
\end{align*}
$$

**Pass** = $\{\text{Accept}_1, \text{Accept}_2\} \times \mathbb{R}_+$

**Inconc** = $\{\ell_0''\} \times [2, \infty) \cup \{\ell_1''\} \times (0, \infty) \cup \{\ell_2''\} \times (0, 1]$

**Fail** = $\{\ell_{\text{Fail}}\} \times \mathbb{R}_+ \cup \{\ell_3'', \ell_4''\} \times (0, \infty) \cup \{\ell_2''\} \times (1, \infty)$

Urgency “preserved” by incorporating the negation of invariants into **Fail**.

**Last “control” step**: avoid **Inconc** states when possible:

- guard intersected with **None** in the source location
  and with **None** $\cup$ **Pass** in the target location for outputs.
Test case properties

**Theorem**

Any generated test case $TC$ is **sound** for $A$.

If $DP$ is **exact** wrt. $P$, $TC$ is **strict** for $A$, and **precise** for $A$ and $TP$.

**Theorem**

If $A$ is **repeatedly observable** (from any state, a future observation) and $DP$ is **exact**, the set of all test cases that can be generated is **exhaustive**.

If $DP$ is not exact: possibly missed **Fail**, unexpected **Pass**.
Conclusion

- off-line test generation algorithm for all (non-deterministic) TAIOs, thanks to approximate determinization,
- precise selection of test cases by test purposes, using symbolic co-reachability analysis
- generated test cases are TAIOs, i.e. complex reactive systems

Other approaches:
- test generation usualy on-line (TorX like algo.)
- off-line test selection often limited to determini(stic/zable) TAs
  - [KT09] less precise, no preservation of urgency,
  - [KCL98], [END01]: less expressive test purposes
  - [DLLN09]: test selection using games (more restrictive).
Some challenges in MBT

- Combine time and data with non-determinism. Approximate determinization?
- Asynchronous testing.
- Modular test generation for composed systems.
- Semantic coverage / structural coverage.
Bibliography


[BJSK12 ], [BSJK15]: journal versions in LMCS 8(4) and FMSD 46(1).


