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
$\Rightarrow$ high cost, low quality, difficult maintenance, ...

$\Rightarrow$ automatization of test synthesis from formal specifications
can be profit earning

$\rightarrow$ *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

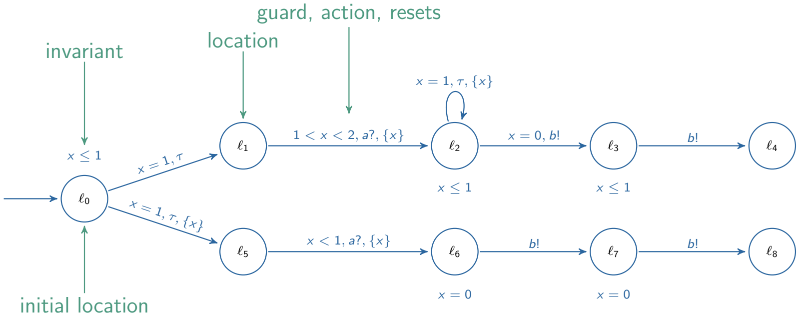
1. Timed Automata with inputs and outputs (TAIOs)
2. The **tioco** testing theory
3. Off-line test case selection

Timed automata with inputs and outputs (TAIOs)

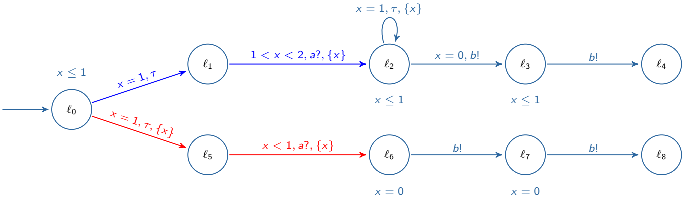
Automata + clocks + inputs /outputs/internal to describe testing artifacts (specif., implem., test cases), extended for test purposes.

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

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

  $\rho_2 = (\ell_0, 0) \xrightarrow{1} (\ell_0, 1) \xrightarrow{(x=1,\tau,\{x\})} (\ell_5, 0) \xrightarrow{.5} (\ell_5, .5) \xrightarrow{(x<1,a?,\{x\})} (\ell_6, 0)$
- **Traces:** $\sigma_1 = \sigma_2 = (1.5).a?$: proj. on observ. delays, actions
- **After**: $\mathcal{A}\ \texttt{after}\,(1.5).a? = \{(\ell_2, 0), (\ell_6, 0)\}$ (non-determinism)
- **Out**: $\texttt{out}(\mathcal{A}\ \texttt{after}\,(1.5).a?) = \texttt{out}(\{(\ell_2, 0), (\ell_6, 0)\}) = \{b\} \cup [0, \infty)$

Some characteristics of TAIOs

A TAIO $\mathcal{A}$ is said

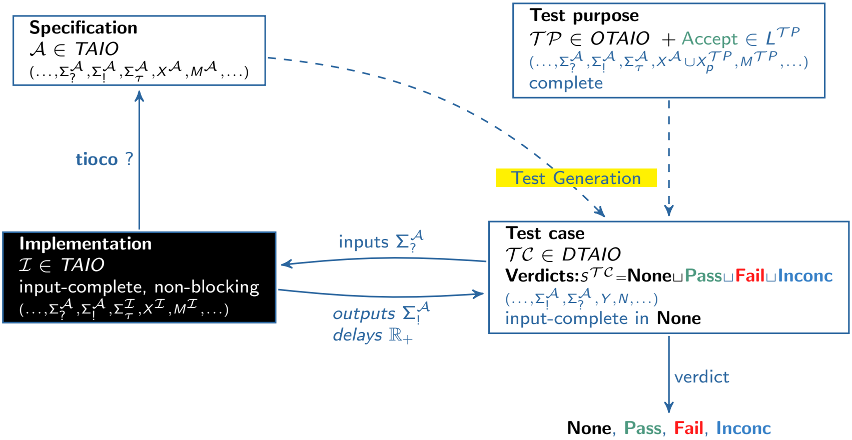
- **deterministic** (DTAIO): no τ action, no intersecting guards in any ℓ
  Ensures that $\forall \sigma \in \text{Traces}(\mathcal{A}), \mathcal{A}\ \texttt{after}\ \sigma$ is a singleton.
- **complete**: in any location, all delays and actions are enabled
  $\forall \ell \in L,\ (I(\ell) = \texttt{true} \wedge \forall a \in \Sigma, \bigvee_{(\ell,g,a,X',\ell') \in E} g = \texttt{true})$
- **input-complete in state** (ℓ, v): ready to receive any input
  $\forall a \in \Sigma_?^{\mathcal{A}},\ (\ell, v) \xrightarrow{a}.$
- **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

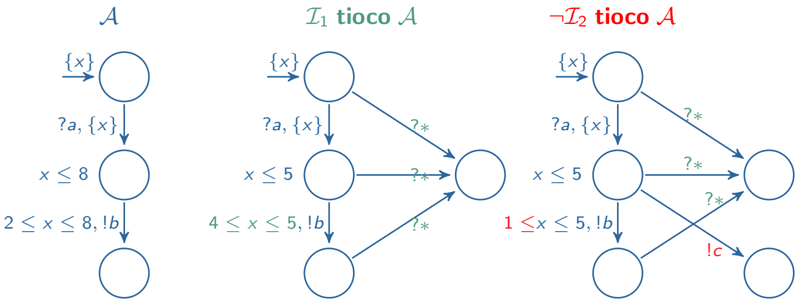
Conformance testing framework



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, $\forall \sigma \in \text{Traces}(\mathcal{A}), \texttt{out}(\mathcal{I} \texttt{ after } \sigma) \subseteq \texttt{out}(\mathcal{A} \texttt{ after } \sigma)$.

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



$\texttt{out}(\mathcal{A} \texttt{ after } ?a.1) = [0,7]$

$\texttt{out}(\mathcal{I}_2 \texttt{ after } ?a.1) = \{b,c\} \cup [0,4]$

$\texttt{out}(\mathcal{A} \texttt{ after } ?a.2) = \{b\} \cup [0,6]$

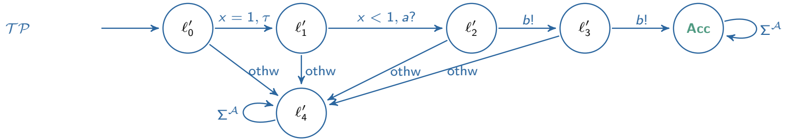
$\texttt{out}(\mathcal{I}_1 \texttt{ after } ?a.2) = [0,3]$

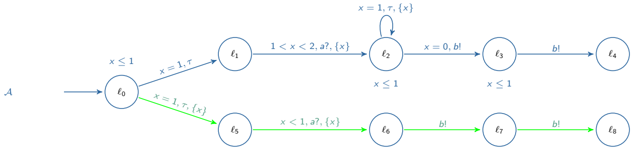
Test purposes

Formalize practice for selecting behaviors of specifications for testing.

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

- $\mathcal{TP} = (L^{\mathcal{TP}}, \ell_0^{\mathcal{TP}}, \Sigma_?^{\mathcal{A}}, \Sigma_!^{\mathcal{A}}, \Sigma_\tau^{\mathcal{A}}, X^{\mathcal{A}}, X^{\mathcal{TP}}, M^{\mathcal{TP}}, I^{\mathcal{TP}}, E^{\mathcal{TP}})$ is a non-intrusive OTAIO: complete, observing $\Sigma^{\mathcal{A}}$ and $X^{\mathcal{A}}$, + proper clocks $X^{\mathcal{TP}}$ enhancing precision
- Accept $\subseteq L^{\mathcal{TP}}$: accepting trap locations.



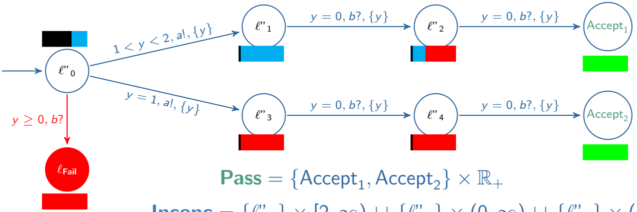


Test cases

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

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

Test suite $\mathcal{TS}$ = set of test cases.



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

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

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

Test execution and verdicts

Test execution

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

Ensures $\boxed{\text{Traces}(\mathcal{I} \| \mathcal{TC}) = \text{Traces}(\mathcal{I}) \cap \text{Traces}(\mathcal{TC}).}$

Failure by a test case

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

$\boxed{\mathcal{I}\ \texttt{fails}\ \mathcal{TC} \equiv \text{Traces}(\mathcal{I}) \cap \text{Traces}_{\textbf{Fail}}(\mathcal{TC}) \neq \emptyset}$

i.e. the execution of $\mathcal{I} \| \mathcal{TC}$ may lead $\mathcal{TC}$ to a **Fail** state.

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

Warning: due to non-controlability, the same $\mathcal{I}$ may produce different verdicts for the same test case.

Expected properties of test suites

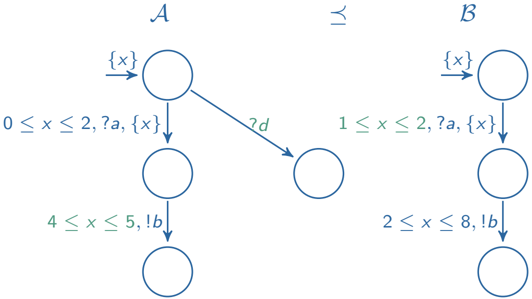
- **Soundness:** $\forall \mathcal{I}, \forall \mathcal{TC} \in \mathcal{TS},\ \mathcal{I}\ \texttt{fails}\ \mathcal{TC} \Rightarrow \neg(\mathcal{I}\ \textbf{tioco}\ \mathcal{A})$
  only non-conformant implementations can be rejected by a test case
- **Exhaustiveness:** $\forall \mathcal{I}, \neg(\mathcal{I}\ \textbf{tioco}\ \mathcal{A}) \Rightarrow \exists \mathcal{TC} \in \mathcal{TS},\ \mathcal{I}\ \texttt{fails}\ \mathcal{TC}$
  all non-conformant implem. may be rejected by some test case
- **Strictness:** $\forall \mathcal{I}, \forall \mathcal{TC} \in \mathcal{TS},\ \neg(\mathcal{I} \| \mathcal{TC}\ \textbf{tioco}\ \mathcal{A}) \Rightarrow \mathcal{I}\ \texttt{fails}\ \mathcal{TC}$
  non-conformant traces traversed during test execution imply rejection
- **Precision:** A test suite $\mathcal{TS}$ for $\mathcal{A}$ and $\mathcal{TP}$ is *precise* if **Pass** verdicts are delivered for traces of runs of $\mathcal{A}$ accepted by $\mathcal{TP}$.
  $\text{Traces}_{\textbf{Pass}}(\mathcal{TC}) = \text{Traces}(\text{Seq}(\mathcal{A}) \uparrow^{X^{\mathcal{TP}}} \cap \text{Seq}_{\text{Accept}^{\mathcal{TP}}}(\mathcal{TP}))$

io-refinement/abstraction

Let $\mathcal{A}$, $\mathcal{B}$ be two TAIOs with same input/output alphabets

$\mathcal{A}$ **io-refines** $\mathcal{B}$ ($\mathcal{B}$ **io-abstracts** $\mathcal{A}$) if
- after traces of $\mathcal{B}$, outputs/delays of $\mathcal{A}$ allowed by $\mathcal{B}$
- after traces of $\mathcal{A}$, inputs of $\mathcal{B}$ allowed by $\mathcal{A}$

$$\mathcal{A} \preceq \mathcal{B} \equiv \begin{cases} \forall \sigma \in \text{Traces}(\mathcal{B}), & \texttt{out}(\mathcal{A}\ \texttt{after}\,\sigma) \subseteq \texttt{out}(\mathcal{B}\ \texttt{after}\,\sigma) \\ \forall \sigma \in \text{Traces}(\mathcal{A}), & \texttt{in}(\mathcal{B}\ \texttt{after}\,\sigma) \subseteq \texttt{in}(\mathcal{A}\ \texttt{after}\,\sigma). \end{cases}$$



io-abstraction and **tioco**

Proposition: io-abstraction preserves conformance

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

Proof sketch: when $\mathcal{I}$ input-complete, $\mathcal{I}$ **tioco** $\mathcal{A} \iff \mathcal{I} \preceq \mathcal{A}$
by transitivity: $\mathcal{I}$ **tioco** $\mathcal{A} \wedge \mathcal{A} \preceq \mathcal{B} \Rightarrow \mathcal{I} \preceq \mathcal{B} \iff \mathcal{I}$ **tioco**$\mathcal{B}$

Corollary: io-refinement preserves soundness

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

Proof sketch: $\mathcal{A} \preceq \mathcal{B} \Rightarrow (\neg(\mathcal{I}$ **tioco** $\mathcal{B}) \Rightarrow \neg(\mathcal{I}$ **tioco** $\mathcal{A}))$
$\mathcal{TS}$ sound for $\mathcal{B} = (\forall \mathcal{I}, \mathcal{I}$ *fails* $\mathcal{TC} \Rightarrow \neg(\mathcal{I}$ **tioco** $\mathcal{B}))$
$\Rightarrow (\forall \mathcal{I}, \mathcal{I}$ *fails* $\mathcal{TC} \Rightarrow \neg(\mathcal{I}$ **tioco** $\mathcal{A})) = \mathcal{TS}$ sound for $\mathcal{A}$.

1. Timed Automata with inputs and outputs (TAIOs)

2. The **tioco** testing theory

3. Off-line test case selection

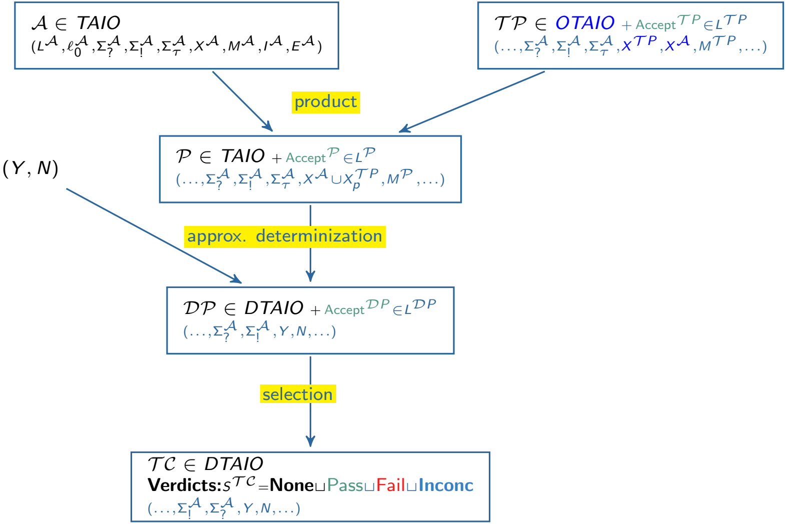
Challenges of test generation

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

- **Selection** of a finite set of $\mathcal{TC}$ by **test purposes** $\mathcal{TP}$:
  $\rightarrow$ precision gained by an expressive model of $\mathcal{TP}$: OTAIOs
- **Off-line** test generation:
  - **determinization** required to foresee outputs after any trace of $\mathcal{A}$,
  - but TAs cannot be determinized in general

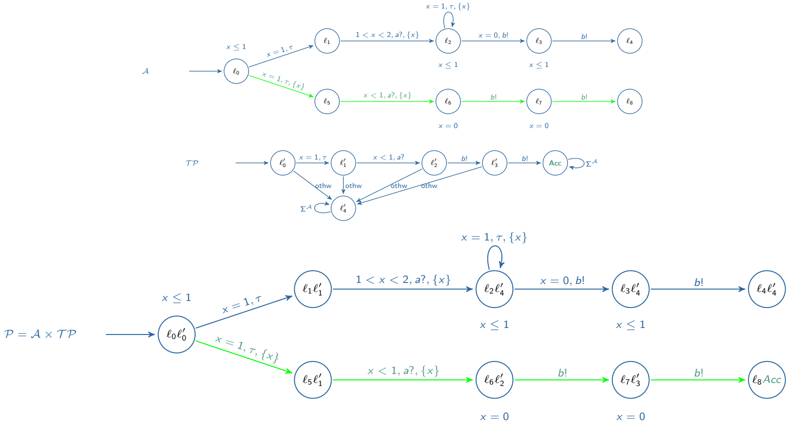
  $\rightarrow$ approximate determinization adapted to **tioco**
- Desired **properties** of $\mathcal{TS}$:
  $\rightarrow$ conditions to ensure soundness ?, exhaustiveness ?, strictness ?

Off-line test case selection with test purposes



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

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

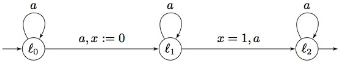


Non-intrusiveness: $\boxed{\text{Traces}(\mathcal{P}) = \text{Traces}(\mathcal{A})}$ $\Rightarrow$ same **tioco** implementations.

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

Determinization

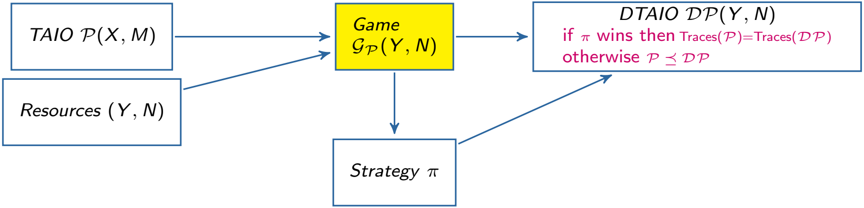
Determinization is crucial to set **Fail** verdicts,
i.e. detect non-conformant traces in $Traces(\mathcal{P}).(\Sigma_! \cup \mathbb{R}^+) \setminus \text{Traces}(\mathcal{P})$
but TAIOs (like TAs) cannot be determinized in general
(some languages of TAIOs cannot be recognized by DTAIOs).



- Restriction to determinizable classes is limited
- Approximate determinization for any TAIO, adapted to **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

  → [BSJK11]: a game approach to determinization

Approximate determinization: general scheme



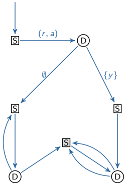
Corollary: approximate determinization preserves soundness

If a test suite $\mathcal{TS}$ is **sound** for $\mathcal{DP}$, it is **sound** for $\mathcal{P}$, thus for $\mathcal{A}$.

Game principles

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

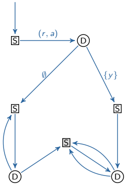
- Config. of game = state estimate (τ-closure + subset construction + clock relations encoding X by Y).
- **Spoiler** chooses an action a and when to fire it (region r on Y)
- **Determinizator** chooses clocks $Y' \subseteq Y$ to reset
- Avoid unsafe states (possible strict io-abstraction).



Game principles

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

- Config. of game = state estimate (τ-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

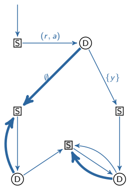
- Strategy of Determinizator → deterministic io-abstraction.
- **Winning** strategy of Determinizator → deterministic equivalent. (with sufficient ressources, winning strategies exist for all known determinizable classes: event-clock, int. reset, non-Zeno TAs).

Complexity: doubly exponential in $|X \cup Y|$, exponential in $|L^{\mathcal{P}}|$.

Game principles

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

- Config. of game = state estimate (τ-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 $|L^{\mathcal{P}}|$.

Game principles

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

- Config. of game = state estimate (τ-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

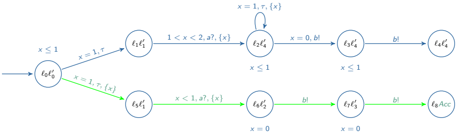
- Strategy of Determinizator $\rightarrow$ deterministic io-abstraction.
- **Winning** strategy of Determinizator $\rightarrow$ deterministic equivalent. (with sufficient ressources, winning strategies exist for all known determinizable classes: event-clock, int. reset, non-Zeno TAs).

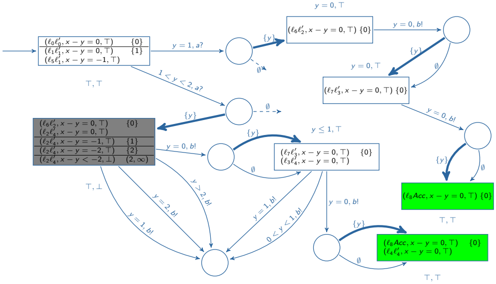
Complexity: doubly exponential in $|X \cup Y|$, exponential in $|L^{\mathcal{P}}|$.

The game $\mathcal{G}_{\mathcal{P}}(Y, N)$ built from $\mathcal{P}$

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

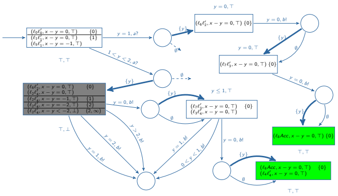
Exact determinization $\Rightarrow$ $\text{Traces}(\mathcal{DP}) = \text{Traces}(\mathcal{P}) \wedge \text{Traces}_{\text{Accept}^{\mathcal{DP}}}(\mathcal{DP}) = \text{Traces}_{\text{Accept}^{\mathcal{P}}}(\mathcal{P})$

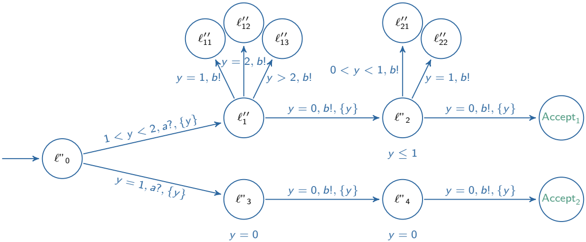




From a strategy to a DTAIO $\mathcal{DP}$

For a strategy π of the game, build a TAIO $\mathcal{DP}$.





Generating $\mathcal{TC}$ from $\mathcal{DP}$: principle

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

- **Fail**: detect non-conformant traces in $Traces(\mathcal{DP}).(\Sigma_! \cup \mathbb{R}^+) \setminus \mathrm{Traces}(\mathcal{DP})$, *i.e.* :
  - **unspecified delays** = violation of invariants, incorporated in **Fail**
    Warning: invariants in $\mathcal{DP}$ transfered to guards in $\mathcal{TC}$
  - **unspecified outputs** by complementation to a new location ℓ_{Fail}
- **Pass**: captured by $\mathrm{Accept}^{\mathcal{DP}}$ locations
- **Inconc**: states not co-reachable from **Pass**.
  Avoid them when controllable.

+ Inversion of input/output alphabets

Generating $\mathcal{TC}$ from $\mathcal{DP}$: formalization

$\mathcal{TC} = (L^{\mathcal{DP}} \sqcup \{\ell_{\mathsf{Fail}}\}, \ell_0^{\mathcal{DP}}, \Sigma_!^{\mathcal{A}}, \Sigma_?^{\mathcal{A}}, Y, N, I^{\mathcal{TC}} = \mathtt{true}, E_I^{\mathcal{DP}} \cup E_{\ell_{\mathsf{Fail}}})$ such that:

- $E_I^{\mathcal{DP}} = \{(\ell, g \wedge I^{\mathcal{DP}}(\ell), a, X', \ell') \mid (\ell, g, a, X', \ell') \in E^{\mathcal{DP}}\}$ and
- $E_{\ell_{\mathsf{Fail}}} = \{(\ell, \neg \bigvee_{(\ell,g,a,X',\ell') \in E^{\mathcal{DP}}} g, a, X_p^{\mathcal{TC}}, \ell_{\mathsf{Fail}}) \mid \ell \in L^{\mathcal{DP}}, a \in \Sigma_!^{\mathcal{A}}\}$.

Generating $\mathcal{TC}$ from $\mathcal{DP}$: formalization

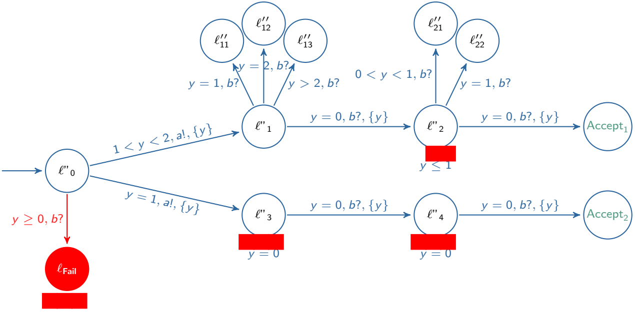
$\mathcal{TC} = (L^{\mathcal{DP}} \sqcup \{\ell_{\mathsf{Fail}}\}, \ell_0^{\mathcal{DP}}, \Sigma_!^{\mathcal{A}}, \Sigma_?^{\mathcal{A}}, Y, N, I^{\mathcal{TC}} = \mathtt{true}, E_I^{\mathcal{DP}} \cup E_{\ell_{\mathsf{Fail}}})$ such that:

- $E_I^{\mathcal{DP}} = \{(\ell, g \wedge I^{\mathcal{DP}}(\ell), a, X', \ell') \mid (\ell, g, a, X', \ell') \in E^{\mathcal{DP}}\}$ and
- $E_{\ell_{\mathsf{Fail}}} = \{(\ell, \neg \bigvee_{(\ell,g,a,X',\ell') \in E^{\mathcal{DP}}} g, a, X_p^{\mathcal{TC}}, \ell_{\mathsf{Fail}}) \mid \ell \in L^{\mathcal{DP}}, a \in \Sigma_!^{\mathcal{A}}\}$.

$$\textbf{Verdicts}: \begin{cases} \textbf{Fail} & = \{\ell_{\mathsf{Fail}}\} \times \mathbb{R}_+^Y \cup \bigcup_{\ell \in \mathcal{L}^{\mathcal{DP}}} (\{\ell\}, \neg I^{\mathcal{DP}}(\ell)) \\ \textbf{Pass} & = \bigcup_{\ell \in \mathsf{Accept}^{\mathcal{DP}}} (\{\ell\} \times I^{\mathcal{DP}}(\ell)) \\ \textbf{None} & = \mathtt{coreach}(\mathcal{DP}, \textbf{Pass}) \setminus \textbf{Pass} \\ \textbf{Inconc} & = S^{\mathcal{DP}} \setminus (\textbf{Pass} \cup \textbf{Fail} \cup \textbf{Inconc}) \end{cases}$$

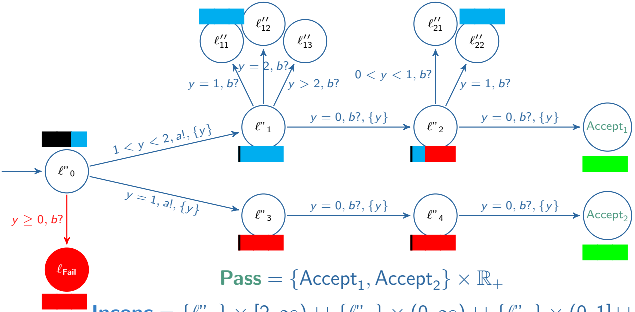
$\mathtt{coreach}(\mathcal{DP}, \textbf{Pass})$ computed symbolically using regions/zones.
Complexity: $\mathcal{O}(|L^{\mathcal{DP}}|.|Y|.N)$

Selection of $\mathcal{TC}$



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

Selection of $\mathcal{TC}$



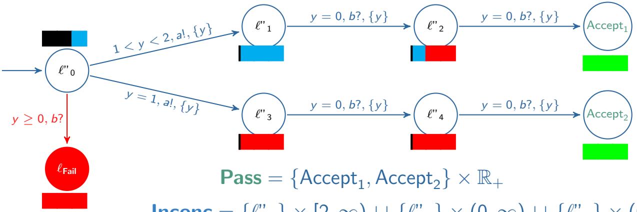
$$\textbf{Pass} = \{\text{Accept}_1, \text{Accept}_2\} \times \mathbb{R}_+$$

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

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

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

Selection of $\mathcal{TC}$



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

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

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

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 $\mathcal{TC}$ is **sound** for $\mathcal{A}$.
If $\mathcal{DP}$ is **exact** wrt. $\mathcal{P}$, $\mathcal{TC}$ is **strict** for $\mathcal{A}$, and **precise** for $\mathcal{A}$ and $\mathcal{TP}$.

Theorem

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

If $\mathcal{DP}$ 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 ?
- Recursion. Pushdown automata. Determinization issue.
- Asynchronous testing.
- Modular test generation for composed systems.
- Semantic coverage / structural coverage.

Bibliography

[BJSK11] N. Bertrand, T. Jéron, A. Stainer, M. Krichen. Off-line Test Selection with Test Purposes for Non-Deterministic Timed Automata. In TACAS'2011.

[BSJK11] N. Bertrand, A. Stainer, T. Jéron, M. Krichen. A game approach to determinize timed automata. In FOSSACS'11.

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

[KT09] M. Krichen and S. Tripakis. Conformance testing for real-time systems. Formal Methods in System Design, 34(3):238-304, 2009.

[LMN04] K. G. Larsen, M. Mikucionis, B. Nielsen. Online testing for real-time systems using Uppaal. In FATES'04.

[KCL98] O. Koné, R. Castanet, and P. Laurencot. On the fly test generation for real time protocols. In ICCCN 1998.

[END03] A. En-Nouaary and R. Dssouli. A guided method for testing timed input output automata. In TestCom 2003.

[DLLN09] A. David, K. G. Larsen, S. Li, and B. Nielsen. Timed testing under partial observability. In ICST 2009.