A fresh look at testing for synchronous communication

Paul Gastin

LSV
ENS de Cachan & CNRS
Paul.Gastin@lsv.ens-cachan.fr

Joint work with Puneet Bhateja and Madhavan Mukund

LAA, June 28th, 2006

Outline

Introduction

Verification of software or hardware

- Proof
- Model checking
- Test

Synchronous testing

- The tester interacts synchronously with the system.
- The tester proposes an action which is either refused or accepted and executed by the system.
- The tester has an immediate feedback.

Asynchronous testing

- The tester communicate asynchronously with the system
- The tester provides inputs and observes outputs.
- The tester does not necessarily know whether its inputs have been used by the system or not.

Introduction

Verification of software or hardware

- Proof
- Model checking
- Test

Synchronous testing

- The tester interacts synchronously with the system.
- The tester proposes an action which is either refused or accepted and executed by the system.
- The tester has an immediate feedback.

Asynchronous testing

- The tester communicate asynchronously with the system
- The tester provides inputs and observes outputs.
- The tester does not necessarily know whether its inputs have been used by the system or not.

Introduction

Verification of software or hardware

- Proof
- Model checking
- Test

Synchronous testing

- The tester interacts synchronously with the system.
- The tester proposes an action which is either refused or accepted and executed by the system.
- The tester has an immediate feedback.

Asynchronous testing

- The tester communicate asynchronously with the system
- The tester provides inputs and observes outputs.
- The tester does not necessarily know whether its inputs have been used by the system or not.

Introduction

Static test generation – Input/Output semantics

- Tests are computed in advance and are sent as a whole stream to the system
- The tester then observes the output streams generated by the system

on the fly test generation – IO-Blocks semantics

- Inputs are supplied incrementally.
- The tester observes the outputs that are triggered by each block of input.

Test equivalence

- Equivalence of two systems for a given test semantics.
- We study the expressiveness and the decidability of some test equivalences.

Introduction

Static test generation – Input/Output semantics

- Tests are computed in advance and are sent as a whole stream to the system
- The tester then observes the output streams generated by the system

on the fly test generation – IO-Blocks semantics

- Inputs are supplied incrementally.
- The tester observes the outputs that are triggered by each block of input.

Test equivalence

- Equivalence of two systems for a given test semantics.
- We study the expressiveness and the decidability of some test equivalences.

Introduction

Static test generation – Input/Output semantics

- Tests are computed in advance and are sent as a whole stream to the system
- The tester then observes the output streams generated by the system

on the fly test generation – IO-Blocks semantics

- Inputs are supplied incrementally.
- The tester observes the outputs that are triggered by each block of input.

Test equivalence

- Equivalence of two systems for a given test semantics.
- We study the expressiveness and the decidability of some test equivalences.

Related work

- I. Castellani and M Hennessy: Testing Theories for Asynchronous Languages, *Proc. FSTTCS '98*, Springer Lecture Notes in Computer Science **1530** (1998) 90–101.
- R. de Nicola and M. Hennessy: Testing equivalences for processes, *Theoretical Computer Science*, **34** (1984) 83–133.
- A. Petrenko, N. Yevtushenko and J.L. Huo: Testing Transition Systems with Input and Output Testers, *Proc IFIP TC6/WG6.1 XV International Conference on Testing of Communicating Systems (TestCom 2003)*, Sophia Antipolis, France, (2003) 129–145.
- J. Tretmans: Test Generation with Inputs, Outputs and Repetitive Quiescence, *Software—Concepts and Tools*, **17**(3) (1996) 103–120.

Outline

Introduction

2 Input/Output semantics

IO-Blocks semantics

Queue semantics (Tretman)

Conclusion

The model

Labelled transition system

$TS = (S, \Sigma, I, T)$ where

- S is the set of states
- $I \subseteq S$ is the set of initial states
- $\Sigma = \Sigma_i \uplus \Sigma_o$ is the set of input/output actions
- $T \subseteq S \times \Sigma \times S$ is the set of transitions

$L(TS) = \{w \in \Sigma^* \mid I \xrightarrow{w} \text{ in } TS\}$.

$s \in S$ is deadlocked if it refuses all output actions: $s \overset{\Sigma_o}{\not\rightarrow}$.

Some further properties

- No infinite output only behaviour.
- Receptivness: $\forall s \in S$, $\forall a \in \Sigma_i$, $s \xrightarrow{a}$
  If this is not the case, we may
  - discard unexpected inputs
  - enter a dead state accepting all inputs and with no possible outputs.

The model

Labelled transition system

$TS = (S, \Sigma, I, T)$ where

- S is the set of states
- $I \subseteq S$ is the set of initial states
- $\Sigma = \Sigma_i \uplus \Sigma_o$ is the set of input/output actions
- $T \subseteq S \times \Sigma \times S$ is the set of transitions

$L(TS) = \{w \in \Sigma^* \mid I \xrightarrow{w} \text{ in } TS\}$.

$s \in S$ is deadlocked if it refuses all output actions: $s \stackrel{\Sigma_o}{\not\rightarrow}$.

Some further properties

- No infinite output only behaviour.
- Receptivness: $\forall s \in S$, $\forall a \in \Sigma_i$, $s \xrightarrow{a}$

  If this is not the case, we may
  - discard unexpected inputs
  - enter a dead state accepting all inputs and with no possible outputs.

The model

Labelled transition system

$TS = (S, \Sigma, I, T)$ where

- S is the set of states
- $I \subseteq S$ is the set of initial states
- $\Sigma = \Sigma_i \uplus \Sigma_o$ is the set of input/output actions
- $T \subseteq S \times \Sigma \times S$ is the set of transitions

$L(TS) = \{w \in \Sigma^* \mid I \xrightarrow{w} \text{ in } TS\}$.

$s \in S$ is deadlocked if it refuses all output actions: $s \stackrel{\Sigma_o}{\not\to}$.

Some further properties

- No infinite output only behaviour.
- Receptivness: $\forall s \in S$, $\forall a \in \Sigma_i$, $s \xrightarrow{a}$
  If this is not the case, we may
  - discard unexpected inputs
  - enter a dead state accepting all inputs and with no possible outputs.

Asynchronous IO-Behaviours

Intuition: Provide some test input $u \in \Sigma_i^*$ up front and observe the maximal outcome $v \in \Sigma_o^*$. Corresponds to static test generation.

Definition: IO-Behaviours

Let $TS = (S, \Sigma, I, T)$. $IOBeh(TS)$ is the set of pairs $(u, v) \in \Sigma_i^* \times \Sigma_o^*$ such that there is a (maximal) run $i \xrightarrow{w} s$ in TS with

- $i \in I$ and s deadlocked
- $\pi_o(w) = v$, and
- either $\pi_i(w) = u$ or there exists $a \in \Sigma_i$ such that $\pi_i(w)a \preceq u$ and $s \overset{a}{\not\rightarrow}$.

Example

$IOBeh(TS_1)$:
$(\varepsilon, \varepsilon)$
(a, x), (a, xy)
(a^2, x), (a^2, xy), (a^2, x^2)
(a^n, x), (a^n, xy), (a^n, x^2) if $n \geq 2$.

Asynchronous IO-Behaviours

Intuition: Provide some test input $u \in \Sigma_i^*$ up front and observe the maximal outcome $v \in \Sigma_o^*$. Corresponds to static test generation.

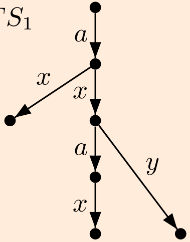
Definition: IO-Behaviours

Let $TS = (S, \Sigma, I, T)$. $IOBeh(TS)$ is the set of pairs $(u, v) \in \Sigma_i^* \times \Sigma_o^*$ such that there is a (maximal) run $i \xrightarrow{w} s$ in TS with

- $i \in I$ and s deadlocked
- $\pi_o(w) = v$, and
- either $\pi_i(w) = u$ or there exists $a \in \Sigma_i$ such that $\pi_i(w)a \preceq u$ and $s \stackrel{a}{\not\to}$.

Example

TS_1



$IOBeh(TS_1)$:
$(\varepsilon, \varepsilon)$
(a, x), (a, xy)
(a^2, x), (a^2, xy), (a^2, x^2)
(a^n, x), (a^n, xy), (a^n, x^2) if $n \geq 2$.

Asynchronous IO-Behaviours

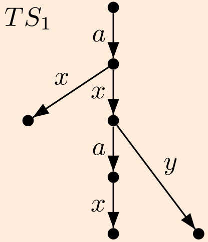
Intuition: Provide some test input $u \in \Sigma_i^*$ up front and observe the maximal outcome $v \in \Sigma_o^*$. Corresponds to static test generation.

Definition: IO-Behaviours

Let $TS = (S, \Sigma, I, T)$. $IOBeh(TS)$ is the set of pairs $(u, v) \in \Sigma_i^* \times \Sigma_o^*$ such that there is a (maximal) run $i \xrightarrow{w} s$ in TS with

- $i \in I$ and s deadlocked
- $\pi_o(w) = v$, and
- either $\pi_i(w) = u$ or there exists $a \in \Sigma_i$ such that $\pi_i(w)a \preceq u$ and $s \stackrel{a}{\not\to}$.

Example



$IOBeh(TS_1)$:
$(\varepsilon, \varepsilon)$
(a, x), (a, xy)
(a^2, x), (a^2, xy), (a^2, x^2)
(a^n, x), (a^n, xy), (a^n, x^2) if $n \geq 2$.

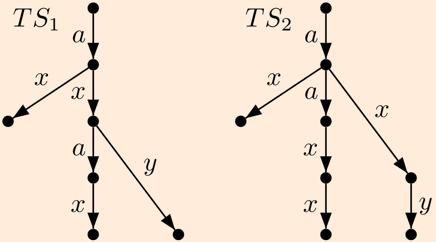
Asynchronous testing equivalence (1)

IO-equivalence

Two transition systems TS and TS' are IO-equivalent, denoted $TS \sim_{io} TS'$ if

$$IOBeh(TS) = IOBeh(TS')$$

Example



$IOBeh(TS_1) = IOBeh(TS_2)$:
$(\varepsilon, \varepsilon)$
(a, x), (a, xy)
(a^n, x), (a^n, xy), (a^n, x^2) if $n \geq 2$.

TS_1 and TS_2 are IO-equivalent.

IO-equivalence corresponds to the queued quiescent trace equivalence of

A. Petrenko, N. Yevtushenko and J.L. Huo: Testing Transition Systems with Input and Output Testers, *Proc of TestCom 2003*.

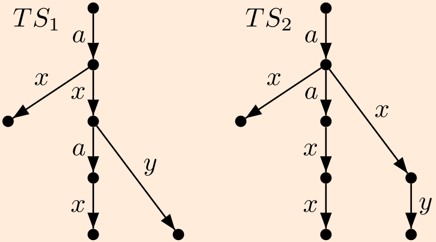
Asynchronous testing equivalence (1)

IO-equivalence

Two transition systems TS and TS' are IO-equivalent, denoted $TS \sim_{io} TS'$ if

$$IOBeh(TS) = IOBeh(TS')$$

Example



$IOBeh(TS_1) = IOBeh(TS_2)$:
$(\varepsilon, \varepsilon)$
(a, x), (a, xy)
(a^n, x), (a^n, xy), (a^n, x^2) if $n \geq 2$.

TS_1 and TS_2 are IO-equivalent.

IO-equivalence corresponds to the queued quiescent trace equivalence of

A. Petrenko, N. Yevtushenko and J.L. Huo: Testing Transition Systems with Input and Output Testers, *Proc of TestCom 2003*.

Rational relations

Definition

Let A, B be two finite (and disjoint) alphabets.
A rational relation over A and B is a rational subset R of the monoid $A^* \times B^*$.

Equivalently, $R \subseteq A^* \times B^*$ is a rational relation if there exists an automaton $\mathcal{A} = (S, A \cup B, I, F, T)$ such that

$$R = \{(u, v) \in A^* \times B^* \mid \exists i \xrightarrow{w} f \text{ in } \mathcal{A} \text{ with } i \in I, f \in F, \pi_A(w) = u, \pi_B(w) = v\}$$

Example

$$\mathcal{R}(\mathcal{A}) = \{(a, x), (a, xy), (a^2, x^2)\}$$

Rational relations

Definition

Let A, B be two finite (and disjoint) alphabets.
A rational relation over A and B is a rational subset R of the monoid $A^* \times B^*$.

Equivalently, $R \subseteq A^* \times B^*$ is a rational relation if there exists an automaton $\mathcal{A} = (S, A \cup B, I, F, T)$ such that

$$R = \{(u, v) \in A^* \times B^* \mid \exists i \xrightarrow{w} f \text{ in } \mathcal{A} \text{ with } i \in I, f \in F, \pi_A(w) = u, \pi_B(w) = v\}$$

Example

$$\mathcal{R}(\mathcal{A}) = \{(a, x), (a, xy), (a^2, x^2)\}$$

Rational relations

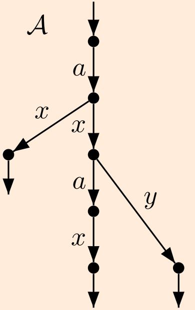
Definition

Let A, B be two finite (and disjoint) alphabets.
A rational relation over A and B is a rational subset R of the monoid $A^* \times B^*$.

Equivalently, $R \subseteq A^* \times B^*$ is a rational relation if there exists an automaton $\mathcal{A} = (S, A \cup B, I, F, T)$ such that

$$R = \{(u,v) \in A^* \times B^* \mid \exists i \xrightarrow{w} f \text{ in } \mathcal{A} \text{ with } i \in I, f \in F, \pi_A(w) = u, \pi_B(w) = v\}$$

Example



$$\mathcal{R}(\mathcal{A}) = \{(a,x), (a,xy), (a^2,x^2)\}$$

From IO-behaviours to rational relations

Proposition

From a transition system $TS = (S, \Sigma, I, T)$, we can construct an automaton $\mathcal{A}$ over $\Sigma = \Sigma_i \uplus \Sigma_o$ such that

$$IOBeh(TS) = \mathcal{R}(\mathcal{A})$$

Proof. Intuition: transform deadlocked states into final states

Let $D \subseteq S$ be the set of deadlocked states of TS. Define $\mathcal{A} = (S', \Sigma, I', F', T')$

- $S' = S \uplus \overline{D} \uplus \{f\}$ where $\overline{D}$ is a copy of D.
- $I' = I \uplus \overline{I \cap D}$ and $F' = \overline{D} \uplus \{f\}$
- $$\begin{aligned} T' = \ T \ & \cup \ \{(r, a, \bar{s}) \mid (r, a, s) \in T \text{ and } s \in D\} \\ & \cup \ \{(\bar{s}, a, f) \mid a \in \Sigma_i \text{ and } s \overset{a}{\nrightarrow}\} \\ & \cup \ \{(f, a, f) \mid a \in \Sigma_i\} \end{aligned}$$

Let $(u, v) \in IOBeh(TS)$ and $i \xrightarrow{w} s$ in TS with $i \in I$, $s \in D$, $\pi_o(w) = v$ and $u = \pi_i(w)au'$ with $s \overset{a}{\nrightarrow}$.

Then, $i \xrightarrow{w} \bar{s} \xrightarrow{a} f \xrightarrow{u'} f$ in $\mathcal{A}$ and $u = \pi_i(wau')$, $w = \pi_o(wau')$.

Hence, $(u, v) \in \mathcal{R}(\mathcal{A})$.

Other cases are similar.

From IO-behaviours to rational relations

Proposition

From a transition system $TS = (S, \Sigma, I, T)$, we can construct an automaton $\mathcal{A}$ over $\Sigma = \Sigma_i \uplus \Sigma_o$ such that

$$IOBeh(TS) = \mathcal{R}(\mathcal{A})$$

Proof. Intuition: transform deadlocked states into final states

Let $D \subseteq S$ be the set of deadlocked states of TS. Define $\mathcal{A} = (S', \Sigma, I', F', T')$

- $S' = S \uplus \overline{D} \uplus \{f\}$ where $\overline{D}$ is a copy of D.
- $I' = I \uplus \overline{I \cap D}$ and $F' = \overline{D} \uplus \{f\}$
- $$\begin{aligned} T' = T \quad & \cup \quad \{(r, a, \bar{s}) \mid (r, a, s) \in T \text{ and } s \in D\} \\ & \cup \quad \{(\bar{s}, a, f) \mid a \in \Sigma_i \text{ and } s \stackrel{a}{\nrightarrow}\} \\ & \cup \quad \{(f, a, f) \mid a \in \Sigma_i\} \end{aligned}$$

Let $(u, v) \in IOBeh(TS)$ and $i \xrightarrow{w} s$ in TS with $i \in I$, $s \in D$, $\pi_o(w) = v$ and $u = \pi_i(w)au'$ with $s \stackrel{a}{\nrightarrow}$.

Then, $i \xrightarrow{w} \bar{s} \xrightarrow{a} f \xrightarrow{u'} f$ in $\mathcal{A}$ and $u = \pi_i(wau')$, $w = \pi_o(wau')$.

Hence, $(u, v) \in \mathcal{R}(\mathcal{A})$.

Other cases are similar.

From IO-behaviours to rational relations

Proposition

From a transition system $TS = (S, \Sigma, I, T)$, we can construct an automaton $\mathcal{A}$ over $\Sigma = \Sigma_i \uplus \Sigma_o$ such that

$$IOBeh(TS) = \mathcal{R}(\mathcal{A})$$

Proof. Intuition: transform deadlocked states into final states

Let $D \subseteq S$ be the set of deadlocked states of TS. Define $\mathcal{A} = (S', \Sigma, I', F', T')$

- $S' = S \uplus \overline{D} \uplus \{f\}$ where $\overline{D}$ is a copy of D.
- $I' = I \uplus \overline{I \cap D}$ and $F' = \overline{D} \uplus \{f\}$
- $$\begin{aligned} T' = \ T \quad & \cup \quad \{(r, a, \bar{s}) \mid (r, a, s) \in T \text{ and } s \in D\} \\ & \cup \quad \{(\bar{s}, a, f) \mid a \in \Sigma_i \text{ and } s \stackrel{a}{\not\to}\} \\ & \cup \quad \{(f, a, f) \mid a \in \Sigma_i\} \end{aligned}$$

Let $(u, v) \in IOBeh(TS)$ and $i \xrightarrow{w} s$ in TS with $i \in I$, $s \in D$, $\pi_o(w) = v$ and $u = \pi_i(w)au'$ with $s \stackrel{a}{\not\to}$.

Then, $i \xrightarrow{w} \bar{s} \xrightarrow{a} f \xrightarrow{u'} f$ in $\mathcal{A}$ and $u = \pi_i(wau')$, $w = \pi_o(wau')$.
Hence, $(u, v) \in \mathcal{R}(\mathcal{A})$.

Other cases are similar.

From IO-behaviours to rational relations

Proposition

From a transition system $TS = (S, \Sigma, I, T)$, we can construct an automaton $\mathcal{A}$ over $\Sigma = \Sigma_i \uplus \Sigma_o$ such that

$$IOBeh(TS) = \mathcal{R}(\mathcal{A})$$

Proof. Intuition: transform deadlocked states into final states

Let $D \subseteq S$ be the set of deadlocked states of TS. Define $\mathcal{A} = (S', \Sigma, I', F', T')$

- $S' = S \uplus \overline{D} \uplus \{f\}$ where $\overline{D}$ is a copy of D.
- $I' = I \uplus \overline{I \cap D}$ and $F' = \overline{D} \uplus \{f\}$
- $$\begin{aligned} T' = T \quad & \cup \quad \{(r, a, \bar{s}) \mid (r, a, s) \in T \text{ and } s \in D\} \\ & \cup \quad \{(\bar{s}, a, f) \mid a \in \Sigma_i \text{ and } s \overset{a}{\not\rightarrow}\} \\ & \cup \quad \{(f, a, f) \mid a \in \Sigma_i\} \end{aligned}$$

Let $(u, v) \in IOBeh(TS)$ and $i \xrightarrow{w} s$ in TS with $i \in I$, $s \in D$, $\pi_o(w) = v$ and $u = \pi_i(w)au'$ with $s \overset{a}{\not\rightarrow}$.

Then, $i \xrightarrow{w} \bar{s} \xrightarrow{a} f \xrightarrow{u'} f$ in $\mathcal{A}$ and $u = \pi_i(wau')$, $w = \pi_o(wau')$.
Hence, $(u, v) \in \mathcal{R}(\mathcal{A})$.
Other cases are similar.

Decidability of IO-equivalence

Theorem

If $|A| = |B| = 1$ then equivalence of rational relations over A and B is decidable.

Corollary

If $|\Sigma_i| = |\Sigma_o| = 1$ then IO-equivalence is decidable.

Decidability of IO-equivalence

Theorem

If $|A| = |B| = 1$ then equivalence of rational relations over A and B is decidable.

Corollary

If $|\Sigma_i| = |\Sigma_o| = 1$ then IO-equivalence is decidable.

From rational relations to IO-behaviours

Several problems:

- Final states may not be deadlocked (easy to fix).
- Deadlocked states may not be final (harder to fix).

Example

Same rational relation: $\mathcal{R}(\mathcal{A}_1) = \{(a^2, x^3)\} = \mathcal{R}(\mathcal{A}_2)$

But different IO-behaviours:

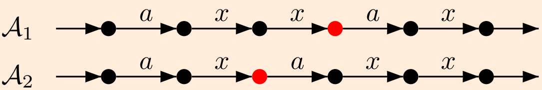
$$\begin{aligned} IOBeh(\mathcal{A}_1) &= \{(\varepsilon,\varepsilon),(a,x^2)\} \cup \{(a^n,x^3) \mid n \geq 2\} \\ IOBeh(\mathcal{A}_2) &= \{(\varepsilon,\varepsilon),(a,x)\} \cup \{(a^n,x^3) \mid n \geq 2\} \end{aligned}$$

From rational relations to IO-behaviours

Several problems:

- Final states may not be deadlocked (easy to fix).
- Deadlocked states may not be final (harder to fix).

Example



Same rational relation: $\mathcal{R}(\mathcal{A}_1) = \{(a^2, x^3)\} = \mathcal{R}(\mathcal{A}_2)$

But different IO-behaviours:

$$\begin{aligned} IOBeh(\mathcal{A}_1) &= \{(\varepsilon,\varepsilon),(a,x^2)\} \cup \{(a^n,x^3) \mid n \geq 2\} \\ IOBeh(\mathcal{A}_2) &= \{(\varepsilon,\varepsilon),(a,x)\} \cup \{(a^n,x^3) \mid n \geq 2\} \end{aligned}$$

From rational relations to IO-behaviours

Several problems:

- Final states may not be deadlocked (easy to fix).
- Deadlocked states may not be final (harder to fix).
- Discarded inputs should be taken care of.

Example

Same IO-behaviours: $IOBeh(\mathcal{A}_1) = \{(\varepsilon, \varepsilon)\} \cup \{(a^n, x) \mid n \geq 1\} = IOBeh(\mathcal{A}_2)$

But different rational relations:

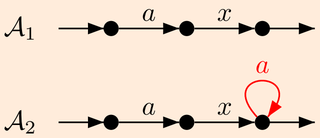
$$\begin{aligned} \mathcal{R}(\mathcal{A}_1) &= \{(a, x)\} \\ \mathcal{R}(\mathcal{A}_2) &= \{(a^n, x) \mid n \geq 1\} \end{aligned}$$

From rational relations to IO-behaviours

Several problems:

- Final states may not be deadlocked (easy to fix).
- Deadlocked states may not be final (harder to fix).
- Discarded inputs should be taken care of.

Example



Same IO-behaviours: $IOBeh(\mathcal{A}_1) = \{(\varepsilon, \varepsilon)\} \cup \{(a^n, x) \mid n \geq 1\} = IOBeh(\mathcal{A}_2)$

But different rational relations:

$$
\begin{aligned}
\mathcal{R}(\mathcal{A}_1) &= \{(a, x)\} \\
\mathcal{R}(\mathcal{A}_2) &= \{(a^n, x) \mid n \geq 1\}
\end{aligned}
$$

$\mathrm{Rat}(B^*)$-automata

Definition

A $\mathrm{Rat}(B^*)$-automaton over A is a tuple $\mathcal{A} = (S, A, \lambda, \mu, \gamma)$ where

- S is the finite set of states
- $\lambda : S \to \mathrm{Rat}(B^*)$ $\qquad \xrightarrow{\lambda_s} s$

  A word in λ_s is emitted when entering $\mathcal{A}$ in state s.
- $\mu : A \to (S \times S \to \mathrm{Rat}(B^*))$ $\qquad r \xrightarrow{a \ / \ \mu(a)_{r,s}} s$

  A word in $\mu(a)_{r,s}$ is emitted when taking a transition from r to s labelled a.
- $\gamma : S \to \mathrm{Rat}(B^*)$ $\qquad s \xrightarrow{\gamma_s}$

  A word in γ_s is emitted when exiting $\mathcal{A}$ in state s.

Then, $(u, v) \in \mathcal{R}(\mathcal{A})$ if there is a path $P = s_0 \xrightarrow{a_1} s_1 \cdots s_{n-1} \xrightarrow{a_n} s_n$ in $\mathcal{A}$ with

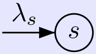
- $u = a_1 \cdots a_n$
- $v \in \lambda_{s_0} \mu(a_1)_{s_0,s_1} \cdots \mu(a_n)_{s_{n-1},s_n} \gamma_{s_n}$.

$\mathrm{Rat}(B^*)$-automata

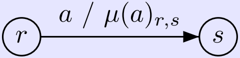
Definition

A $\mathrm{Rat}(B^*)$-automaton over A is a tuple $\mathcal{A} = (S, A, \lambda, \mu, \gamma)$ where

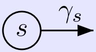
- S is the finite set of states
- $\lambda : S \to \mathrm{Rat}(B^*)$



  A word in λ_s is emitted when entering $\mathcal{A}$ in state s.
- $\mu : A \to (S \times S \to \mathrm{Rat}(B^*))$



  A word in $\mu(a)_{r,s}$ is emitted when taking a transition from r to s labelled a.
- $\gamma : S \to \mathrm{Rat}(B^*)$



  A word in γ_s is emitted when exiting $\mathcal{A}$ in state s.

Then, $(u, v) \in \mathcal{R}(\mathcal{A})$ if there is a path $P = s_0 \xrightarrow{a_1} s_1 \cdots s_{n-1} \xrightarrow{a_n} s_n$ in $\mathcal{A}$ with

- $u = a_1 \cdots a_n$
- $v \in \lambda_{s_0} \mu(a_1)_{s_0,s_1} \cdots \mu(a_n)_{s_{n-1},s_n} \gamma_{s_n}$.

$\mathrm{Rat}(B^*)$-automata and rational relations

Theorem

A relation $R \subseteq A^* \times B^*$ is rational iff there exists a $\mathrm{Rat}(B^*)$-automaton $\mathcal{A}$ with $R = \mathcal{R}(\mathcal{A})$.

Theorem

If $|A| \geq 2$ then equivalence is undecidable for $\mathrm{Rat}(B^*)$-automata over A.
This holds even if

- $|B| = 1$
- We use only finite languages: $\mathcal{P}_{\text{fin}}(B^*)$-automata
- There is no output when entering the automaton: $\lambda_s \neq \emptyset$ implies $\lambda_s = \{\varepsilon\}$
- There is no output when exiting the automaton: $\gamma_s \neq \emptyset$ implies $\gamma_s = \{\varepsilon\}$
- All transitions are visible: $\varepsilon \notin \mu(a)_{r,s}$

$\mathrm{Rat}(B^*)$-automata and rational relations

Theorem

A relation $R \subseteq A^* \times B^*$ is rational iff there exists a $\mathrm{Rat}(B^*)$-automaton $\mathcal{A}$ with $R = \mathcal{R}(\mathcal{A})$.

Theorem

If $|A| \geq 2$ then equivalence is undecidable for $\mathrm{Rat}(B^*)$-automata over A.
This holds even if

- $|B| = 1$
- We use only finite languages: $\mathcal{P}_{\mathsf{fin}}(B^*)$-automata
- There is no output when entering the automaton: $\lambda_s \neq \emptyset$ implies $\lambda_s = \{\varepsilon\}$
- There is no output when exiting the automaton: $\gamma_s \neq \emptyset$ implies $\gamma_s = \{\varepsilon\}$
- All transitions are visible: $\varepsilon \notin \mu(a)_{r,s}$

Undecidability of IO-equivalence

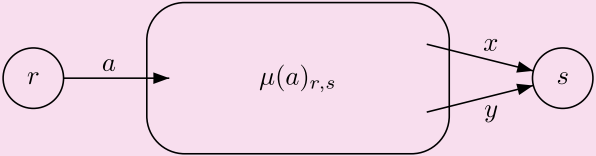
Theorem

IO-equivalence is undecidable.

Proof

Let $\mathcal{A} = (S, A, \lambda, \mu, \gamma)$ be a $\mathcal{P}_{\mathsf{fin}}(B^+)$-automaton. Define $\mathcal{A}' = (S', \Sigma, I', T')$ by

- $\Sigma_i = A$, $\Sigma_o = B \uplus \{\#\}$ and $I' = \{s \in I \mid \lambda_s \neq \emptyset$ (i.e., $\lambda_s = \{\varepsilon\}$)$\}$
- transitions $r \xrightarrow{a \ / \ \mu(a)_{r,s}} s$ of $\mathcal{A}$ are replaced in $\mathcal{A}'$ by



Note that deadlocked states of $\mathcal{A}'$ are exactly the states of $\mathcal{A}$.

Claim: $(u, v) \in IOBeh(\mathcal{A}')$ iff there is a path $s_0 \xrightarrow{a_1} s_1 \cdots s_{n-1} \xrightarrow{a_n} s_n$ in $\mathcal{A}$ with $\lambda_{s_0} = \{\varepsilon\}$, $v \in \mu(a_1)_{s_0,s_1} \cdots \mu(a_n)_{s_{n-1},s_n}$, and $u = a_1 \cdots a_n$ or $u = a_1 \cdots a_n a u'$ with $\mu(a)_{s_n,s} = \emptyset$ for all $s \in S$.

Undecidability of IO-equivalence

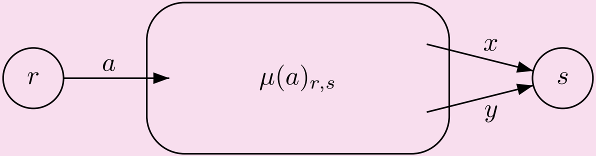
Theorem

IO-equivalence is undecidable.

Proof

Let $\mathcal{A} = (S, A, \lambda, \mu, \gamma)$ be a $\mathcal{P}_{\text{fin}}(B^+)$-automaton. Define $\mathcal{A}' = (S', \Sigma, I', T')$ by

- $\Sigma_i = A$, $\Sigma_o = B \uplus \{\#\}$ and $I' = \{s \in I \mid \lambda_s \neq \emptyset$ (i.e., $\lambda_s = \{\varepsilon\}$)$\}$
- transitions $r \xrightarrow{a \ / \ \mu(a)_{r,s}} s$ of $\mathcal{A}$ are replaced in $\mathcal{A}'$ by



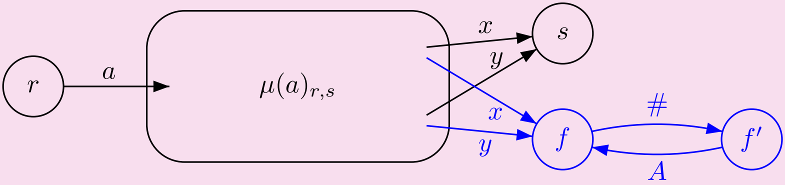
Note that deadlocked states of $\mathcal{A}'$ are exactly the states of $\mathcal{A}$.

Claim: $(u, v) \in IOBeh(\mathcal{A}')$ iff there is a path $s_0 \xrightarrow{a_1} s_1 \cdots s_{n-1} \xrightarrow{a_n} s_n$ in $\mathcal{A}$ with $\lambda_{s_0} = \{\varepsilon\}$, $v \in \mu(a_1)_{s_0,s_1} \cdots \mu(a_n)_{s_{n-1},s_n}$, and $u = a_1 \cdots a_n$ or $u = a_1 \cdots a_n a u'$ with $\mu(a)_{s_n,s} = \emptyset$ for all $s \in S$.

Undecidability of IO-equivalence

Proof continued

Define $\mathcal{A}'' = (S'', \Sigma, I', T'')$ by adding to $\mathcal{A}'$ when $\gamma_s = \{\varepsilon\}$:



Note that deadlocked states of $\mathcal{A}'$ are exactly the states in $S \uplus \{f'\}$.

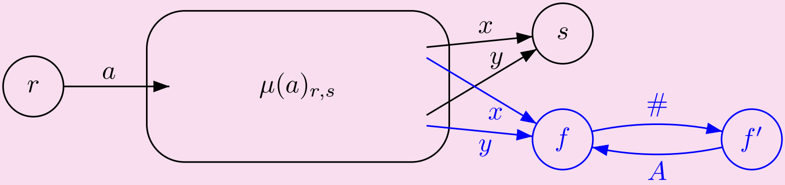
Lemma $IOBeh(\mathcal{A}'') = IOBeh(\mathcal{A}') \cup \mathcal{R}(\mathcal{A}) \cdot \{(x, \#^{1+|x|}) \mid x \in A^*\}$.

Lemma $\mathcal{A}' \uplus \mathcal{B}'' \sim_{io} \mathcal{A}'' \uplus \mathcal{B}'$ if and only if $\mathcal{R}(\mathcal{A}) = \mathcal{R}(\mathcal{B})$.

Undecidability of IO-equivalence

Proof continued

Define $\mathcal{A}'' = (S'', \Sigma, I', T'')$ by adding to $\mathcal{A}'$ when $\gamma_s = \{\varepsilon\}$:



Note that deadlocked states of $\mathcal{A}'$ are exactly the states in $S \uplus \{f'\}$.

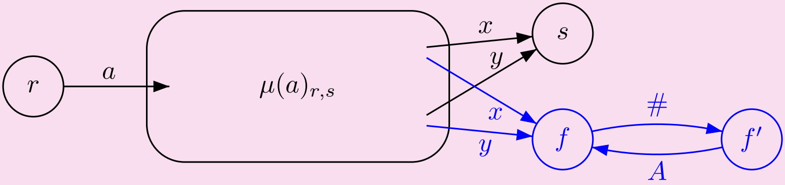
Lemma $IOBeh(\mathcal{A}'') = IOBeh(\mathcal{A}') \cup \mathcal{R}(\mathcal{A}) \cdot \{(x, \#^{1+|x|}) \mid x \in A^*\}$.

Lemma $\mathcal{A}' \uplus \mathcal{B}'' \sim_{io} \mathcal{A}'' \uplus \mathcal{B}'$ if and only if $\mathcal{R}(\mathcal{A}) = \mathcal{R}(\mathcal{B})$.

Undecidability of IO-equivalence

Proof continued

Define $\mathcal{A}'' = (S'', \Sigma, I', T'')$ by adding to $\mathcal{A}'$ when $\gamma_s = \{\varepsilon\}$:



Note that deadlocked states of $\mathcal{A}'$ are exactly the states in $S \uplus \{f'\}$.

Lemma $IOBeh(\mathcal{A}'') = IOBeh(\mathcal{A}') \cup \mathcal{R}(\mathcal{A}) \cdot \{(x, \#^{1+|x|}) \mid x \in A^*\}$.

Lemma $\mathcal{A}' \uplus \mathcal{B}'' \sim_{io} \mathcal{A}'' \uplus \mathcal{B}'$ if and only if $\mathcal{R}(\mathcal{A}) = \mathcal{R}(\mathcal{B})$.

Outline

Asynchronous IO-blocks semantics

Definition

A *block observation* of $TS = (S, \Sigma, I, T)$ is a sequence $(u_0, v_0)(u_1, v_1) \cdots (u_n, v_n)$ where

- $u_0 \in \Sigma_i^*$ and $u_j \in \Sigma_i^+$ for $1 \leq j \leq n$,
- $v_k \in \Sigma_o^*$ for $0 \leq k \leq n$

and there is a maximal run $r \xrightarrow{w_0} s_0 \xrightarrow{w_1} \cdots \xrightarrow{w_n} s_n$ with $r \in I$ such that:

- The states s_0, s_1, s_2, ..., s_n are the only deadlocked states along this run.
- $\forall 0 \leq j \leq n$, $\pi_o(w_j) = v_j$.
- $\forall 0 \leq j < n$, $\pi_i(w_j) = u_j$.
- Either $\pi_i(w_n) = u_n$ or there exists $a \in \Sigma_i$ with $\pi_i(w_n)a \preceq u_n$ and $s_n \stackrel{a}{\not\rightarrow}$.

Let $IOBlocks(TS)$ denote the set of block observations of TS.

IO-block equivalence

IO-block equivalence

Two transition systems TS and TS' are IO-block equivalent if

$$IOBlocks(TS) = IOBlocks(TS')$$

This equivalence is denoted $TS \sim_{ioblock} TS'$.

Remark

IO-block equivalence corresponds to the queued suspenstion trace equivalence of

A. Petrenko, N. Yevtushenko and J.L. Huo: Testing Transition Systems with Input and Output Testers, *Proc of TestCom 2003*.

IO-block equivalence

IO-block equivalence

Two transition systems TS and TS' are IO-block equivalent if

$$IOBlocks(TS) = IOBlocks(TS')$$

This equivalence is denoted $TS \sim_{ioblock} TS'$.

Remark

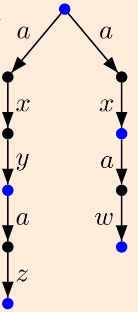
IO-block equivalence corresponds to the queued suspenstion trace equivalence of

A. Petrenko, N. Yevtushenko and J.L. Huo: Testing Transition Systems with Input and Output Testers, *Proc of TestCom 2003*.

IO-block equivalence

Example

TS_1



$IOBlocks(TS_1)$:
$(\varepsilon, \varepsilon)$
(a, xy)
$(a, xy)(a^n, z)$ for $n \geq 1$
(a, x)
$(a, x)(a^n, w)$ for $n \geq 1$

$IOBeh(TS_1)$:
$(\varepsilon, \varepsilon)$
(a, xy)
(a^n, xyz) for $n \geq 2$
(a, x)
(a^n, xw) for $n \geq 2$

Proposition

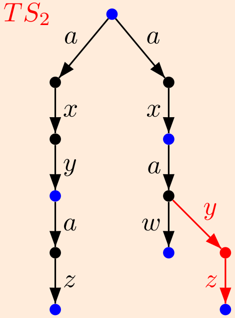
If $TS_1 \sim_{ioblock} TS_2$, then $TS_1 \sim_{io} TS_2$.

Proof

$$IOBeh(TS) = \{(u_0 u_1 \dots u_n, v_0 v_1 \dots v_n) \mid (u_0, v_0)(u_1, v_1) \dots (u_n, v_n) \in IOBlocks(TS)\}$$

IO-block equivalence

Example



$IOBlocks(\color{red}{TS_2})$:
$(\varepsilon, \varepsilon)$
(a, xy)
$(a, xy)(a^n, z)$ for $n \geq 1$
(a, x)
$(a, x)(a^n, w)$ for $n \geq 1$
$\color{red}{(a, x)(a^n, yz)}$ for $\color{red}{n \geq 1}$

$IOBeh(\color{red}{TS_2})$:
$(\varepsilon, \varepsilon)$
(a, xy)
(a^n, xyz) for $n \geq 2$
(a, x)
(a^n, xw) for $n \geq 2$

Proposition

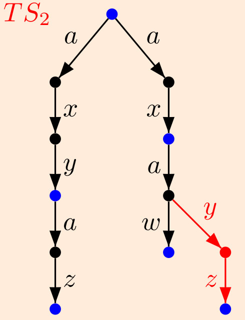
If $TS_1 \sim_{ioblock} TS_2$, then $TS_1 \sim_{io} TS_2$.

Proof

$$IOBeh(TS) = \{(u_0u_1 \ldots u_n, v_0v_1 \ldots v_n) \mid (u_0, v_0)(u_1, v_1) \ldots (u_n, v_n) \in IOBlocks(TS)\}$$

IO-block equivalence

Example



$IOBlocks(TS_2)$:
$(\varepsilon, \varepsilon)$
(a, xy)
$(a, xy)(a^n, z)$ for $n \geq 1$
(a, x)
$(a, x)(a^n, w)$ for $n \geq 1$
$(a, x)(a^n, yz)$ for $n \geq 1$

$IOBeh(TS_2)$:
$(\varepsilon, \varepsilon)$
(a, xy)
(a^n, xyz) for $n \geq 2$
(a, x)
(a^n, xw) for $n \geq 2$

Proposition

If $TS_1 \sim_{ioblock} TS_2$, then $TS_1 \sim_{io} TS_2$.

Proof

$$IOBeh(TS) = \{(u_0 u_1 \ldots u_n, v_0 v_1 \ldots v_n) \mid (u_0, v_0)(u_1, v_1) \ldots (u_n, v_n) \in IOBlocks(TS)\}$$

Decidability of IO-block equivalence

Definition

A transition system is well-structured if every state either refuses Σ_i or refuses Σ_o.

Theorem

For finite well structured transition systems, $\sim_{ioblock}$ is decidable.

Proof

Let $D = \{s \in S \mid s \text{ is deadlocked}\}$.
For $a \in \Sigma_i$, let $D_a = \{s \in S \mid s \text{ is deadlocked and } s \stackrel{a}{\not\to}\}$.
For $X \subseteq S$, let $L_X(TS) = \{w \in \Sigma^* \mid I \xrightarrow{w} X\}$.

Claim. $TS_1 \sim_{ioblock} TS_2$ iff

- $L_D(TS_1) = L_D(TS_2)$, and
- $L_{D_a}(TS_1) = L_{D_a}(TS_2)$ for each $a \in \Sigma_i$.

Indeed, for well structured transition systems, we have

$$IOBlocks(TS) = \{(\varepsilon, v_0)(a_1, v_1)\dots(a_n, v_n) \mid i \xrightarrow{v_0} s_0 \xrightarrow{a_1 v_1} s_1 \cdots s_{n-1} \xrightarrow{a_n v_n} s_n\}$$

$$\cup \{(\varepsilon, v_0)(a_1, v_1)\dots(a_n au, v_n) \mid i \xrightarrow{v_0} s_0 \xrightarrow{a_1 v_1} s_1 \cdots s_{n-1} \xrightarrow{a_n v_n} s_n \stackrel{a}{\not\to}\}$$

Decidability of IO-block equivalence

Definition

A transition system is well-structured if every state either refuses Σ_i or refuses Σ_o.

Theorem

For finite well structured transition systems, $\sim_{ioblock}$ is decidable.

Proof

Let $D = \{s \in S \mid s \text{ is deadlocked}\}$.
For $a \in \Sigma_i$, let $D_a = \{s \in S \mid s \text{ is deadlocked and } s \overset{a}{\not\to}\}$.
For $X \subseteq S$, let $L_X(TS) = \{w \in \Sigma^* \mid I \xrightarrow{w} X\}$.

Claim. $TS_1 \sim_{ioblock} TS_2$ iff

- $L_D(TS_1) = L_D(TS_2)$, and
- $L_{D_a}(TS_1) = L_{D_a}(TS_2)$ for each $a \in \Sigma_i$.

Indeed, for well structured transition systems, we have

$$IOBlocks(TS) = \{(\varepsilon, v_0)(a_1, v_1) \dots (a_n, v_n) \mid i \xrightarrow{v_0} s_0 \xrightarrow{a_1 v_1} s_1 \cdots s_{n-1} \xrightarrow{a_n v_n} s_n\}$$

$$\cup \{(\varepsilon, v_0)(a_1, v_1) \dots (a_n au, v_n) \mid i \xrightarrow{v_0} s_0 \xrightarrow{a_1 v_1} s_1 \cdots s_{n-1} \xrightarrow{a_n v_n} s_n \overset{a}{\not\to}\}$$

Outline

Queue semantics (Tretmans)

Definition

Let $TS = (S, \Sigma, I, T)$ be a transition system. Define $Q(TS) = (S', \Sigma, I', T')$ by

- $S' = S \times \Sigma_i^* \times \Sigma_o^*$: configurations of TS.
- $I' = I \times \{\varepsilon\} \times \{\varepsilon\}$: initial configurations
- Transitions of TS are broken up into two moves, one visible and one invisible (labelled τ):

Input
$$\frac{}{(s, \sigma_i, \sigma_o) \xrightarrow{a} (s, \sigma_i a, \sigma_o)} \qquad \frac{s \xrightarrow{a} s'}{(s, a\sigma_i, \sigma_o) \xrightarrow{\tau} (s', \sigma_i, \sigma_o)}$$

Output
$$\frac{s \xrightarrow{x} s'}{(s, \sigma_i, \sigma_o) \xrightarrow{\tau} (s', \sigma_i, \sigma_o x)} \qquad \frac{}{(s, \sigma_i, x\sigma_o) \xrightarrow{x} (s, \sigma_i, \sigma_o)}$$

- $L(Q(TS))$ is the set of traces of $Q(TS)$.

Deadlocked traces (Tretmans)

Deadlocked traces

- A trace $w \in L(Q(TS))$ is deadlocked if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, \sigma_i, \varepsilon)$ with $r \in I$ and $(s, \sigma_i, \varepsilon)$ deadlocked in $Q(TS)$.
- We denote by $\delta_{\text{traces}}(Q(TS))$ the set of deadlocked traces of $Q(TS)$.

Empty and blocked deadlocked traces

- A trace $w \in L(Q(TS))$ is an empty deadlock if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, \varepsilon, \varepsilon)$ with $r \in I$ and s deadlocked in TS.
- We denote by $\delta_{\text{empty}}(Q(TS))$ the empty deadlocked traces of $Q(TS)$.
- A trace $w \in L(Q(TS))$ is an blocked deadlock if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, a\sigma_i, \varepsilon)$ with $r \in I$ and in TS, s deadlocked and $s \stackrel{a}{\not\rightarrow}$.
- We denote by $\delta_{\text{block}}(Q(TS))$ the blocked deadlocked traces of $Q(TS)$.

Proposition: $\delta_{\text{traces}}(Q(TS)) = \delta_{\text{empty}}(Q(TS)) \cup \delta_{\text{block}}(Q(TS))$

Deadlocked traces (Tretmans)

Deadlocked traces

- A trace $w \in L(Q(TS))$ is deadlocked if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, \sigma_i, \varepsilon)$ with $r \in I$ and $(s, \sigma_i, \varepsilon)$ deadlocked in $Q(TS)$.
- We denote by $\delta_{\text{traces}}(Q(TS))$ the set of deadlocked traces of $Q(TS)$.

Empty and blocked deadlocked traces

- A trace $w \in L(Q(TS))$ is an empty deadlock if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, \varepsilon, \varepsilon)$ with $r \in I$ and s deadlocked in TS.
- We denote by $\delta_{\text{empty}}(Q(TS))$ the empty deadlocked traces of $Q(TS)$.
- A trace $w \in L(Q(TS))$ is an blocked deadlock if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, a\sigma_i, \varepsilon)$ with $r \in I$ and in TS, s deadlocked and $s \overset{a}{\nrightarrow}$.
- We denote by $\delta_{\text{block}}(Q(TS))$ the blocked deadlocked traces of $Q(TS)$.

Proposition: $\delta_{\text{traces}}(Q(TS)) = \delta_{\text{empty}}(Q(TS)) \cup \delta_{\text{block}}(Q(TS))$

Deadlocked traces (Tretmans)

Deadlocked traces

- A trace $w \in L(Q(TS))$ is deadlocked if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, \sigma_i, \varepsilon)$ with $r \in I$ and $(s, \sigma_i, \varepsilon)$ deadlocked in $Q(TS)$.
- We denote by $\delta_{\text{traces}}(Q(TS))$ the set of deadlocked traces of $Q(TS)$.

Empty and blocked deadlocked traces

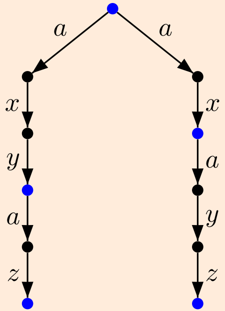
- A trace $w \in L(Q(TS))$ is an empty deadlock if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, \varepsilon, \varepsilon)$ with $r \in I$ and s deadlocked in TS.
- We denote by $\delta_{\text{empty}}(Q(TS))$ the empty deadlocked traces of $Q(TS)$.
- A trace $w \in L(Q(TS))$ is an blocked deadlock if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, a\sigma_i, \varepsilon)$ with $r \in I$ and in TS, s deadlocked and $s \overset{a}{\nrightarrow}$.
- We denote by $\delta_{\text{block}}(Q(TS))$ the blocked deadlocked traces of $Q(TS)$.

Proposition: $\delta_{\text{traces}}(Q(TS)) = \delta_{\text{empty}}(Q(TS)) \cup \delta_{\text{block}}(Q(TS))$

Deadlocked traces (Tretmans)

Example

TS_1

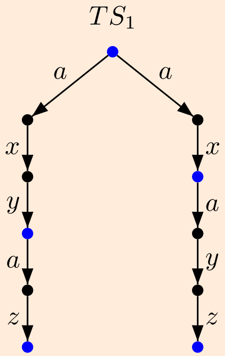


$\delta_{\text{empty}}(TS_1)$:
ε
ax
axy
$axayz$
$axyaz$
$aaxyz$

$\delta_{\text{block}}(TS_1)$:
$axyaza^n$ for $n \geq 1$
$axayza^n$ for $n \geq 1$
$aaxyza^n$ for $n \geq 1$
axa^nyaz for $n \geq 1$
$\ldots$

Deadlocked traces (Tretmans)

Example



$\delta_{\mathsf{empty}}(TS_1)$:
ε
ax
axy
$axayz$
$axyaz$
$aaxyz$

$\delta_{\mathsf{block}}(TS_1)$:
$axyaza^n$ for $n \geq 1$
$axayza^n$ for $n \geq 1$
$aaxyza^n$ for $n \geq 1$
axa^nyaz for $n \geq 1$
...

Queue equivalence (Tretmans)

Definition

$$TS \sim_Q TS' \quad \stackrel{\text{def}}{=} \quad Q(TS) \sim_{syn} Q(TS').$$

Intuitively, synchronous testing equivalence $\sim_{syn}$ corresponds to failure semantics.

Proposition (Tretmans)

$TS \sim_Q TS'$ iff $L(Q(TS)) = L(Q(TS'))$ and $\delta_{\text{traces}}(Q(TS)) = \delta_{\text{traces}}(Q(TS'))$

Ape relation (Tretmans)

Ape relation for the queue semantics

- Output actions may always be postponed:
  For $x \in \Sigma_o$ and $a \in \Sigma_i$, we have
  $$w_1 x a w_2 \in L(Q(TS)) \text{ implies } w_1 a x w_2 \in L(Q(TS)).$$
- Input actions may always be added:
  For $a \in \Sigma_i$, we have
  $$w \in L(Q(TS)) \text{ implies } wa \in L(Q(TS)).$$
- We denote @ the reflexive and transitive closure of the relations postponing an output action or adding an input action.
- $\mathrm{Tracks}(TS)$ is the set of @-minimal words in $L(Q(TS))$.
- $L(Q(TS))$ is @-upward closure of $\mathrm{Tracks}(TS)$.
- $\mathrm{Tracks}(TS) \subseteq L(TS)$.
- $\delta_{\mathsf{block}}(Q(TS)) = \{w \in \Sigma^* \mid \exists\, r \xrightarrow{w'} s$ in TS with $r \in I, s$ deadlocked, and $\exists\, a \in \Sigma_i$ such that $s \overset{a}{\not\to}$ and $w'a \;@\; w\}$.

Ape relation (Tretmans)

Ape relation for the queue semantics

- Output actions may always be postponed:
  For $x \in \Sigma_o$ and $a \in \Sigma_i$, we have
  $$w_1 x a w_2 \in L(Q(TS)) \text{ implies } w_1 a x w_2 \in L(Q(TS)).$$
- Input actions may always be added:
  For $a \in \Sigma_i$, we have
  $$w \in L(Q(TS)) \text{ implies } wa \in L(Q(TS)).$$
- We denote @ the reflexive and transitive closure of the relations postponing an output action or adding an input action.
- $\mathrm{Tracks}(TS)$ is the set of @-minimal words in $L(Q(TS))$.
- $L(Q(TS))$ is @-upward closure of $\mathrm{Tracks}(TS)$.
- $\mathrm{Tracks}(TS) \subseteq L(TS)$.
- $\delta_{\mathsf{block}}(Q(TS)) = \{w \in \Sigma^* \mid \exists\, r \xrightarrow{w'} s$ in TS with $r \in I, s$ deadlocked, and $\exists\, a \in \Sigma_i$ such that $s \overset{a}{\nrightarrow}$ and $w'a \,@\, w\}$.

Ape relation (Tretmans)

Ape relation for the queue semantics

- Output actions may always be postponed:
  For $x \in \Sigma_o$ and $a \in \Sigma_i$, we have
  $$w_1 x a w_2 \in L(Q(TS)) \text{ implies } w_1 a x w_2 \in L(Q(TS)).$$
- Input actions may always be added:
  For $a \in \Sigma_i$, we have
  $$w \in L(Q(TS)) \text{ implies } wa \in L(Q(TS)).$$
- We denote @ the reflexive and transitive closure of the relations postponing an output action or adding an input action.
- $\mathrm{Tracks}(TS)$ is the set of @-minimal words in $L(Q(TS))$.
- $L(Q(TS))$ is @-upward closure of $\mathrm{Tracks}(TS)$.
- $\mathrm{Tracks}(TS) \subseteq L(TS)$.
- $\delta_{\mathsf{block}}(Q(TS)) = \{w \in \Sigma^* \mid \exists\ r \xrightarrow{w'} s$ in TS with $r \in I, s$ deadlocked, and $\exists\ a \in \Sigma_i$ such that $s \stackrel{a}{\nrightarrow}$ and $w'a \mathbin{@} w\}$.

Ape relation (Tretmans)

Ape relation for the queue semantics

- Output actions may always be postponed:
  For $x \in \Sigma_o$ and $a \in \Sigma_i$, we have
  $$w_1 x a w_2 \in L(Q(TS)) \text{ implies } w_1 a x w_2 \in L(Q(TS)).$$
- Input actions may always be added:
  For $a \in \Sigma_i$, we have
  $$w \in L(Q(TS)) \text{ implies } wa \in L(Q(TS)).$$
- We denote @ the reflexive and transitive closure of the relations postponing an output action or adding an input action.
- $\mathrm{Tracks}(TS)$ is the set of @-minimal words in $L(Q(TS))$.
- $L(Q(TS))$ is @-upward closure of $\mathrm{Tracks}(TS)$.
- $\mathrm{Tracks}(TS) \subseteq L(TS)$.
- $\delta_{\text{block}}(Q(TS)) = \{w \in \Sigma^* \mid \exists\, r \xrightarrow{w'} s$ in TS with $r \in I, s$ deadlocked, and $\exists\, a \in \Sigma_i$ such that $s \stackrel{a}{\nrightarrow}$ and $w'a \ @ \ w\}$.

Ape relation (Tretmans)

Ape relation for the queue semantics

- Output actions may always be postponed:
  For $x \in \Sigma_o$ and $a \in \Sigma_i$, we have
  $$w_1 x a w_2 \in L(Q(TS)) \text{ implies } w_1 a x w_2 \in L(Q(TS)).$$
- Input actions may always be added:
  For $a \in \Sigma_i$, we have
  $$w \in L(Q(TS)) \text{ implies } wa \in L(Q(TS)).$$
- We denote @ the reflexive and transitive closure of the relations postponing an output action or adding an input action.
- $\mathrm{Tracks}(TS)$ is the set of @-minimal words in $L(Q(TS))$.
- $L(Q(TS))$ is @-upward closure of $\mathrm{Tracks}(TS)$.
- $\mathrm{Tracks}(TS) \subseteq L(TS)$.
- $\delta_{\mathsf{block}}(Q(TS)) = \{w \in \Sigma^* \mid \exists\, r \xrightarrow{w'} s$ in TS with $r \in I, s$ deadlocked, and $\exists\, a \in \Sigma_i$ such that $s \overset{a}{\not\to}$ and $w'a \mathbin{@} w\}$.

Ape relation (Tretmans)

Ape relation for the queue semantics

- Output actions may always be postponed:
  For $x \in \Sigma_o$ and $a \in \Sigma_i$, we have
  $$w_1 x a w_2 \in L(Q(TS)) \text{ implies } w_1 a x w_2 \in L(Q(TS)).$$
- Input actions may always be added:
  For $a \in \Sigma_i$, we have
  $$w \in L(Q(TS)) \text{ implies } wa \in L(Q(TS)).$$
- We denote @ the reflexive and transitive closure of the relations postponing an output action or adding an input action.
- $\mathrm{Tracks}(TS)$ is the set of @-minimal words in $L(Q(TS))$.
- $L(Q(TS))$ is @-upward closure of $\mathrm{Tracks}(TS)$.
- $\mathrm{Tracks}(TS) \subseteq L(TS)$.
- $\delta_{\mathsf{block}}(Q(TS)) = \{w \in \Sigma^* \mid \exists\, r \xrightarrow{w'} s$ in TS with $r \in I, s$ deadlocked, and $\exists\, a \in \Sigma_i$ such that $s \not\xrightarrow{a}$ and $w'a \;@\; w\}$.

Ape relation (Tretmans)

Ape relation for the queue semantics

- Output actions may always be postponed:
  For $x \in \Sigma_o$ and $a \in \Sigma_i$, we have
  $$w_1 x a w_2 \in L(Q(TS)) \text{ implies } w_1 a x w_2 \in L(Q(TS)).$$
- Input actions may always be added:
  For $a \in \Sigma_i$, we have
  $$w \in L(Q(TS)) \text{ implies } wa \in L(Q(TS)).$$
- We denote @ the reflexive and transitive closure of the relations postponing an output action or adding an input action.
- $\mathrm{Tracks}(TS)$ is the set of @-minimal words in $L(Q(TS))$.
- $L(Q(TS))$ is @-upward closure of $\mathrm{Tracks}(TS)$.
- $\mathrm{Tracks}(TS) \subseteq L(TS)$.
- $\delta_{\mathsf{block}}(Q(TS)) = \{w \in \Sigma^* \mid \exists\ r \xrightarrow{w'} s$ in TS with $r \in I, s$ deadlocked, and $\exists\ a \in \Sigma_i$ such that $s \stackrel{a}{\nrightarrow}$ and $w'a \ @ \ w\}$.

Strict ape relation (Tretmans)

Strict ape relation for the queue semantics

- We denote $|@|$ the reflexive and transitive closure of the relation postponing an output action.
- $\delta_{\text{empty}}(Q(TS)) = \{w \in \Sigma^* \mid \exists\, r \xrightarrow{w'} s$ in TS with $r \in I$, s deadlocked and $w' \;|@|\; w\}$.

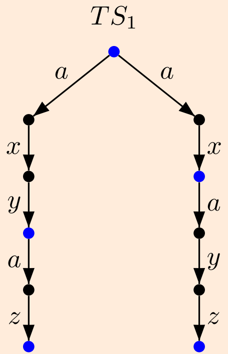
Strict ape relation (Tretmans)

Strict ape relation for the queue semantics

- We denote $|@|$ the reflexive and transitive closure of the relation postponing an output action.
- $\delta_{\text{empty}}(Q(TS)) = \{w \in \Sigma^* \mid \exists\, r \xrightarrow{w'} s$ in TS with $r \in I$, s deadlocked and $w' \; |@| \; w\}$.

Deadlocked traces (Tretmans)

Example



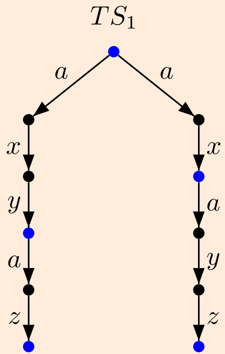
$\delta_{\mathsf{empty}}(TS_1)$:
$|@|$-upper closure of
ε
ax
axy
$axyaz$

$axyaz \; |@| \; axayz$
$axyaz \; |@| \; aaxyz$

$\delta_{\mathsf{block}}(TS_1)$:
$@$-upper closure of
$axyaza$

Deadlocked traces (Tretmans)

Example



$\delta_{\mathsf{empty}}(TS_1)$:
$|@|$-upper closure of
ε
ax
axy
$axyaz$

$axyaz \; |@| \; axayz$
$axyaz \; |@| \; aaxyz$

$\delta_{\mathsf{block}}(TS_1)$:
@-upper closure of
$axyaza$

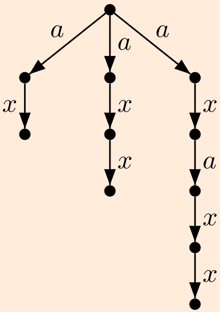
Comparing the equivalences

Proposition

If $TS_1 \sim_Q TS_2$, then $TS_1 \sim_{io} TS_2$.

The converse does not hold

TS_1



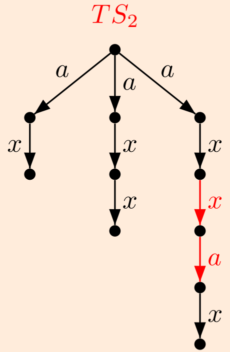
$\text{Tracks}(TS_1)$:	$IOBeh(TS_1)$:
ε	$(\varepsilon, \varepsilon)$
ax	(a^n, x) for $n \geq 1$
axx	(a^n, x^2) for $n \geq 1$
$axaxx$	(a^n, x^3) for $n \geq 2$

Comparing the equivalences

Proposition

If $TS_1 \sim_Q TS_2$, then $TS_1 \sim_{io} TS_2$.

The converse does not hold



$\mathrm{Tracks}(TS_2)$:
ε
ax
axx
$axxax$

$axxax$ @ $axaxx$

$IOBeh(TS_2)$:
$(\varepsilon, \varepsilon)$
(a^n, x) for $n \geq 1$
(a^n, x^2) for $n \geq 1$
(a^n, x^3) for $n \geq 2$

Undecidability of $Qtest$

Theorem

$\sim_Q$ is undecidable

Proof

Reduction from the PCP problem.

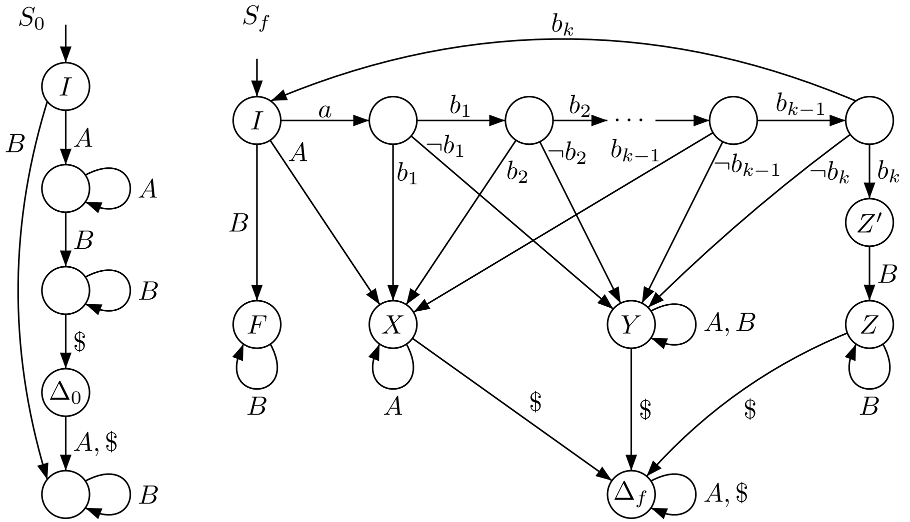
A PCP instance consists in two morphisms $f, g : A^+ \to B^+$ where A, B are finite alphabets.

The PCP instance f, g has a solution if there exists $u \in A^+$ such that $f(u) = g(u)$.

We construct two systems M_1 and M_2 such that the PCP instance (f, g) has no solution iff $M_1 \sim_Q M_2$.

Reduction from the PCP problem

Let $f, g : A^+ \to B^+$ be a PCP instance. We define



Reduction from the PCP problem

We want to compare the following two systems:

- $M_1 = S_0 + S_f + S_g$
- $M_2 = S_f + S_g$

Lemma

$\delta_{\mathsf{block}}(M_1) = \delta_{\mathsf{block}}(M2) = \emptyset$.

Lemma

$\mathrm{Tracks}(M_1) = \mathrm{Tracks}(M_2) = \mathrm{Tracks}(S_f) = B^*$.

Lemma

- $\delta_{\mathsf{empty}}(S_0)$ is the $|@|$-upper closure of $A^+B^+\$$.
- Let $u \in A^+$ and $v \in B^+$. Then, $uv\$ \in \delta_{\mathsf{empty}}(S_f)$ if and only if $v \neq f(u)$.

Theorem

$M_1 \sim_Q M_2$ iff the PCP instance (f, g) has no solution.

Outline

Conclusion

Summary

- We have investigated 3 asynchronous testing equivalences.
- We have shown that $\sim_{io}$ is strictly weaker than $\sim_Q$ and $\sim_{ioblock}$, but $\sim_Q$ and $\sim_{ioblock}$ are incomparable.
- $\sim_{ioblock}$ is decidable, while $\sim_{io}$ and $\sim_Q$ are undecidable.

Open problems

- Construct test suites based on the IO-Blocks semantics.
- Investigate distributed testing.
  See e.g. C. Jard: Synthesis of distributed testers from true-concurrency models of reactive systems, Information & Software Technology, 2003.

Conclusion

Summary

- We have investigated 3 asynchronous testing equivalences.
- We have shown that $\sim_{io}$ is strictly weaker than $\sim_Q$ and $\sim_{ioblock}$, but $\sim_Q$ and $\sim_{ioblock}$ are incomparable.
- $\sim_{ioblock}$ is decidable, while $\sim_{io}$ and $\sim_Q$ are undecidable.

Open problems

- Construct test suites based on the IO-Blocks semantics.
- Investigate distributed testing.
  See e.g. C. Jard: Synthesis of distributed testers from true-concurrency models of reactive systems, Information & Software Technology, 2003.