Testing with asynchronous communication

Paul Gastin

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

Joint work with Puneet Bhateja and Madhavan Mukund

October 29th, 2008
Introduction

Input/Output semantics

IO-Blocks semantics

Queue semantics (Tretman)

Conclusion
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.
Related work


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 quiescent if it refuses all output actions: \( s \xrightarrow{\Sigma_o} \).

Some further properties

- No infinite output-only behaviour.
- Receptiveness: \( \forall s \in S \) quiescent, \( \forall a \in \Sigma_i, s \xrightarrow{a} \)
  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 run \( i \xrightarrow{w} s \) in \( TS \) with

- \( i \in I \) and \( s \) quiescent
- \( \pi_o(w) = v \), and
- either \( \pi_i(w) = u \) or there exists \( a \in \Sigma_i \) such that \( \pi_i(w)a \preceq u \) and \( s \xrightarrow{a} \).

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 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

\( TS_1 \)

\( \begin{align*}
T \rightarrow & a \\
x \rightarrow & x \\
a \rightarrow & a \\
x \rightarrow & y \\
\end{align*} \)

\( TS_2 \)

\( \begin{align*}
T \rightarrow & a \\
x \rightarrow & x \\
a \rightarrow & a \\
x \rightarrow & xy \\
\end{align*} \)

\( IOBeh(TS_1) = IOBeh(TS_2): \)

\( (\varepsilon, \varepsilon), (a, x), (a, xy), (a^n, x), (a^n, xy), (a^n, x^2) \) if \( n \geq 2 \).

\( TS_1 \) and \( TS_2 \) are IO-equivalent.

IO-equivalence corresponds to the queued quiescent trace equivalence of

**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 $A = (S, A \cup B, I, F, T)$ such that

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

**Example**

$$\mathcal{R}(A) = \{(a, x), (a, xy), (a^2, x^2)\}$$
From IO-behaviours to rational relations

Proposition

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

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

Proof. Intuition: transform quiescent states into final states

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

- $S' = S \uplus \overline{D} \uplus \{f\}$ where $\overline{D}$ is a copy of $D$.
- $I' = I \uplus \overline{I \cap D}$ and $F' = \overline{D} \uplus \{f\}$
- $T' = T \cup \{(r, a, \bar{s}) | (r, a, s) \in T \text{ and } s \in D\}$
  $\cup \{(\bar{s}, a, f) | a \in \Sigma_i \text{ and } s \not\twoheadrightarrow\}$
  $\cup \{(f, a, f) | a \in \Sigma_i\}$

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 $s \not\twoheadrightarrow$.

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

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

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.
From rational relations to IO-behaviours

Several problems:

- Final states may not be quiescent (easy to fix).
- Quiescent states may not be final (harder to fix).

Example

Same rational relation: $\mathcal{R}(A_1) = \{(a^2, x^3)\} = \mathcal{R}(A_2)$

But different IO-behaviours:

$$IOBeh(A_1) = \{(\varepsilon, \varepsilon), (a, x^2)\} \cup \{(a^n, x^3) \mid n \geq 2\}$$

$$IOBeh(A_2) = \{(\varepsilon, \varepsilon), (a, x)\} \cup \{(a^n, x^3) \mid n \geq 2\}$$
From rational relations to IO-behaviours

Several problems:

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

Example

\[ \mathcal{A}_1 \]

\[ \mathcal{A}_2 \]

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:

\[ R(\mathcal{A}_1) = \{(a, x)\} \]
\[ R(\mathcal{A}_2) = \{(a^n, x) \mid n \geq 1\} \]
Definition

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

- $S$ is the finite set of states
- $\lambda : S \to \text{Rat}(B^*)$
  
  A word in $\lambda_s$ is emitted when entering $\mathcal{A}$ in state $s$.
- $\mu : A \to (S \times S \to \text{Rat}(B^*))$
  
  A word in $\mu(a)_{r,s}$ is emitted when taking a transition from $r$ to $s$ labelled $a$.
- $\gamma : S \to \text{Rat}(B^*)$
  
  A word in $\gamma_s$ 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

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

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

**Theorem**

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

- $|B| = 1$
- We use only finite languages: $P_{\text{fin}}(B^*)$-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: $\varepsilon \notin \mu(a)_{r,s}$
Undecidability of IO-equivalence

**Theorem**

IO-equivalence is undecidable if $|\Sigma_i| \geq 2$ and $|\Sigma_o| \geq 2$.

**Proof**

Let $\mathcal{A} = (S, A, \lambda, \mu, \gamma)$ be a $\mathcal{P}_{\text{fin}}(B^+)$-automaton with $|A| = 2$ and $|B| = 1$. Define $\mathcal{A}' = (S', \Sigma, I', T')$ by:

- $\Sigma_i = A$, $\Sigma_o = B \cup \{\#\}$ 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:

  ![Diagram](attachment:diagram.png)

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

**Claim:** $(u, v) \in \text{IOBeh}(\mathcal{A}')$ 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 $u = a_1 \cdots a_n au'$ with $\mu(a)_{s_n,s} = \emptyset$ for all $s \in S$. 

Undecidability of IO-equivalence

**Theorem**
IO-equivalence is undecidable if $|\Sigma_i| \geq 2$ and $|\Sigma_o| \geq 2$.

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

![Diagram](image)

Note that quiescent states of $A'$ are exactly the states in $S \cup \{f'\}$.

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

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

Introduction

Input/Output semantics

IO-Blocks semantics

Queue semantics (Tretman)

Conclusion
Asynchronous IO-blocks semantics

Definition

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

1. \(u_1 \in \Sigma_i^*\) and \(u_j \in \Sigma_i^+\) for \(1 < j \leq n\),
2. \(v_k \in \Sigma_o^*\) for \(1 \leq k \leq n\)

and there is a run \(s_0 \xrightarrow{w_1} s_1 \cdots \xrightarrow{w_k} s_k\) with \(s_0 \in I, 1 \leq k \leq n\) and:

1. \(s_1, s_2, \ldots, s_k\) are quiescent.
2. \(\pi_o(w_j) = v_j\) for \(1 \leq j \leq k\) and \(v_j = \epsilon\) for \(k < j \leq n\).
3. \(\pi_i(w_j) = u_j\) for \(0 \leq j < k\).
4. Either \(k = n\) and \(\pi_i(w_n) = u_n\) or there exists \(a \in \Sigma_i\) with \(\pi_i(w_k)a \preceq u_k\) and \(s_k \xrightarrow{a}\).

Let \(IOBlocks(TS)\) denote the set of block observations of \(TS\).
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 suspension trace equivalence of

IO-block equivalence

Example

\[ TS_1 \]

\[ \begin{align*}
  & IOBlocks(TS_1): \\
  & (\varepsilon, \varepsilon) \\
  & (a, xy) \\
  & (a, x) \\
  & (a, xy)(a^n, z) \text{ for } n \geq 1 \\
  & (a, x)(a^n, w) \text{ for } n \geq 1 \\
  & (a^n, xyz) \text{ for } n \geq 2 \\
  & (a^n, xw) \text{ for } n \geq 2
\end{align*} \]

\[ \begin{align*}
  & IOBeh(TS_1): \\
  & (\varepsilon, \varepsilon) \\
  & (a, xy) \\
  & (a, x) \\
  & (a^n, xyz) \text{ for } n \geq 2 \\
  & (a^n, xw) \text{ for } n \geq 2
\end{align*} \]

Proposition

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

Proof

\[ IOBeh(TS) = IOBlocks(TS) \cap (\Sigma_i^* \times \Sigma_o^*) \]
**Example**

IOBlocks(TS₂):
- (ε, ε)
- (a, xy)
- (a, x)
- (a, xy)(aⁿ, z) for n ≥ 1
- (a, x)(aⁿ, w) for n ≥ 1
- (aⁿ, xyz) for n ≥ 2
- (aⁿ, xw) for n ≥ 2
- (a, x)(aⁿ, yz) for n ≥ 1

IOBeh(TS₂):
- (ε, ε)
- (a, xy)
- (a, x)
- (aⁿ, xyz) for n ≥ 2
- (aⁿ, xw) for n ≥ 2

**Proposition**

If TS₁ ∼ᵢοₜᵢₒblock TS₂, then TS₁ ∼ᵢₒ TS₂.

**Proof**

IOBeh(TS) = IOBlocks(TS) ∩ (Σᵢ⁺ × Σₒ⁺)
Decidability of IO-block equivalence

Definition
A transition system is well-structured if every state either refuses $\Sigma_i$ or refuses $\Sigma_o$.

Definition
A block observation $\alpha = (u_1, v_1) \cdots (u_n, v_n)$ is reduced if $u_1 = \varepsilon$ and $u_j \in \Sigma_i$ for $1 < j \leq n$.

$\text{redIOBlocks}(TS)$ denotes the set of reduced block observations of $TS$.

Definition
Let $\alpha$ and $\beta$ be block-observations. We say that $\alpha$ is finer than $\beta$, denoted $\alpha \preceq \beta$, if $\beta$ can be obtained from $\alpha$ by merging consecutive blocks.

Lemma
Let $TS$ be well-structured. Then, $\text{IOBlocks}(TS) = \uparrow \text{redIOBlocks}(TS)$
where $\uparrow$ denotes the upward closure for $\preceq$. 
Decidability of IO-block equivalence

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

Proof

For $w = v_1a_2v_2 \cdots a_nv_n \in \Sigma^*$ with $v_j \in \Sigma_o^*$ and $a_j \in \Sigma_i$, we define the reduced block observation $f(w) = (\varepsilon, v_1)(a_2, v_2) \cdots (a_n, v_n)$.

Let $L_\delta(TS)$ be the language accepted by $TS$ with quiescent states as final states. For $a \in \Sigma_i$, let $L_{\delta,a}(TS)$ be the language accepted by $TS$ with quiescent states that refuse $a$ as final states.

$$
\text{redIOBlocks}(TS) = f \left( L_\delta(TS) \cup \bigcup_{a \in \Sigma_i} L_{\delta,a}(TS) \cdot a \cdot \Sigma_i^* \right)
$$

$$
f^{-1}(IOBlocks(TS)) = L_\delta(TS) \cup \bigcup_{a \in \Sigma_i} L_{\delta,a}(TS) \cdot a \cdot \Sigma_i^*
$$
Outline

Introduction

Input/Output semantics

IO-Blocks semantics

4 Queue semantics (Tretman)

Conclusion
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 $\tau$):
  - **Input**
    
    $$(s, \sigma_i, \sigma_o) \xrightarrow{a} (s, \sigma_i a, \sigma_o)$$

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

  - **Output**
    
    $$s \xrightarrow{x} s'$$

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

    $$(s, \sigma_i, x\sigma_o) \xrightarrow{x} (s, \sigma_i, \sigma_o)$$

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

**Definition**

\[ TS \sim_Q TS' \overset{\text{def}}{=} Q(TS) \sim_{syn} Q(TS'). \]

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

**Definition**

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

**Proposition (Tretmans)**

\[ TS \sim_Q TS' \iff L(Q(TS)) = L(Q(TS')) \text{ and } L_\delta(Q(TS)) = L_\delta(Q(TS')). \]

Pb: characterization of \( \sim_Q \) on \( TS \) instead of \( Q(TS) \).
Ape relation (Tretmans)

Ape relation for the queue semantics

- Output actions may always be postponed: \( w_1xaw_2 \Rightarrow w_1axw_2 \)
  For \( x \in \Sigma_o \) and \( a \in \Sigma_i \), we have
  \[
  w_1xaw_2 \in L(Q(TS)) \text{ implies } w_1axw_2 \in L(Q(TS)).
  \]

- Input actions may always be added: \( w \Rightarrow wa \)
  For \( a \in \Sigma_i \), we have
  \[
  w \in L(Q(TS)) \text{ implies } wa \in L(Q(TS)).
  \]

- We denote \( \Rightarrow \) the reflexive and transitive closure of the relations postponing an output action: \( w_1xaw_2 \Rightarrow w_1axw_2 \)
  or adding an input action: \( w \Rightarrow wa \).

- \( \text{Tracks}(TS) \) is the set of \( \Rightarrow \)-minimal words in \( L(Q(TS)) \).
  \( \Rightarrow \)-minimal: no trailing input, outputs as early as possible.

- \( L(Q(TS)) \) is the \( \Rightarrow \)-upward closure of \( \text{Tracks}(TS) \).

- \( \text{Tracks}(TS) \subseteq L(TS) \).

- \( L(Q(TS)) = L(Q(TS')) \) iff \( \text{Tracks}(TS) = \text{Tracks}(TS') \).
Tracks (Tretmans)

Example

Tracks(TS):
- $\varepsilon$
- $ax$
- $axy$
- $axyaz$
- $axaw$
- not $axayz$

$L(Q(TS))$:
- $a^*$
- $a^+ xa^*$
- $a^+ xa^* ya^*$
- $a^+ xa^* ya^* za^*$
- $a^+ xa^* ya^* za^*$
- $a^+ xa^* wa^*$
Comparing the equivalences

**Proposition**

If $T S_1 \sim_Q T S_2$, then $T S_1 \sim_{io} T S_2$.

**The converse does not hold**

\[
\begin{align*}
&\text{Tracks}(T S_1): \\
&\quad \varepsilon \\
&\quad ax \\
&\quad axx \\
&\quad axaxx \\
&\text{IOBeh}(T S_1): \\
&\quad (\varepsilon, \varepsilon) \\
&\quad (a^n, x) \text{ for } n \geq 1 \\
&\quad (a^n, x^2) \text{ for } n \geq 1 \\
&\quad (a^n, x^3) \text{ for } n \geq 2
\end{align*}
\]
Comparing the equivalences

Proposition

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

The converse does not hold

$TS_2$

Tracks($TS_2$):
- $\epsilon$
- $ax$
- $axx$
- $axxx$

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

$axxx \oplus axaxx$
Quiescent traces (Tretmans)

Empty and blocked quiescent traces

- $w \in L(Q(TS))$ is an **empty quiescent trace** if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, \varepsilon, \varepsilon)$ with $r \in I$ and $s$ quiescent in $TS$.
  
  We denote by $L_{\delta}^{\text{empty}}(Q(TS))$ the empty quiescent traces of $Q(TS)$.

- $w \in L(Q(TS))$ is a **blocked quiescent trace** if there is a run $(r, \varepsilon, \varepsilon) \xrightarrow{w} (s, a\sigma_i, \varepsilon)$ with $r \in I$ and in $TS$, $s$ quiescent and $s \xrightarrow{\alpha}$.  
  
  We denote by $L_{\delta}^{\text{block}}(Q(TS))$ the blocked quiescent traces of $Q(TS)$.

Proposition

$$L_{\delta}(Q(TS)) = L_{\delta}^{\text{empty}}(Q(TS)) \cup L_{\delta}^{\text{block}}(Q(TS))$$
Quiescent traces (Tretmans)

Example

\[ L^\text{empty}_\delta(TS_1): \]
\[ \varepsilon \]
\[ ax \]
\[ axy \]
\[ axayz \]
\[ axyaz \]
\[ aaxyz \]

\[ L^\text{block}_\delta(TS_1): \]
\[ a^+xya^+za^+ \]
\[ a^+xa^+yza^+ \]
\[ aa^+xyza^+ \]
\[ a^+xa^+ya^+za^* \]
\[ \ldots \]
\[ \triangleright\text{-upper closure of } axyaza \]

Lemma

\[ L^\text{block}_\delta(Q(TS)) = \{ w \in \Sigma^* \mid \exists r \xrightarrow{w'} s \text{ in } TS \text{ with } r \in I, s \text{ quiescent, and } \exists a \in \Sigma_i \text{ such that } s \xrightarrow{a} \text{ and } w'a \triangleright w \}. \]
Strict ape relation (Tretmans)

Strict ape relation for the queue semantics

We denote $\left|\mathord{@}\right|$ the reflexive and transitive closure of the relation postponing an output action: $w_1xaw_2 \mathord{@} w_1axw_2$.

Lemma

$$L_{\delta}^{\text{empty}}(Q(TS)) = \{ w \in \Sigma^* | \exists r \xrightarrow{w'} s \text{ in } TS \text{ with } r \in I, s \text{ quiescent}, w' \left|\mathord{@}\right| w \}$$

Example

$L_{\delta}^{\text{empty}}(TS_1)$: $\left|\mathord{@}\right|$-upper closure of
- $\epsilon$
- $ax$
- $axy$
- $axya$

$L_{\delta}^{\text{block}}(TS_1)$: $\left|\mathord{@}\right|$-upper closure of
- $axya$
- $axya$ $\left|\mathord{@}\right|$ $axayz$
- $axya$ $\left|\mathord{@}\right|$ $aaaxyz$
Undecidability of $\sim Q$

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 \( f, g : A^+ \rightarrow B^+ \) 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

$L_{\delta}^{\text{block}}(M_1) = L_{\delta}^{\text{block}}(M_2) = \emptyset$.

Lemma

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

Lemma

- $L_{\delta}^{\emptyset}(S_0)$ is the $|@|$-upper closure of $A^+B^+\$.
- Let $u \in A^+$ and $v \in B^+$. Then, $uv\$ \in L_{\delta}^{\emptyset}(S_f)$ 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

Introduction

Input/Output semantics

IO-Blocks semantics

Queue semantics (Tretman)

Conclusion
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.