Testing Nondeterministic and Probabilistic Processes

Matthew Hennessy

(joint work with Yuxin Deng, Rocco DeNicola, Rob van Glabbeek, Carroll Morgan, Chenyi Zhang)

BASICS, Shanghai October 09
Outline

Background  why bother ?

Testing theory

Testing nondeterministic processes

Testing Probabilistic and nondeterministic processes
Outline

Background  why bother?

Testing theory

Testing nondeterministic processes

Testing Probabilistic and nondeterministic processes
Background

Goal: Specification and proof methodologies for probabilistic concurrent systems

Nondeterminism + Probability – why necessary?

- “Nondeterminism” intrinsic to specification development à la CSP
  - underspecified components expressed using “nondeterminism”

\[
\text{COMP} \sqcap \text{OPTION} \leq \text{COMP} \\
\text{underspecified} \quad \text{more specified}
\]

- Analysis of concurrent systems requires “nondeterminism”

\( \sqcap \) - internal choice of CSP
Goal: Specification and proof methodologies for probabilistic concurrent systems

Nondeterminism + Probability – why necessary?

- “Nondeterminism” intrinsic to specification development à la CSP
  - underspecified components expressed using “nondeterminism”

\[
\text{COMP} \sqcap \text{OPTION} \leq \text{COMP} \\
\text{underspecified} \quad \text{more specified}
\]

- Analysis of concurrent systems requires “nondeterminism”
Background

Goal: Specification and proof methodologies for probabilistic concurrent systems

Nondeterminism + Probability – why necessary?

- “Nondeterminism” intrinsic to specification development à la CSP
  - underspecified components expressed using “nondeterminism”

$$\text{COMP } \square \text{ OPTION } \leq \text{ COMP}$$

underspecified more specified

- Analysis of concurrent systems requires “nondeterminism”

\(\square\) - internal choice of CSP
Background

Goal: Specification and proof methodologies for probabilistic concurrent systems

Nondeterminism + Probability – why necessary?

- “Nondeterminism” intrinsic to specification development à la CSP
  - underspecified components expressed using “nondeterminism”

\[
\text{COMP } \sqcap \text{OPTION } \leq \text{COMP}
\]

underspecified  more specified

- Analysis of concurrent systems requires “nondeterminism”

\(\sqcap\) - internal choice of CSP
Analysis of concurrent systems

**Sys1:**

\[ \text{Sys1} \leftarrow (\text{new } s)(A \mid Sw) \]
\[ A \leftarrow up.U + s?down.D \]
\[ Sw \leftarrow s!stop \]

**Sys2:**

\[ \text{Sys2} \leftarrow (\text{new } s)(B \mid Sw) \]
\[ B \leftarrow s?(up.U + down.D) + s?down.D \]
\[ Sw \leftarrow s!stop \]
Analysis of concurrent systems

In CSP theory:

\[ \text{Sys1} \approx \text{Sys2} \]

semantically equivalent

Both equivalent to the nondeterministic

\[(\text{up.}U + \text{down.}D) \sqcap \text{down.}D\]

concurrency = nondeterminism + interleaving

probabilistic concurrency = probability + nondeterminism + interleaving
Analysis of concurrent systems

In CSP theory:

\[ \text{Sys1} \approx \text{Sys2} \]

semantically equivalent

Both equivalent to the nondeterministic

\[ (up.U + down.D) \sqcap down.D \]

concurrency = nondeterminism + interleaving

probabilistic concurrency = probability + nondeterminism + interleaving
Analysis of concurrent systems

In CSP theory:

\[ \text{Sys}1 \approx \text{Sys}2 \]

semantically equivalent

Both equivalent to the nondeterministic

\[ (up.U + down.D) \cap down.D \]

concurrency = nondeterminism + interleaving

probabilistic concurrency = probability + nondeterminism + interleaving
Analysis of concurrent systems

In CSP theory:

\[ \text{Sys1} \approx \text{Sys2} \]

semantically equivalent

Both equivalent to the nondeterministic

\[(up.U + down.D) \cap down.D\]

concurrency = nondeterminism + interleaving

probabilistic concurrency = probability + nondeterminism + interleaving
Outline

Background why bother?

Testing theory

Testing nondeterministic processes

Testing Probabilistic and nondeterministic processes
Testing scenario

- a set of processes $\mathcal{Proc}$
- a set of tests $\mathcal{T}$
- a set of outcomes $\mathcal{O}$

Apply: $\mathcal{T} \times \mathcal{Proc} \rightarrow \mathcal{P}^+(\mathcal{O})$ – the non-empty set of possible results of applying a test to a process

Comparing sets of outcomes:

- $\mathcal{O}_1 \sqsubseteq_{Ho} \mathcal{O}_2$ if for every $o_1 \in \mathcal{O}_1$ there exists some $o_2 \in \mathcal{O}_2$ such that $o_1 \leq o_2$

- $\mathcal{O}_1 \sqsubseteq_{Sm} \mathcal{O}_2$ if for every $o_2 \in \mathcal{O}_2$ there exists some $o_1 \in \mathcal{O}_1$ such that $o_1 \leq o_2$

$o_1 \leq o_2$: means $o_2$ is as least as good as $o_1$
Testing scenario

- a set of processes $\mathcal{P}roc$
- a set of tests $\mathcal{T}$
- a set of outcomes $\mathcal{O}$
- $Apply : \mathcal{T} \times \mathcal{P}roc \rightarrow \mathcal{P}^+ (\mathcal{O})$ – the non-empty set of possible results of applying a test to a process

Comparing sets of outcomes:

- $\mathcal{O}_1 \sqsubseteq_{Ho} \mathcal{O}_2$ if for every $o_1 \in \mathcal{O}_1$ there exists some $o_2 \in \mathcal{O}_2$ such that $o_1 \leq o_2$
- $\mathcal{O}_1 \sqsubseteq_{Sm} \mathcal{O}_2$ if for every $o_2 \in \mathcal{O}_2$ there exists some $o_1 \in \mathcal{O}_1$ such that $o_1 \leq o_2$

$o_1 \leq o_2$ : means $o_2$ is as least as good as $o_1$
Testing preorders

- \( P \sqsubseteq_{\text{may}} Q \) if \( \text{Apply}(T, P) \sqsubseteq_{\text{Ho}} \text{Apply}(T, Q) \) for every test \( T \)
- \( P \sqsubseteq_{\text{must}} Q \) if \( \text{Apply}(T, P) \sqsubseteq_{\text{Sm}} \text{Apply}(T, Q) \) for every test \( T \)

Standard testing:
Use as outcomes \( \mathcal{O} = \{ \top, \bot \} \) with \( \bot \leq \top \)

Comparisons:
Possible outcome sets: \( \{ \bot \} \quad \{ \bot, \top \} \quad \{ \top \} \)
- May: \( \{ \bot \} <_{\text{Ho}} \{ \bot, \top \} =_{\text{Ho}} \{ \top \} \)
- Must: \( \{ \bot \} =_{\text{Sm}} \{ \bot, \top \} <_{\text{Sm}} \{ \top \} \)
Testing preorders

- $P \sqsubseteq_{\text{may}} Q$ if $\text{Apply}(T, P) \sqsubseteq_{\text{Ho}} \text{Apply}(T, Q)$ for every test $T$
- $P \sqsubseteq_{\text{must}} Q$ if $\text{Apply}(T, P) \sqsubseteq_{\text{Sm}} \text{Apply}(T, Q)$ for every test $T$

Standard testing:
Use as outcomes $\mathcal{O} = \{\top, \bot\}$ with $\bot \leq \top$

Comparisons:
Possible outcome sets: $\{\bot\}$  $\{\bot, \top\}$  $\{\top\}$
- May: $\{\bot\} \leq_{\text{Ho}} \{\bot, \top\} =_{\text{Ho}} \{\top\}$
- Must: $\{\bot\} =_{\text{Sm}} \{\bot, \top\} <_{\text{Sm}} \{\top\}$
Testing preorders

- \( P \sqsubseteq_{\text{may}} Q \) if \( \text{Apply}(T, P) \sqsubseteq_{\text{Ho}} \text{Apply}(T, Q) \) for every test \( T \)
- \( P \sqsubseteq_{\text{must}} Q \) if \( \text{Apply}(T, P) \sqsubseteq_{\text{Sm}} \text{Apply}(T, Q) \) for every test \( T \)

Standard testing:
Use as outcomes \( \mathcal{O} = \{ \top, \bot \} \) with \( \bot \leq \top \)

Comparisons:
Possible outcome sets: \( \{ \bot \} \quad \{ \bot, \top \} \quad \{ \top \} \)
- May: \( \{ \bot \} <_{\text{Ho}} \{ \bot, \top \} =_{\text{Ho}} \{ \top \} \)
- Must: \( \{ \bot \} =_{\text{Sm}} \{ \bot, \top \} <_{\text{Sm}} \{ \top \} \)
Testing preorders

- $P \sqsubseteq_{\text{may}} Q$ if $\text{Apply}(T, P) \sqsubseteq_{\text{Ho}} \text{Apply}(T, Q)$ for every test $T$
- $P \sqsubseteq_{\text{must}} Q$ if $\text{Apply}(T, P) \sqsubseteq_{\text{Sm}} \text{Apply}(T, Q)$ for every test $T$

Probabilistic testing:
Use as $\mathcal{O}$ the unit interval $[0, 1]$
Intuition: with $0 \leq p \leq q \leq 1$, passing a test with probability $q$
better than passing with probability $p$

Comparisons:

- May: $O_1 \sqsubseteq_{\text{Ho}} O_2$ is every possibility $p \in O_1$ can be improved
  on by some $q \in O_2$
- Must: $O_1 \sqsubseteq_{\text{Sm}} O_2$ if every possibility $q \in O_2$ is an
  improvement on some $p \in O_1$
Testing preorders

- \( P \sqsubseteq_{\text{may}} Q \) if \( \text{Apply}(T, P) \sqsubseteq_{\text{Ho}} \text{Apply}(T, Q) \) for every test \( T \)
- \( P \sqsubseteq_{\text{must}} Q \) if \( \text{Apply}(T, P) \sqsubseteq_{\text{Sm}} \text{Apply}(T, Q) \) for every test \( T \)

Probabilistic testing:
Use as \( O \) the unit interval \([0, 1]\)
Intuition: with \( 0 \leq p \leq q \leq 1 \), passing a test with probability \( q \) better than passing with probability \( p \)

Comparisons:

- May: \( O_1 \sqsubseteq_{\text{Ho}} O_2 \) is every possibility \( p \in O_1 \) can be improved on by some \( q \in O_2 \)
- Must: \( O_1 \sqsubseteq_{\text{Sm}} O_2 \) if every possibility \( q \in O_2 \) is an improvement on some \( p \in O_1 \)
Outline

Background why bother?

Testing theory

Testing nondeterministic processes

Testing Probabilistic and nondeterministic processes
Nondeterministic processes

Intensional semantics:

A process is a state in an LTS

Labelled Transition Systems:

\[ \langle S, \text{Act}_\tau, \rightarrow \rangle \]

- \( S \) - states
- \( \rightarrow \subseteq S \times \text{Act}_\tau \times S \)

\( s_1 \xrightarrow{\mu} s_2 \): process \( s_1 \) can perform action \( \mu \) and continue as \( s_2 \)

\( s_1 \xrightarrow{\tau} s_2 \) special internal action
Nondeterministic processes

Intensional semantics:

A process is a state in an LTS

Labelled Transition Systems:

\[ \langle S, \text{Act}_\tau, \rightarrow \rangle \]

- \( S \) - states
- \( \rightarrow \subseteq S \times \text{Act}_\tau \times S \)

\( s_1 \xrightarrow{\mu} s_2 \): process \( s_1 \) can perform action \( \mu \) and continue as \( s_2 \)

\( s_1 \xrightarrow{\tau} s_2 \) special internal action
Process calculi: Syntax for LTSs

Example process calculus CCS:

- **0** Do nothing
- **μ. P** Perform μ then act as P
- **P | Q** Run P and Q in parallel . . . communicating via complementary actions
- **P + Q** Nondeterministic choice between P and Q
- **recursive definitions** D ⇐ P

Actions

P ←−μ→ Q defined inductively

lots of other process calculi
Process calculi: Syntax for LTSs

Example process calculus CCS:

- **0** Do nothing
- **μ. P** Perform μ then act as P
- **P | Q** Run P and Q in parallel ... communicating via complementary actions
- **P + Q** Nondeterministic choice between P and Q
- recursive definitions \( D \iff P \)

Actions

\( P \xrightarrow{\mu} Q \) defined inductively

lots of other process calculi
Process calculi: Syntax for LTSs

Example process calculus CCS:

- **0** Do nothing
- **μ.P** Perform μ then act as P
- **P | Q** Run P and Q in parallel ... communicating via complementary actions
- **P + Q** Nondeterministic choice between P and Q
- **recursive definitions** $D \leftarrow P$

**Actions**

$P \xrightarrow{\mu} Q$ defined inductively

lots of other process calculi
Process calculi: Syntax for LTSs

Example process calculus CCS:

- $0$ Do nothing
- $\mu. P$ Perform $\mu$ then act as $P$
- $P | Q$ Run $P$ and $Q$ in parallel ... communicating via complementary actions
- $P + Q$ Nondeterministic choice between $P$ and $Q$
- recursive definitions $D \leftarrow P$

Actions

$P \xrightarrow{\mu} Q$ defined inductively

lots of other process calculi
Testing nondeterministic processes

Tests:
Any process which may contain new report success action/state $\omega$

\[ T \leftarrow \overline{a}.\omega + \overline{b}.T + \overline{c}.0: \]

- requests an $a$ action . . .
- after an arbitrary number of $b$ actions . . .
- without doing any $c$ action

Applying test $T$ to process $P$:

- Run the combined process $(T \mid P)$
- Each execution succeeds or fails
- Each execution contributes $\top$ or $\bot$ to $\text{Apply}(T, P)$

Nondeterministic $(T \mid P)$ resolved to a set of deterministic executions
Testing nondeterministic processes

Tests:
Any process which may contain new *report success* action/state $\omega$
$T \leftarrow \overline{a}.\omega + \overline{b}.T + \overline{c}.0$:

- requests an $a$ action . . .
- after an arbitrary number of $b$ actions . . .
- without doing any $c$ action

Applying test $T$ to process $P$:

- Run the combined process $(T | P)$
- Each *execution* succeeds or fails
- Each *execution* contributes $\top$ or $\bot$ to $Apply(T, P)$

Nondeterministic $(T | P)$ resolved to a set of deterministic executions
Testing nondeterministic processes

Tests:
Any process which may contain new report success action/state \( \omega \nondeterm \Rightarrow \alpha.\omega + \beta.T + \gamma.0 \):

- requests an \( \alpha \) action . . .
- after an arbitrary number of \( \beta \) actions . . .
- without doing any \( \gamma \) action

Applying test \( T \) to process \( P \):

- Run the combined process \( (T \mid P) \)
- Each execution succeeds or fails
- Each execution contributes \( \top \) or \( \bot \) to \( \text{Apply}(T, P) \)

Nondeterministic \( (T \mid P) \) resolved to a set of deterministic executions
Example

Test: \( T \leftarrow \overline{a}.\omega + \overline{b}.T + \overline{c}.0 \)  
Process: \( P \leftarrow b.(a.Q + b.P) \)

Deterministic executions:

\[
\begin{align*}
T | P \xrightarrow{\tau} b \xrightarrow{\tau} a \omega | - & \quad T \\
T | P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} a \omega | - & \quad T \\
T | P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} a \omega | - & \quad T \\
T | P \quad \ldots & \quad T \\
T | P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} b \ldots \ldots \xrightarrow{\tau} b \ldots & \quad \bot
\end{align*}
\]

Result:

\[ \text{Apply}(T, P) = \{ \bot, T \} \]
Example

Test: $T \leftarrow \overline{a}.\omega + \overline{b}.T + \overline{c}.0$

Process: $P \leftarrow b.(a.Q + b.P)$

Deterministic executions:

- $T | P \xrightarrow{\tau} b \xrightarrow{\tau} a \omega | -$  
  $T | P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} a \omega | -$  
  $T | P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} a \omega | -$  
  $T | P \ldots$  
  $T | P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} b \ldots \ldots \xrightarrow{\tau} b \ldots$  

Result:

$Apply(T, P) = \{ \bot, \top \}$
Example

Test: $T \leftarrow \overline{a}.\omega + \overline{b}.T + \overline{c}.0$

Process: $P \leftarrow b.(a.Q + b.P)$

Deterministic executions:

$T \mid P \xrightarrow{\tau} b \xrightarrow{\tau} a \omega \mid \_ \quad \top$

$T \mid P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} a \omega \mid \_ \quad \top$

$T \mid P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} a \omega \mid \_ \quad \top$

$T \mid P \ldots \quad \top$

$T \mid P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} b \ldots \ldots \xrightarrow{\tau} b \ldots \quad \bot$

Result:

$\text{Apply}(T, P) = \{\bot, \top\}$
Example

Test: $T \leftarrow \overline{a}.\omega + \overline{b}.T + \overline{c}.0$

Process: $P \leftarrow b.(a.Q + b.P)$

Deterministic executions:

$T \mid P \xrightarrow{\tau} b \xrightarrow{\tau} a \omega \mid _-$

$\top$

$T \mid P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} a \omega \mid _-$

$\top$

$T \mid P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} a \omega \mid _-$

$\top$

$T \mid P \ldots$

$\top$

$T \mid P \xrightarrow{\tau} b \xrightarrow{\tau} b \xrightarrow{\tau} b \ldots \ldots \xrightarrow{\tau} b \ldots$

$\bot$

Result:

$\text{Apply}(T, P) = \{ \bot, \top \}$
May-testing nondeterministic processes

Divergence not important:

\[ P + \tau \cdot \tau^\infty \simeq_{\text{may}} P \]

Choice not important:
May-testing nondeterministic processes

Divergence not important:

$$P + \tau.\tau^\infty \cong_{\text{may}} P$$

Choice not important:
Must testing nondeterministic processes

Divergence catastrophic:

\[ P + \tau \cdot \tau^\infty \sim_{\text{must}} \tau^\infty \]

Internal choices not very important:
Must testing nondeterministic processes

Divergence catastrophic:

$$P + \tau.\tau^\infty \sim_{\text{must}} \tau^\infty$$

Internal choices not very important:
Characterising nondeterministic processes

Ingredients:

- **Traces:** $a_1a_2\ldots a_n$ in $\text{Traces}(P)$ whenever

  $P \xrightarrow{\tau} \ast \xrightarrow{a_1} \xrightarrow{\tau} \ast \ldots \ldots \xrightarrow{\tau} \ast \xrightarrow{a_n} \xrightarrow{\tau} \ast \ xrightarrow{\tau} \ast \ P'$

- **Divergences/convergences:** $P \Downarrow$ whenever there is no infinite execution

  $P \xrightarrow{\tau} \xrightarrow{\tau} \ldots \ldots \xrightarrow{\tau} \ldots \ldots$

- **Failures/Acceptances:** $P \text{ acc } A$ whenever $P \Downarrow$ and

  $P \xrightarrow{\tau} \ast \ P'$ implies $P' \xrightarrow{a}$ for some $a$ in $A$
Characterising nondeterministic processes

Ingredients:

- **Traces:** $a_1a_2\ldots a_n$ in $\text{Traces}(P)$ whenever
  \[ P \xrightarrow{\tau}^* a_1 \xrightarrow{\tau}^* \ldots \xrightarrow{\tau}^* a_n \xrightarrow{\tau}^* P' \]

- **Divergences/convergences:** $P \Downarrow$ whenever there is no infinite execution
  \[ P \xrightarrow{\tau} \xrightarrow{\tau} \ldots \xrightarrow{\tau} \ldots \]

- **Failures/Acceptances:** $P \text{ acc } A$ whenever $P \Downarrow$ and
  \[ P \xrightarrow{\tau}^* P' \text{ implies } P' \xrightarrow{a} \text{ for some } a \text{ in } A \]
Outline

Background why bother?

Testing theory

Testing nondeterministic processes

Testing Probabilistic and nondeterministic processes
Probabilistic and nondeterministic processes

Intensional semantics:

A process is a distribution in an pLTS

Probabilistic Labelled Transition Systems:

\[ \langle S, \text{Act}_\tau, \longrightarrow \rangle \]

- \( S \) - states
- \( \longrightarrow \subseteq S \times \text{Act}_\tau \times \mathcal{D}(S) \)

\( \mathcal{D}(S) \): Mappings \( \Delta : S \rightarrow [0, 1] \) with \( \sum_{s \in S} \Delta(s) = 1 \)

\( s_1 \xrightarrow{\mu} \Delta \): process \( s_1 \)
- can perform action \( \mu \)
- with probability \( \Delta(s_2) \) it continues as process \( s_2 \)
Probabilistic and nondeterministic processes

Intensional semantics:

A process is a distribution in an pLTS

Probabilistic Labelled Transition Systems:

\( \langle S, \text{Act}_\tau, \rightarrow \rangle \)

- \( S \) - states
- \( \rightarrow \subseteq S \times \text{Act}_\tau \times \mathcal{D}(S) \)

\( \mathcal{D}(S) \): Mappings \( \Delta : S \rightarrow [0, 1] \) with \( \sum_{s \in S} \Delta(s) = 1 \)

\( s_1 \xrightarrow{\mu} \Delta \): process \( s_1 \)
- can perform action \( \mu \)
- with probability \( \Delta(s_2) \) it continues as process \( s_2 \)
Probabilistic and nondeterministic processes

Intensional semantics:

A process is a distribution in an pLTS.

Probabilistic Labelled Transition Systems:

\[ \langle S, \text{Act}_\tau, \rightarrow \rangle \]

- \( S \) - states
- \( \rightarrow \subseteq S \times \text{Act}_\tau \times \mathcal{D}(S) \)

\( \mathcal{D}(S) \): Mappings \( \Delta : S \rightarrow [0, 1] \) with \( \sum_{s \in S} \Delta(s) = 1 \)

\( s_1 \xrightarrow{\mu} \Delta \): process \( s_1 \)

- can perform action \( \mu \)
- with probability \( \Delta(s_2) \) it continues as process \( s_2 \)
Example probabilistic processes

What is the probability of action $a$ happening?
Example probabilistic processes

What is the probability of action $a$ happening?
Probabilistic process calculi: Syntax for pLTSs

Example process calculus pCCS:

State terms $S$, $T$:

- $0$
- $\mu.P$
- $S | T$
- $S + T$
- recursive definitions

Process terms $P$, $Q$:

- $S$
- $P \oplus Q$ probabilistic choice between $P$ and $Q$

Actions
$s \xrightarrow{\mu} \Delta$ defined inductively

process terms are distributions over states
Probabilistic process calculi: Syntax for pLTSs

Example process calculus pCCS:

State terms $S$, $T$:

- $0$
- $\mu . P$
- $S \parallel T$ $S + T$
- recursive definitions

Process terms $P$, $Q$:

- $S$
- $P \oplus Q$ probabilistic choice between $P$ and $Q$

Actions

$s \xrightarrow{\mu} \Delta$ defined inductively

process terms are distributions over states
Probabilistic process calculi: Syntax for pLTSs

Example process calculus pCCS:

State terms $S$, $T$:

- $0$
- $\mu. P$
- $S \ | \ T$
- $S + T$
- recursive definitions

Process terms $P$, $Q$:

- $S$
- $P \oplus Q$ probabilistic choice between $P$ and $Q$

Actions

$s \xrightarrow{\mu} \Delta$ defined inductively

process terms are distributions over states
Probabilistic process calculi: Syntax for pLTSs

Example process calculus pCCS:

State terms $S$, $T$:

- $0$
- $\mu P$
- $S \mid T$
- $S + T$
- recursive definitions

Process terms $P$, $Q$:

- $S$
- $P \oplus Q$ probabilistic choice between $P$ and $Q$

Actions

$s \xrightarrow{\mu} \Delta$ defined inductively

process terms are distributions over states
Probabilistic process calculi: Syntax for pLTSs

Example process calculus pCCS:

State terms $S, T$:

- $0$
- $\mu.P$
- $S \mid T$
- $S + T$
- recursive definitions

Process terms $P, Q$:

- $S$
- $P \oplus Q$ probabilistic choice between $P$ and $Q$

Actions

$s \xrightarrow{\mu} \Delta$ defined inductively

process terms are distributions over states
Testing probabilistic processes

Tests:
Any (prob...) process which may contain report success
action/state ω
\(a.\omega \frac{1}{4} \oplus (b + c.\omega)\):
- 25% of time requests an \(a\) action
- 75% requests a \(c\) action
- 75% requires that \(b\) is not possible in a must test

Applying test \(T\) to process \(P\):
- Execute the combined process \((T \mid P)\)
- Each execution contributes some probability \(p\) to \(Apply(T, P)\)
- Each execution is a deterministic resolution of \((T \mid P)\)
Testing probabilistic processes

Tests:
Any (prob...) process which may contain *report success*
action/state $\omega$
$a.\omega \frac{1}{4} \oplus (b + c.\omega)$:
- 25% of time requests an $a$ action
- 75% requests a $c$ action
- 75% requires that $b$ is not possible in a must test

Applying test $T$ to process $P$:
- Execute the combined process $(T \mid P)$
- Each execution contributes some probability $p$ to $\text{Apply}(T, P)$
- Each execution is a deterministic resolution of $(T \mid P)$
Executing probabilistic nondeterministic processes \((T \mid P)\)

- Choice points occur during an execution
  - choices are made
    - statically
    - or dynamically
  - choices are made
    - by schedulers
    - adversaries
    - policies

Executions:
- give deterministic behaviour - but may be probabilistic
- contribute a probability to \(Apply(T, P)\)
Executing probabilistic nondeterministic processes \((T \mid P)\)

- Choice points occur during an execution
- choices are made
  - statically
  - or dynamically
- choices are made
  - by schedulers
  - adversaries
  - policies

Executions:
- give deterministic behaviour - but may be probabilistic
- contribute a probability to \(Apply(T, P)\)
Executing probabilistic nondeterministic processes \((T \mid P)\)

- Choice points occur during an execution
- Choices are made
  - Statically
  - Or dynamically
- Choices are made
  - By schedulers
  - Adversaries
  - Policies

Executions:

- Give deterministic behaviour - but may be probabilistic
- Contribute a probability to \(Apply(T, P)\)
Executing probabilistic nondeterministic processes \((T \mid P)\)

- Choice points occur during an execution
- choices are made
  - statically
  - or dynamically
- choices are made
  - by schedulers
  - adversaries
  - policies

Executions:
- give deterministic behaviour - but may be probabilistic
- contribute a probability to \(\text{Apply}(T, P)\)
Example of executions

Static Policies:

\[ pp_1 : s_1 \rightarrow s_0 \]
\[ pp_2 : s_1 \rightarrow t_{bd} \]

Possible results:

Using \( pp_1 \):
\[
\frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4} + \left( \frac{3}{4} \right)^2 \cdot \frac{1}{4} + \ldots + \ldots = 1
\]

Using \( pp_2 \):
\[
\frac{1}{4} + \frac{3}{4} \cdot 0 = \frac{1}{4}
\]
Example of executions

Static Policies:

- \( pp_1 : s_1 \rightarrow s_0 \)
- \( pp_2 : s_1 \rightarrow t_{bd} \)

Possible results:

Using \( pp_1 \):
\[
\frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4} + \left(\frac{3}{4}\right)^2 \cdot \frac{1}{4} + \ldots + \ldots = 1
\]

Using \( pp_2 \):
\[
\frac{1}{4} + \frac{3}{4} \cdot 0 = \frac{1}{4}
\]
Example of executions

Static Policies:

\[ pp_1 : s_1 \rightarrow s_0 \]
\[ pp_2 : s_1 \rightarrow t_{bd} \]

Possible results:

Using \( pp_1 \):
\[
\frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4} + \left(\frac{3}{4}\right)^2 \cdot \frac{1}{4} + \ldots + \ldots = 1
\]

Using \( pp_2 \):
\[
\frac{1}{4} + \frac{3}{4} \cdot 0 = \frac{1}{4}
\]
More executions

Arbitrary policies: combinations of

\[
\begin{align*}
pp_1 & : s_1 \rightarrow s_0 \\
pp_2 & : s_1 \rightarrow t_{bd}
\end{align*}
\]

Possible results:

Using \(pp_1\): \(\frac{1}{4}\)

Using \(pp_2\): \(\frac{1}{4}\)

In general: \(p \cdot 1 + (1 - p) \cdot \frac{1}{4}\) for some \(0 \leq p \leq 1\)
More executions

Arbitrary policies: combinations of

\[ pp_1 : s_1 \xrightarrow{} s_0 \]
\[ pp_2 : s_1 \xrightarrow{} t_{bd} \]

Possible results:

Using \( pp_1 \):
\[ 1 \]

Using \( pp_2 \):
\[ \frac{1}{4} \]

In general:
\[ p \cdot 1 + (1 - p) \cdot \frac{1}{4} \text{ for some } 0 \leq p \leq 1 \]
Formalising executions I

From pLTSS to LTSs

\[ \Delta \xrightarrow{\mu} \Theta \]

- \( \Delta \) represents a cloud of possible process states
- each possible state must be able to perform \( \mu \)
- all possible residuals combine to \( \Theta \)

Examples:

- \((a.b + a.c) \oplus a.d \xrightarrow{a} b \oplus d\)
- \((a.b + a.c) \oplus a.d \xrightarrow{a} (b \oplus c) \oplus d\)
- \((a.b + a.c) \oplus a.d \xrightarrow{a} (b \oplus c) \oplus d\)
- \((\tau.a + \tau.b) \oplus (\tau.a + \tau.c) \xrightarrow{\tau} a \oplus (b \oplus c)\)
Formalising executions I

From pLTSS to LTSs

\[ \Delta \xrightarrow{\mu} \Theta \]

- \( \Delta \) represents a cloud of possible process states
- each possible state must be able to perform \( \mu \)
- all possible residuals combine to \( \Theta \)

Examples:

- \((a.b + a.c) \frac{1}{2} \oplus a.d \xrightarrow{a} b \frac{1}{2} \oplus d\)
- \((a.b + a.c) \frac{1}{2} \oplus a.d \xrightarrow{a} (b \frac{1}{2} \oplus c) \frac{1}{2} \oplus d\)
- \((a.b + a.c) \frac{1}{2} \oplus a.d \xrightarrow{a} (b_p \oplus c) \frac{1}{2} \oplus d\)
- \((\tau.a + \tau.b) \frac{1}{2} \oplus (\tau.a + \tau.c) \xrightarrow{\tau} a \frac{1}{2} \oplus (b \frac{1}{2} \oplus c)\)
Formalising executions I

From pLTSS to LTSs

\[ \Delta \xrightarrow{\mu} \Theta \]

- \( \Delta \) represents a cloud of possible process states
- each possible state must be able to perform \( \mu \)
- all possible residuals combine to \( \Theta \)

Examples:

- \((a \cdot b + a \cdot c) \frac{1}{2} \oplus a \cdot d \xrightarrow{a} b \frac{1}{2} \oplus d\)
- \((a \cdot b + a \cdot c) \frac{1}{2} \oplus a \cdot d \xrightarrow{a} (b \frac{1}{2} \oplus c) \frac{1}{2} \oplus d\)
- \((a \cdot b + a \cdot c) \frac{1}{2} \oplus a \cdot d \xrightarrow{a} (b_p \oplus c) \frac{1}{2} \oplus d\)
- \((\tau \cdot a + \tau \cdot b) \frac{1}{2} \oplus (\tau \cdot a + \tau \cdot c) \xrightarrow{\tau} a \frac{1}{2} \oplus (b \frac{1}{2} \oplus c)\)
Formalising executions I
From pLTSS to LTSs

\[ \Delta \xrightarrow{\mu} \Theta \]

- \(\Delta\) represents a cloud of possible process states
- each possible state must be able to perform \(\mu\)
- all possible residuals combine to \(\Theta\)

Examples:

- \((a.b + a.c) \frac{1}{2} \oplus a.d \xrightarrow{a} b \frac{1}{2} \oplus d\)
- \((a.b + a.c) \frac{1}{2} \oplus a.d \xrightarrow{a} (b \frac{1}{2} \oplus c) \frac{1}{2} \oplus d\)
- \((a.b + a.c) \frac{1}{2} \oplus a.d \xrightarrow{a} (b_p \oplus c) \frac{1}{2} \oplus d\)
- \((\tau.a + \tau.b) \frac{1}{2} \oplus (\tau.a + \tau.c) \xrightarrow{\tau} a \frac{1}{2} \oplus (b \frac{1}{2} \oplus c)\)
Formalising executions I

From pLTSS to LTSs

$$\Delta \xrightarrow{\mu} \Theta$$

- $\Delta$ represents a cloud of possible process states
- each possible state must be able to perform $\mu$
- all possible residuals combine to $\Theta$

Examples:

- $$(a.b + a.c) \frac{1}{2} \oplus a.d \xrightarrow{a} b \frac{1}{2} \oplus d$$
- $$(a.b + a.c) \frac{1}{2} \oplus a.d \xrightarrow{a} (b \frac{1}{2} \oplus c) \frac{1}{2} \oplus d$$
- $$(a.b + a.c) \frac{1}{2} \oplus a.d \xrightarrow{a} (b \oplus c) \frac{1}{2} \oplus d$$
- $$(\tau.a + \tau.b) \frac{1}{2} \oplus (\tau.a + \tau.c) \xrightarrow{\tau} a \frac{1}{2} \oplus (b \frac{1}{2} \oplus c)$$
From pLTSS to LTSs: formally

\[ \Delta \xrightarrow{\mu} \Theta \]

whenever

- \( \Delta = \sum_{i \in I} p_i \cdot s_i \), \( I \) a finite index set
- For each \( i \in I \) there is a distribution \( \Theta_i \) s.t. \( s_i \xrightarrow{\mu} \Theta_i \)
- \( \Theta = \sum_{i \in I} p_i \cdot \Theta_i \)
- \( \sum_{i \in I} p_i = 1 \)

Note: in decomposition \( \sum_{i \in I} p_i \cdot s_i \) states \( s_i \) are not necessarily unique
From pLTSS to LTSs: formally

\[ \Delta \xrightarrow{\mu} \Theta \]

whenever

- \[ \Delta = \sum_{i \in I} p_i \cdot s_i \], \( I \) a finite index set
- For each \( i \in I \) there is a distribution \( \Theta_i \) s.t. \( s_i \xrightarrow{\mu} \Theta_i \)
- \[ \Theta = \sum_{i \in I} p_i \cdot \Theta_i \]
- \[ \sum_{i \in I} p_i = 1 \]

Note: in decomposition \( \sum_{i \in I} p_i \cdot s_i \) states \( s_i \) are not necessarily unique
Formalising executions II

Executing \((T | P)\) to \(\Theta\):

\[
\begin{align*}
(T | P) & \quad \xRightarrow{} \quad \Theta \\
\Delta_0 & \quad \xrightarrow{\tau} \quad \Delta_0^{\rightarrow} + \Delta_1^{\rightarrow} + \Delta_1^{\text{stop}} \\
\cdots & \quad \xrightarrow{\tau} \quad \cdots \\
\Delta_k & \quad \xrightarrow{\tau} \quad \Delta_{(k+1)}^{\rightarrow} + \Delta_{(k+1)}^{\text{stop}} \\
\cdots & \quad \xrightarrow{\tau} \quad \cdots \\
\cdots & \quad \xrightarrow{\tau} \quad \cdots \\
\end{align*}
\]

Total:

\[
\Theta = \sum_{k=0}^{\infty} \Delta_k^{\text{stop}}
\]

- \(\Delta^{\text{stop}}\): all states in \(\Delta\) which
  - are successful \(s \xrightarrow{\omega}\)
  - or are stuck \(s \xrightarrow{\not\tau}\)

- \(\Delta^{\rightarrow}\): all other states, which can proceed \(s \xrightarrow{\tau}\)

note: subdistributions
Formalising executions II

Executing \((T \mid P)\) to \(\Theta\):

\[
\begin{align*}
(T \mid P) & \quad \Rightarrow \quad \Theta \\
\Delta_0 & \quad \Rightarrow \quad \Delta_0 + \\
\Delta_1 & \quad \Rightarrow \quad \Delta_1 + \\
\Delta_{k} & \quad \Rightarrow \quad \Delta_{(k+1)} + \\
\text{Total:} & \quad \Theta = \sum_{k=0}^{\infty} \Delta_{k}^\text{stop}
\end{align*}
\]

- \(\Delta_{\text{stop}}\): all states in \(\Delta\) which
  - are successful \(s \xrightarrow{\omega}\)
  - or are stuck \(s \nrightarrow\)
- \(\Delta_{\rightarrow}\): all other states, which can proceed \(s \xrightarrow{\tau}\)

Note: subdistributions
Formalising executions II

Executing \((T | P)\) to \(\Theta\):

\[
\begin{align*}
(T | P) & \overset{\tau}{\longrightarrow} \Delta_0^+ + \Delta_1^+ \\
\Delta_0 & \overset{\tau}{\longrightarrow} \Delta_1 \\
\ldots & \\
\Delta_k & \overset{\tau}{\longrightarrow} \Delta_{(k+1)}^+ + \Delta_{(k+1)}^{\text{stop}} \\
\ldots & \\
\ldots & \\
\end{align*}
\]

Total: \(\Theta = \sum_{k=0}^{\infty} \Delta_k^{\text{stop}}\)

- \(\Delta^{\text{stop}}\): all states in \(\Delta\) which
  - are successful \(s \xrightarrow{\omega}\)
  - or are stuck \(s \xrightarrow{\tau}\)
- \(\Delta\rightleftarrows\): all other states, which can proceed \(s \xrightarrow{\tau}\)

note: subdistributions
Formalising executions II

Executing \((T | P)\) to \(\Theta\):

\[
\begin{align*}
(T | P) & \quad \xrightarrow{\tau} \quad (T | P) \\
\Delta_0 & \quad \xrightarrow{\tau} \quad \Delta_1 \\
& \quad \vdots \quad \vdots \\
\Delta_k & \quad \xrightarrow{\tau} \quad \Delta_{(k+1)} \\
& \quad \vdots \quad \vdots
\end{align*}
\]

Total:

\[
\Theta = \sum_{k=0}^{\infty} \Delta_{k}^{\text{stop}}
\]

- \(\Delta_{\text{stop}}\): all states in \(\Delta\) which
  - are successful \(s \xrightarrow{\omega}\)
  - or are stuck \(s \xrightarrow{\n}\)
- \(\Delta\): all other states, which can proceed \(s \xrightarrow{\tau}\)

Note: subdistributions
Formalising executions II

Executing \((T \mid P)\) to \(\Theta\):

\[
(T \mid P) \xrightarrow{\tau} (T \mid P)
\]

\[
\Delta_0 \xrightarrow{\tau} \Delta_0 + \Delta_0^{\text{stop}}
\quad \Delta_1 \xrightarrow{\tau} \Delta_1 + \Delta_1^{\text{stop}}
\quad \ldots
\quad \Delta_k \xrightarrow{\tau} \Delta(k+1) + \Delta(k+1)^{\text{stop}}
\quad \ldots
\]

Total:

\[
\Theta = \sum_{k=0}^{\infty} \Delta_k^{\text{stop}}
\]

**\(\Delta^{\text{stop}}\):** all states in \(\Delta\) which

- are successful \(s \xrightarrow{\omega}\)
- or are stuck \(s \xrightarrow{\tau}\)

**\(\Delta^{\rightarrow}\):** all other states, which can proceed \(s \xrightarrow{\tau}\)

Note: subdistributions
Applying tests to processes: \( Apply(T, P) \)

- find all executions from \( (T | P) \):
  \[
  (T | P) \xrightarrow{} \Theta
  \]

- calculate contribution of each \( \Theta \)

Contribution of \( \Theta \):

- all states in \( \Theta \) are successful \( s \xrightarrow{\omega} \) or stuck \( s \not\xrightarrow{\tau} \)
- \[ V(\Theta) = \sum \{ \Theta(s) \mid s \xrightarrow{\omega} \} \] weight of success

\[
Apply(T, P) = \{ V(\Theta) \mid (T | P) \xrightarrow{} \Theta \}
\]

Problem: set of executions \( \{ \Theta \mid (T | P) \xrightarrow{} \Theta \} \) difficult to calculate
Applying tests to processes: \( \text{Apply}(T, P) \)

- find all executions from \((T \mid P)\):

  \[
  (T \mid P) \xrightarrow{\Theta}
  \]

- calculate contribution of each \(\Theta\)

### Contribution of \(\Theta\):

- all states in \(\Theta\) are successful \(s \xrightarrow{\omega}\) or stuck \(s \not\xrightarrow{\tau}\)

- \(\forall(\Theta) = \sum\{ \Theta(s) \mid s \xrightarrow{\omega} \}\) weight of success

\[
\text{Apply}(T, P) = \{ \forall(\Theta) \mid (T \mid P) \xrightarrow{\Theta} \}
\]

Problem: set of executions \(\{ \Theta \mid (T \mid P) \xrightarrow{\Theta} \}\) difficult to calculate
Applying tests to processes: \( \text{Apply}(T, P) \)

- find all executions from \((T \mid P)\):
  \[(T \mid P) \rightarrow \Theta\]
- calculate contribution of each \(\Theta\)

**Contribution of \(\Theta\):**

- all states in \(\Theta\) are successful \(s \xrightarrow{\omega}\) or stuck \(s \xrightarrow{\tau}\)
- \(\mathbb{V}(\Theta) = \sum\{ \Theta(s) \mid s \xrightarrow{\omega} \}\) weight of success

\[\text{Apply}(T, P) = \{ \mathbb{V}(\Theta) \mid (T \mid P) \rightarrow \Theta \}\]

**Problem:** set of executions \(\{ \Theta \mid (T \mid P) \rightarrow \Theta \}\) difficult to calculate
Applying tests to processes: \( \text{Apply}(T, P) \)

- find all executions from \((T \mid P)\):
  \[
  (T \mid P) \xrightarrow{} \Theta
  \]
- calculate contribution of each \(\Theta\)

Contribution of \(\Theta\):

- all states in \(\Theta\) are successful \(s \xrightarrow{\omega}\) or stuck \(s \xrightarrow{\tau}\)
- \(\nabla(\Theta) = \sum\{ \Theta(s) \mid s \xrightarrow{\omega} \}\) weight of success

\[
\text{Apply}(T, P) = \{ \nabla(\Theta) \mid (T \mid P) \xrightarrow{} \Theta \}
\]

Problem: set of executions \(\{ \Theta \mid (T \mid P) \xrightarrow{} \Theta \}\) difficult to calculate
Alternative strategy

Recall:

- \( P \sqsubseteq_{p_{\text{may}}} Q \) if \( \text{Apply}(T, P) \sqsubseteq_{\text{Ho}} \text{Apply}(T, Q) \) for every test \( T \)

- \( P \sqsubseteq_{p_{\text{must}}} Q \) if \( \text{Apply}(T, P) \sqsubseteq_{\text{Sm}} \text{Apply}(T, Q) \) for every test \( T \)

Maybe:

- \( P \sqsubseteq_{p_{\text{may}}} Q \) if \( \sup(\text{Apply}(T, P)) \leq \sup(\text{Apply}(T, Q)) \) for every test \( T \)

- \( P \sqsubseteq_{p_{\text{must}}} Q \) if \( \inf(\text{Apply}(T, P)) \sqsubseteq_{\text{Sm}} \inf(\text{Apply}(T, Q)) \) for every test \( T \)

Strategy:

- calculate \( \inf(\text{Apply}(T, \neg)) \) and \( \sup(\text{Apply}(T, \neg)) \) directly

- do not calculate the entire set \( \text{Apply}(T, \neg) \)
Alternative strategy

Recall:

- \( P \sqsubseteq_{p\text{may}} Q \) if \( \text{Apply}(T, P) \sqsubseteq_{\text{Ho}} \text{Apply}(T, Q) \) for every test \( T \)
- \( P \sqsubseteq_{p\text{must}} Q \) if \( \text{Apply}(T, P) \sqsubseteq_{\text{Sm}} \text{Apply}(T, Q) \) for every test \( T \)

Maybe:

- \( P \sqsubseteq_{p\text{may}} Q \) if \( \sup(\text{Apply}(T, P)) \leq \sup(\text{Apply}(T, Q)) \) for every test \( T \)
- \( P \sqsubseteq_{p\text{must}} Q \) if \( \inf(\text{Apply}(T, P)) \sqsubseteq_{\text{Sm}} \inf(\text{Apply}(T, Q)) \) for every test \( T \)

Strategy:

- calculate \( \inf(\text{Apply}(T, -)) \) and \( \sup(\text{Apply}(T, -)) \) directly
- do not calculate the entire set \( \text{Apply}(T, -) \)
Alternative strategy

Recall:

- $P \sqsubseteq_{p_{\text{may}}} Q$ if $\text{Apply}(T, P) \subseteq_{\text{Ho}} \text{Apply}(T, Q)$ for every test $T$
- $P \sqsubseteq_{p_{\text{must}}} Q$ if $\text{Apply}(T, P) \subseteq_{\text{Sm}} \text{Apply}(T, Q)$ for every test $T$

Maybe:

- $P \sqsubseteq_{p_{\text{may}}} Q$ if $\sup(\text{Apply}(T, P)) \leq \sup(\text{Apply}(T, Q))$ for every test $T$
- $P \sqsubseteq_{p_{\text{must}}} Q$ if $\inf(\text{Apply}(T, P)) \subseteq_{\text{Sm}} \inf(\text{Apply}(T, Q))$ for every test $T$

Strategy:

- calculate $\inf(\text{Apply}(T, -))$ and $\sup(\text{Apply}(T, -))$ directly
- do not calculate the entire set $\text{Apply}(T, -)$
Example: Calculating the \( \text{sup} \)

\[
s_1 = \max\{r_{\text{sup}}, t_{bd}\}
\]

\[
s_2 = t_{gd}
\]

\[
t_{bd} = 0
\]

\[
t_{gd} = 1
\]

\[
\sup(\text{Apply}(T, P)) \text{ is least solution: } r_{\text{sup}} = 1
\]

\[
r_{\text{sup}} = \frac{3}{4} \cdot s_1 + \frac{1}{4} \cdot s_2
\]

\[
s_1 = \max\{r_{\text{sup}}, t_{bd}\}
\]

\[
s_2 = t_{gd}
\]

\[
t_{bd} = 0
\]

\[
t_{gd} = 1
\]
Example: Calculating the sup

\[
\begin{align*}
s_1 &= \frac{3}{4} s_1 + \frac{1}{4} s_2 \\
s_2 &= t_{gd}
\end{align*}
\]

\[
s_1 = \max \{ r_{sup}, t_{bd} \}
\]

\[
s_2 = t_{gd}
\]

\[
t_{bd} = 0
\]

\[
t_{gd} = 1
\]

\[
sup(Apply(T, P)) \text{ is least solution: } r_{sup} = 1
\]
Example: Calculating the sup

\[ r_{sup} = \frac{3}{4} \cdot s_1 + \frac{1}{4} \cdot s_2 \]

\[ s_1 = \max\{ r_{sup}, t_{bd} \} \]

\[ s_2 = t_{gd} \]

\[ t_{bd} = 0 \]

\[ t_{gd} = 1 \]

\[ \sup(Appl\{T, P\}) \text{ is least solution: } r_{sup} = 1 \]
Example: Calculating the \( \inf \)

\(
\begin{align*}
\inf(\text{Apply}(T, P)) \text{ is least solution:} & \quad r_{\inf} = \frac{1}{4} \\
\text{inf-equation:} & \quad r_{\inf} = \frac{3}{4} \cdot s_1 + \frac{1}{4} \cdot s_2 \\
& \quad s_1 = \text{min}\{r_{\inf}, t_{bd}\} \\
& \quad s_2 = t_{gd} \\
& \quad t_{bd} = 0 \\
& \quad t_{gd} = 1
\end{align*}
\)
Example: Calculating the inf

inf-equation:

\[ r_{inf} = \frac{3}{4} \cdot s_1 + \frac{1}{4} \cdot s_2 \]

\[ s_1 = \min\{r_{inf}, t_{bd}\} \]
\[ s_2 = t_{gd} \]
\[ t_{bd} = 0 \]
\[ t_{gd} = 1 \]

inf(\text{Apply}(T, P)) is least solution:

\[ r_{inf} = \frac{1}{4} \]
Example: Calculating the inf

\[
inf(Apply(T,P)) \text{ is least solution: } r_{inf} = \frac{1}{4}
\]

\[
r_{inf} = \frac{3}{4} \cdot s_1 + \frac{1}{4} \cdot s_2
\]

\[
s_1 = min\{r_{inf}, t_{bd}\}
\]

\[
s_2 = t_{gd}
\]

\[
t_{bd} = 0
\]

\[
t_{gd} = 1
\]
Finitary pLTSs

Whenever

- set of states are finite
- set of actions are finite

In a finitary pLTS:

- execution sets \( \{ \Theta \mid \text{Apply}(T, P) \Rightarrow \Theta \} \) are closed
- \( P \sqsubseteq_{\text{may}} Q \) iff \( \inf(\text{Apply}(T, P)) \leq \inf(\text{Apply}(T, Q)) \) for every test \( T \)
- \( P \sqsubseteq_{\text{must}} Q \) iff \( \sup(\text{Apply}(T, P)) \sqsubseteq_{\text{sm}} \sup(\text{Apply}(T, Q)) \) for every test \( T \)
- \( \inf(\text{Apply}(T, -)) \) is least solution of inf-equation
- \( \sup(\text{Apply}(T, -)) \) is least solution of sup-equation
Finitary pLTSs

Whenever

- set of states are finite
- set of actions are finite

In a finitary pLTS:

- execution sets \( \{ \Theta \mid \text{Apply}(T, P) \rightarrowarrow \Theta \} \) are closed
- \( P \sqsubseteq_{\text{may}} Q \) iff \( \inf(\text{Apply}(T, P)) \leq \inf(\text{Apply}(T, Q)) \) for every test \( T \)
- \( P \sqsubseteq_{\text{must}} Q \) iff \( \sup(\text{Apply}(T, P)) \sqsubseteq_{\text{Sm}} \sup(\text{Apply}(T, Q)) \) for every test \( T \)
- \( \inf(\text{Apply}(T, -)) \) is least solution of inf-equation
- \( \sup(\text{Apply}(T, -)) \) is least solution of sup-equation
Finitary pLTSs

Whenever

- set of states are finite
- set of actions are finite

In a finitary pLTS:

- execution sets \{ \Theta \mid \text{Apply}(T, P) \Rightarrow \Theta \} are closed
- \( P \sqsubseteq_{\text{may}} Q \) iff \( \inf(\text{Apply}(T, P)) \leq \inf(\text{Apply}(T, Q)) \) for every test \( T \)
- \( P \sqsubseteq_{\text{must}} Q \) iff \( \sup(\text{Apply}(T, P)) \sqsubseteq_{\text{Sm}} \sup(\text{Apply}(T, Q)) \) for every test \( T \)
- \( \inf(\text{Apply}(T, -)) \) is least solution of inf-equation
- \( \sup(\text{Apply}(T, -)) \) is least solution of sup-equation
Finitary pLTSs

Whenever

- set of states are finite
- set of actions are finite

In a finitary pLTS:

- execution sets \( \{ \Theta \mid Apply(T, P) \rightarrow \Theta \} \) are closed
- \( P \sqsubseteq_{\text{may}} Q \) iff \( \inf(Apply(T, P)) \leq \inf(Apply(T, Q)) \) for every test \( T \)
- \( P \sqsubseteq_{\text{must}} Q \) iff \( \sup(Apply(T, P)) \sqsubseteq_{\text{Sm}} \sup(Apply(T, Q)) \) for every test \( T \)
- \( \inf(Apply(T, \neg \)) \) is least solution of inf-equation
- \( \sup(Apply(T, \neg \)) \) is least solution of sup-equation
Example

\[ r_1 = a.(τ.b + τ.c) \quad r_2 = a.b + a.c \quad T = \overline{a}.(b.ω \frac{1}{2} \oplus \overline{c}.ω) \]

\[ \text{Apply}(T,r_1) = \begin{cases} \inf : & 0 \\ \sup : & 1 \end{cases} \]

\[ \text{Apply}(T,r_2) = \begin{cases} \inf : & \frac{1}{2} \\ \sup : & \frac{1}{2} \end{cases} \]

So choice points do matter:

\[ r_1 \not\sim_{\text{pmay}} r_2 \quad r_1 \not\sim_{\text{pmust}} r_2 \]
Example

\[ r_1 = a.(\tau.b + \tau.c) \quad r_2 = a.b + a.c \quad T = \overline{a}.(\overline{b}.\omega + \overline{c}.\omega) \]

Apply(\(T, r_1\)) = \begin{cases} 
\inf : & 0 \\
\sup : & 1 
\end{cases}

Apply(\(T, r_2\)) = \begin{cases} 
\inf : & \frac{1}{2} \\
\sup : & \frac{1}{2} 
\end{cases}

So choice points do matter: \( r_1 \not\sim_{\text{pmay}} r_2 \quad r_1 \not\sim_{\text{pmust}} r_2 \)
Example

\[
\text{Apply}(\overline{a}.\omega, P) = \begin{cases} 
\inf: & \frac{1}{2} \\
\sup: & 1 
\end{cases}
\]

\[
P \simeq_{p\text{may}} a.0
\]

\[
P \subseteq_{p\text{must}} a.0
\]

\[
a.0 \nsubseteq_{p\text{must}} P
\]
Example

Apply(\bar{a}.\omega, P) = \begin{cases} 
\inf : \frac{1}{2} \\
\sup : 1 
\end{cases}

\[ P \sim_{\text{p\text{-}may}} a.\mathbf{0} \quad P \sqsubseteq_{\text{p\text{-}must}} a.\mathbf{0} \quad a.\mathbf{0} \nsubseteq_{\text{p\text{-}must}} P \]
Example

\[
\text{Apply}(\overline{a}.\omega, P) = \begin{cases} 
\inf : & \frac{1}{2} \\
\sup : & 1
\end{cases}
\]

\[P \simeq_{\text{pmay}} a.0\]

\[P \sqsubseteq_{\text{pmust}} a.0\]

\[a.0 \not\sqsubseteq_{\text{pmust}} P\]
Example

\[
\text{Apply}(\overline{a}.\omega, P) = \begin{cases}
\inf & : \frac{1}{2} \\
\sup & : 1
\end{cases}
\]

\[P \simeq_{\text{pmay}} a.0\]

\[P \sqsubseteq_{\text{pmust}} a.0\]

\[a.0 \not\sqsubseteq_{\text{pmust}} P\]
Coming up:

Reasoning techniques for probabilistic processes

Are these distinguishable by any test?
Coming up:

Reasoning techniques for probabilistic processes

Are these distinguishable by any test?