Concurrency theory

Equivalences

Francesco Zappa Nardelli

INRIA Rocquencourt, MOSCOVA research team

francesco.zappa_nardelli@inria.fr

together with

Frank Valencia (INRIA Futurs) Catuscia Palamidessi (INRIA Futurs) Roberto Amadio (PPS)

How can we tell the difference?

Which of the (one-shot) vending machines do you want in your office?

$$
\begin{aligned}
V_1 &= \text{coin}.\overline{\text{coffe}} + \text{coin}.\overline{\text{tea}} \\
V_2 &= \text{coin}.(\overline{\text{coffe}} + \overline{\text{tea}}) \\
V_3 &= \text{coin} + \text{coin}.(\overline{\text{coffe}} + \overline{\text{tea}}) \\
V_4 &= \text{coin}.\overline{\text{coffe}} + \text{coin}.(\overline{\text{coffe}} + \overline{\text{tea}}) \\
V_5 &= \text{coin}.(\overline{\text{coffe}} + \overline{\text{tea}}) + \text{coin}.(\overline{\text{coffe}} + \overline{\text{tea}})
\end{aligned}
$$

Chosing requires a deeper understanding of *nondeterminism*.

Answer 1: any will do

Intuition: all the automatas accepts a coin and give back a tea or a coffe.

Definition $\sigma \in \mathtt{Act}^*$ is a *trace* of a process P, denoted $P \xrightarrow{\sigma}$, if

- $\sigma = \epsilon$, or
- $\sigma = \mu.\sigma'$ and there exists Q such that $P \xrightarrow{\mu} Q$ and σ' is a trace for Q.

Definition Let $\mathcal{T}(P)$ be the set of all the traces of P. Two processes P and Q are *trace equivalent*, denoted $P =_{\mathcal{T}} Q$, if $\mathcal{T}(P) = \mathcal{T}(Q)$.

Example: the processes $V_1 \ldots V_5$ are all trace equivalent.

Answer 2: any will do, except V_3

Intuition: V_3 might "eat" a coin.

Definition $\sigma \in \texttt{Act}^*$ is a *completed trace* of a process P if $P \xrightarrow{\sigma} \mathbf{0}$.

Definition Let $\mathcal{CT}(P)$ be the set of all the completed traces of P. Two processes P and Q are *completed trace equivalent*, denoted $P =_{\mathcal{CT}} Q$, if $\mathcal{CT}(P) = \mathcal{CT}(Q)$.

Example: the processes V_1, V_2, V_4, V_5 are completed trace equivalent. They are not completed trace equivalent to V_3.

Answer 3: in V_1 something looks fishy

Intuition: V_1 does not let me the choice after accepting a coin.

Definition $(\sigma, X) \in \mathtt{Act}^* \times \mathcal{P}(\mathtt{Act})$ is a *failure pair* of a process P if there is a process Q such that $P \xrightarrow{\sigma} Q$ and forall $\mu \in X$, $Q \not\xrightarrow{\mu}$.

Definition Let $\mathcal{F}(P)$ be the set of all the failure pairs of P. Two processes P and Q are *failures equivalent*, denoted $P =_{\mathcal{F}} Q$, if $\mathcal{F}(P) = \mathcal{F}(Q)$.

Exercise: which of $V_1, \ldots, V_5$ are failures equivalent?

Answer 4: let's play a game

1. We choose two automatas (I play with $P = V_i$, you with $Q = V_j$).

2. If I cannot play a transition $P \xrightarrow{\mu} P'$ you win.
   Otherwise I play a transition $P \xrightarrow{\mu} P'$.

   (a) If you cannot reply with a transition $Q \xrightarrow{\mu} Q'$ for some Q' I win,

   (b) Otherwise you play $Q \xrightarrow{\mu} Q'$, and we go back to 2. with $P = P'$ and $Q = Q'$.

If you can reliably win, then we say that V_j *simulates* V_i.

Simulation, formally

Definition: a simulation is a binary relation $\mathcal{R}$ on the set of processes such that for all P, Q, if $P \mathrel{\mathcal{R}} Q$ then

$$\forall \mu, P',\ P \xrightarrow{\mu} P' \ \Rightarrow\ \exists Q',\ Q \xrightarrow{\mu} Q' \text{ and } P' \mathrel{\mathcal{R}} Q'\ .$$

We say that Q simulates P if there exists a simulation $\mathcal{R}$ such that $P \mathrel{\mathcal{R}} Q$.

Exercise: are there simulations among $V_1, \ldots, V_5$?

Question: why did we introduce this notion?

Answer 5: let's play another game

1. We choose two automatas $P = V_i$ and $Q = V_j$.

2. I choose either P or Q.
   (in what follows suppose I chosed P – similarly for Q).

3. If I cannot play a transition $P \xrightarrow{\mu} P'$ you win.
   Otherwise I play a transition $P \xrightarrow{\mu} P'$.
   (a) If you cannot reply with a transition $Q \xrightarrow{\mu} Q'$ for some Q' I win,
   (b) Otherwise you play $Q \xrightarrow{\mu} Q'$, and we go back to 2. with $P = P'$ and $Q = Q'$.

If you can reliably win, then we say that V_i and V_j are *bisimilar*.

Bisimulation, formally

Definition: a bisimulation is a binary relation $\mathcal{R}$ on the set of processes such that for all P, Q, if $P \mathrel{\mathcal{R}} Q$ then

$$- \forall \mu, P',\ P \xrightarrow{\mu} P' \ \Rightarrow\ \exists Q',\ Q \xrightarrow{\mu} Q' \text{ and } P' \mathrel{\mathcal{R}} Q' \ ;$$

$$- \forall \mu, Q',\ Q \xrightarrow{\mu} Q' \ \Rightarrow\ \exists P',\ P \xrightarrow{\mu} P' \text{ and } P' \mathrel{\mathcal{R}} Q' \ .$$

We say that P and Q are bisimilar, denoted $P \sim Q$, if there exists a bisimulation $\mathcal{R}$ such that $P \mathrel{\mathcal{R}} Q$.

The relation $\sim$, defined as the union of all the bisimulations, is the *largest bisimulation* and (more on this later).

Bisimulation, ctd.

Exercise: are there bisimulations among $V_1, \ldots, V_5$?

Notation: $R^{-1} = \{(Q, P) : P \mathcal{R} Q\}$.

Alternative definition for bisimulation: a bisimulation is a binary relation $\mathcal{R}$ on the set of processes such that $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations.

Remark: P simulates Q and Q simulates P *does not imply* that P and Q are bisimilar!

Exercise: find an example to validate the remark above.

Properties of bisimilarity

Theorem: Bisimilarity $\sim$ is an equivalence relation.

Exercise: Prove the theorem above.

Question: does bisimilarity exists?

To answer to this question, we need some mathematics...

Monotonous functions

A function $f : D \mapsto E$, where D, E are partial orders, is *monotonous* if

$$\forall\, x, y \quad x \leq y \;\Rightarrow\; f(x) \leq f(y)$$

Given a monotonous $f : D \mapsto D$:

- a *prefixpoint* of f is a point x such that $f(x) \leq x$;
- a *postfixpoint* of f is a point x such that $x \leq f(x)$;
- a *fixpoint* of f is a point x such that $x = f(x)$;

Monotonous functions, ctd.

Any monotonous function $G : \mathcal{P}(A) \mapsto \mathcal{P}(A)$ has

- a *least* prefixpoint, which is moreover a *fixpoint*, and
- a *greatest* postfixpoint, which is moreover a *fixpoint*.

They are respectively (Knaster-Tarsky):

$$\begin{array}{rcl} \mathrm{lfp}(G) & = & \bigcap\{R : G(R) \subseteq R\} \\ \mathrm{gfp}(G) & = & \bigcup\{R : R \subseteq G(R)\} \end{array}$$

Inductively defined sets via rules

A rule instance comprises its premises and a conclusion:

$$\frac{x_1, x_2, \ldots}{y} \qquad\qquad \text{also written } (X, y)$$

Intuition: if the premises $x_1, x_2, \ldots$ are in the set being defined, then so is the conclusion y. We look for the least set with this property.

Inductively defined sets

Given a set A, let K be a set of rules each of the form (X, y) for $X \subseteq A$ and $x \in A$.

Definition: We say a set Q is K*-closed* iff

$$\forall (X, y) \in K,\ (X \subseteq Q \Rightarrow y \in Q)\ .$$

Now K defines a *monotonous* operator $G : \mathcal{P}(A) \to \mathcal{P}(A)$:

$$G_K(R) = \{y \in A : \exists (X, y) \in K \text{ and } X \subseteq R\}\ .$$

Remark: the prefixpoints of G_K are exactly the K-closed sets.

The *inductively defined set* of K is the least K-closed set, or:

$$\mathrm{lfp}(G) = \bigcap \{Q : Q \text{ is } K\text{-closed }\} = \bigcap \{Q : G_K(Q) \subseteq Q\}\ .$$

Coinductively defined sets

Given a set A, let K be a set of rules each of the form (X, y) for $X \subseteq A$ and $x \in A$.

Definition: We say a set Q is K*-closed backward* iff

$$\forall x \in Q,\ \exists (Y, x) \in K\ Y \subseteq R$$

Remark: the postfixpoints of G_K are exactly the K-closed backward sets.

The *coinductively defined set* of K is the greatest K-closed backward set, or:

$$\mathrm{gfp}(G) = \bigcup \{Q : Q \text{ is } K\text{-closed backward}\} = \bigcup \{Q : Q \subseteq G_K(Q)\}\ .$$

Bisimilarity as coinductively defined set

Bisimulation is defined by a set of rules: take K to be the set of all

$$\frac{\{(P', f(\mu, P')) : P \xrightarrow{\mu} P'\} \cup \{(g(\mu, Q'), Q') : Q \xrightarrow{\mu} Q'\}}{(P, Q)}$$

where

- f is any function mapping each pair (μ, P') such that $P \xrightarrow{\mu} P'$ to a process $f(\mu, P')$ such that $Q \xrightarrow{\mu} f(\mu, P')$;
- g is any function mapping each pair (μ, Q') such that $Q \xrightarrow{\mu} Q'$ to a process $g(\mu, Q')$ such that $P \xrightarrow{\mu} g(\mu, Q')$.

Bisimilarity is a congruence

Define $\hat{\sim}$ inductively by the following rules:

$$\frac{P \sim Q}{P \mathrel{\hat{\sim}} Q} \qquad \frac{P \mathrel{\hat{\sim}} Q}{Q \mathrel{\hat{\sim}} P} \qquad \frac{P \mathrel{\hat{\sim}} Q \quad Q \mathrel{\hat{\sim}} R}{P \mathrel{\hat{\sim}} R}$$

$$\frac{\forall i \in I \;\; P_i \mathrel{\hat{\sim}} Q_i}{\Sigma_{i \in I} \mu_i . P_i \mathrel{\hat{\sim}} \Sigma_{i \in I} \mu_i . Q_i} \qquad \frac{P_1 \mathrel{\hat{\sim}} Q_1 \quad P_2 \mathrel{\hat{\sim}} Q_2}{P_1 \mid\mid P_2 \mathrel{\hat{\sim}} Q_1 \mid\mid Q_2} \qquad \frac{P \mathrel{\hat{\sim}} Q}{(\boldsymbol{\nu} a)P \mathrel{\hat{\sim}} (\boldsymbol{\nu} a)Q}$$

By construction $\sim \subseteq \hat{\sim}$ and $\hat{\sim}$ is a congruence. It is enough to show that $\hat{\sim}$ is a bisimulation (which implies $\hat{\sim} \subseteq \sim$).

Bisimulation is a congruence, ctd.

Proof by rule induction. We detail the case $P_1 \mid\mid P_2 \hat{\sim} Q_1 \mid\mid Q_2$.

- (backward) decomposition phase: if $P_1 \mid\mid P_2 \xrightarrow{\mu} P'$, then $P' = P'_1 \mid\mid P'_2$ and three cases may occur, corresponding to the three rules for parallel composition is the labelled operational semantics. We only consider the synchronisation case. If $P_1 \xrightarrow{a} P'_1$ and $P_2 \xrightarrow{\overline{a}} P'_2$, then
- by induction there exists Q'_1 such that $Q_1 \xrightarrow{a} Q'_1$ and $P'_1 \hat{\sim} Q'_1$, and there exists Q'_2 such that $Q_2 \xrightarrow{a} Q'_2$ and $P'_2 \hat{\sim} Q'_2$.
- Hence (forward phase) we have $Q_1 \mid\mid Q_2 \xrightarrow{\tau} Q'_1 \mid\mid Q'_2$ and $P'_1 \mid\mid P'_2 \hat{\sim} Q'_1 \mid\mid Q'_2$.

Bisimulation is a congruence, ctd. (recursion)

Proposition: for any process S with free variables in $\vec{K}$:

$$\forall \vec{Q}, \vec{Q}' \; (\vec{Q} \sim \vec{Q}' \;\Rightarrow\; S[\; \vec{K} \to \vec{Q} \;] \sim S[\; \vec{K} \to \vec{Q} \;])$$

Exercise: prove it. Hint: the proof is by induction on the size of S. The non-recursion cases follow by congruence. For the recursive definition case $S = \mathrm{let}\ \vec{L} = \vec{P}\ \mathrm{in}\ L_j$, the trick is to unfold...

Exercises

1. Show that structural congruence $\equiv$ implies bisimilarity $\sim$.

2. Consider the processes $H(a)$ and $K(a)$ defined by $H(x) = x.H(x)$ and $K(x) = x.K(x) \mid\mid x.K(x)$. Are they bisimilar?

3. Prove that $P + P \sim P$ but (in general) $P \mid\mid P \not\sim P$.

4. Which is the smallest bisimulation?

Proof techniques for bisimulation

A bisimulation up-to $\sim$ is a relation $\mathcal{R}$ such that for all P, Q:

$$P \mathrel{\mathcal{R}} Q \quad \Rightarrow \quad \forall \mu, P' \; (P \xrightarrow{\mu} P' \Rightarrow \exists Q' \; Q \xrightarrow{\mu} Q' \text{ and } P' \sim\mathcal{R}\sim Q')$$

and conversely.

Exercise: prove that if $\mathcal{R}$ is a strong bisimulation up-to $\sim$, then $\mathcal{R} \subseteq \sim$.

Hence to show $P \sim Q$ it is enough to find a bisimulation up-to $\sim$ such that $P \mathrel{\mathcal{R}} Q$.

Semaphores, again

$$\begin{array}{rcl} \texttt{Sem} & = & P.\texttt{Sem}' \\ \texttt{Sem}' & = & V.\texttt{Sem} \end{array}$$

$$\begin{array}{rcl} \texttt{Sem}^0 & = & P.\texttt{Sem}^1 \\ \texttt{Sem}^1 & = & P.\texttt{Sem}^2 + V.\texttt{Sem}^0 \\ \texttt{Sem}^2 & = & P.\texttt{Sem}^3 + V.\texttt{Sem}^1 \\ \texttt{Sem}^3 & = & V.\texttt{Sem}^2 \end{array}$$

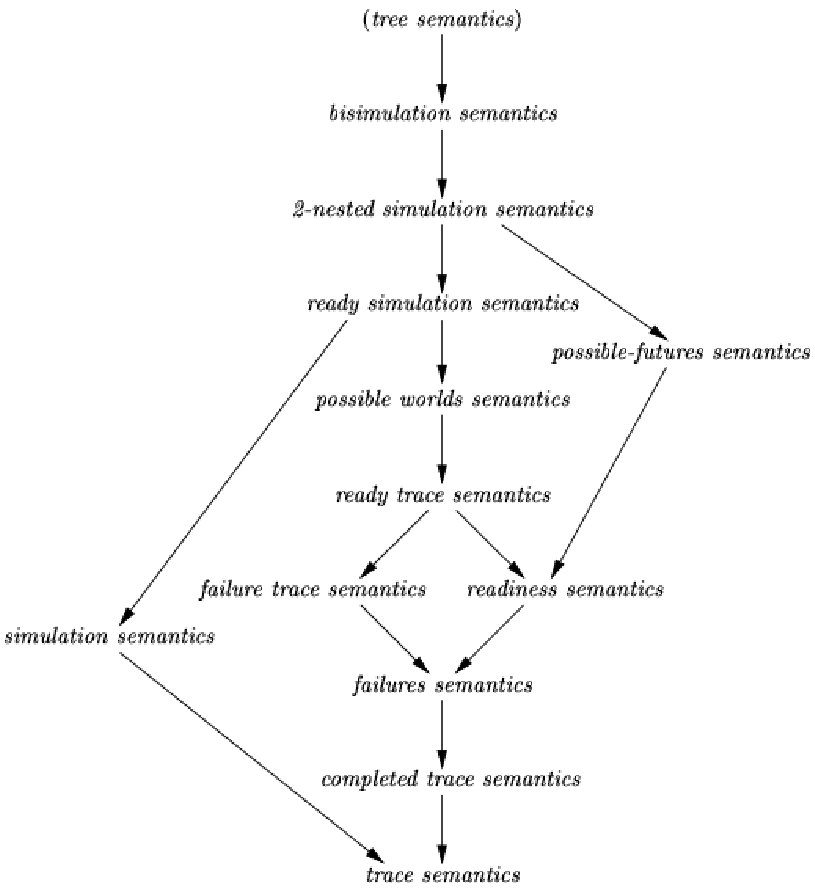
Using the up to $\sim$ proof technique, we can show that

$$\texttt{Sem} \bigg|\bigg| \texttt{Sem} \bigg|\bigg| \texttt{Sem} \ \sim\ \texttt{Sem}^0$$

by exhibiting the simple relation:

$$\begin{array}{l} \{\ (\texttt{Sem} \ || \ \texttt{Sem} \ || \ \texttt{Sem}\ ,\ \texttt{Sem}^0)\ ;\ (\texttt{Sem}' \ || \ \texttt{Sem} \ || \ \texttt{Sem}\ ,\ \texttt{Sem}^1)\ ; \\ \quad (\texttt{Sem}' \ || \ \texttt{Sem}' \ || \ \texttt{Sem}\ ,\ \texttt{Sem}^2)\ ;\ (\texttt{Sem}' \ || \ \texttt{Sem}' \ || \ \texttt{Sem}'\ ,\ \texttt{Sem}^3)\ \} \end{array}$$

The big plan



From "The linear time–branching time spectrum", Glaabbek

Think about which semantics you are interested in!

Suppose that an high-level language L is compiled into a target language T using cryptographic logs to ensure some security property in presence of attackers.

Theorem: Given a trace ϕ in the target language T,

1. there exists a corresponding trace in the source language L, or
2. the trace ϕ can be extended into a trace that ends with the discovery of the attacker.

In this case, reasoning with traces captures exactly the security property we are interested in!

Think about which semantics you are interested in! (ctd.)

Consider

$$P = a.b.c + a.b \qquad\qquad Q = a.(b.c + b)$$

These processes are

- equated by the *failure semantics*,
- but are told apart by *bisimilarity*.

We might want to consider these processes as equivalent, as in both cases it is the process that choses if the action c will be available, not the environment.

However, the proof techniques associated with bisimilarity are a big win with respect to failure semantics, and we are ready to take bisimilarity as our reference equivalence.

In general, we are interested in “weaker” semantics

Consider:

$$P = a.\overline{b}.P + \overline{b}.a.P \qquad\qquad Q = a.\overline{b}.\tau.Q + \overline{b}.a.\tau.Q$$

We might want to identify these processes, for instance if the τ reductions of Q correspond to *uninteresting implementation details*.

Question: Is it possible to give a semantics to CCS that abstracts from internal reductions?

One step backward: comparing paths

For any LTS, one can change $\mathtt{Act}$ to $\mathtt{Act}^*$ (words of actions), setting

$$P \xrightarrow{s} Q \text{ if } \begin{cases} s = \mu_1, \ldots, \mu_n \text{ and} \\ \exists P_1, \ldots, P_n \; (P_n = Q \text{ and } P \xrightarrow{\mu_1} P_1 \ldots \xrightarrow{\mu_n} P_n) \end{cases}$$

This yields a new LTS, call it LTS* (the path LTS).

Then the notions of LTS and of LTS* bisimulation coincide.

From strong to weak bisimulation

Take the LTS of CCS, with $\mathtt{Act} = L \cup \overline{L} \cup \{\tau\}$, call it strong. The bisimulation for this system is called strong bisimulation. Take *Strong** (its path LTS).

Consider the following LTS, call it *Weak*†, with the same set of actions as *Strong**:

$$P \stackrel{s}{\Longrightarrow} Q \text{ if and only if } \exists t\ P \stackrel{t}{\longrightarrow} Q \text{ and } \hat{s} = \hat{t}$$

where the function $s \mapsto \hat{s}$ is defined as follows:

$$\hat{\epsilon} = \epsilon \qquad \hat{\tau} = \epsilon \qquad \hat{\alpha} = \alpha \qquad \widehat{s\mu} = \hat{s}\hat{\mu}$$

The idea is that weak bisimulation is *bisimulation with possibly τ actions interspersed.*

From strong to weak bisimulation, ctd.

Let *Weak* be the LTS on `Act` whose transitions are $P \stackrel{\mu}{\Longrightarrow} Q$, that is:

$$P \stackrel{\tau}{\Longrightarrow} Q \text{ iff } P \xrightarrow{\tau}{}^{*} Q \qquad\qquad P \stackrel{\alpha}{\Longrightarrow} Q \text{ iff } P \xrightarrow{\tau}{}^{*} \xrightarrow{\alpha}\xrightarrow{\tau}{}^{*} Q$$

It holds $\mathit{Weak}^{\dagger} = \mathit{Weak}^{\star}$.

Unfortunately, none of the three equivalent definition of weak bisimulation obtained from the LTS's (*Weak*, $\mathit{Weak}^{\dagger}$, $\mathit{Weak}^{\star}$) is practical.

From strong to weak bisimulation, ctd.

A weak bisimulation is a relation $\mathcal{R}$ such that

$$P \mathrel{\mathcal{R}} Q \quad \Rightarrow \quad \forall \mu, P' \; (P \xrightarrow{\mu} P' \Rightarrow \exists Q' \; Q \overset{\mu}{\Longrightarrow} Q' \text{ and } P' \mathrel{\mathcal{R}} Q')$$

and conversely.

(Note the dissimetry between the use of $\xrightarrow{\mu}$ on the left and of $\overset{\mu}{\Longrightarrow}$ on the right.)

Two processes are weakly bisimilar, denoted $P \approx Q$ if there exists a weak bisimulation $\mathcal{R}$ such that $P \mathrel{\mathcal{R}} Q$.

Let weak bisimilarity be the largest weak bisimulation.

Weak bisimulation is a congruence

Weak bisimilarity is also a congruence (for our choice of language with guarded sums).

Same proof technique of the strong case: define $\hat{\approx}$. For the forward phase, we use the following properties:

$$(P \stackrel{\mu}{\Longrightarrow} P') \quad \Rightarrow \quad ((\boldsymbol{\nu} a)P \stackrel{\mu}{\Longrightarrow} ((\boldsymbol{\nu} a)Q') \text{ for } \mu \neq a, \overline{a}$$

$$(Q_1 \stackrel{\mu}{\Longrightarrow} Q_1') \quad \Rightarrow \quad (Q_1 \mid\mid Q_2 \stackrel{\mu}{\Longrightarrow} Q_1' \mid\mid Q_2')$$

$$(Q_1 \stackrel{a}{\Longrightarrow} Q_1' \text{ and } Q_2 \stackrel{\overline{\alpha}}{\Longrightarrow} Q_2') \quad \Rightarrow \quad (Q_1 \mid\mid Q_2 \stackrel{\tau}{\Longrightarrow} Q_1' \mid\mid Q_2')$$

Exercise: Prove it.

Next lecture

I will be away (ICFP), so James Leifer (Moscova research team, INRIA) will talk about

- much more on weak semantics;
- axiomatizations;
- powerful proof techniques;
- amazing examples.

Do not miss his lecture!