Semantics and Verification

Lecture 3

12 February 2010

Overview

Last lecture:

- CCS: syntax
- semantics of CCS (as transition systems)

This lecture:

- Behavioral equivalences: strong bisimilarity

Next lecture:

- Weak bisimilarity

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange integer values.

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil$$

$$\downarrow \tau$$

$$Nil \mid \overline{save(3)}.Nil$$

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange integer values.

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil \mid Bank(100)$$

$$\downarrow \tau$$

$$Nil \mid \overline{save(3)}.Nil \mid Bank(100)$$

$$\downarrow \tau$$

$$Nil \mid Nil \mid Bank(103)$$

Parametrized Process Constants

For example: $Bank(total) \stackrel{\text{def}}{=} save(x).Bank(total + x)$.

Translation of Value Passing CCS to Standard CCS

Value Passing CCS

$$C \stackrel{\text{def}}{=} in(x).C'(x)$$

$$C'(x) \stackrel{\text{def}}{=} \overline{out(x)}.C$$

$$\longrightarrow$$

Standard CCS

$$C \stackrel{\text{def}}{=} \sum_{i \in \mathbb{N}} in(i).C'_i$$

$$C'_i \stackrel{\text{def}}{=} \overline{out(i)}.C$$

$$\begin{tikzcd}
C \arrow[d, bend left, "in(x)"] \\
C'(x) \arrow[u, bend left, "\overline{out(x)}"]
\end{tikzcd}$$

symbolic LTS

$$\cdots$$

$$\begin{tikzcd}
C'_i \arrow[r, bend left, "\overline{out(i)}"] & C \arrow[l, bend left, "in(i)"] \arrow[r, bend left, "in(2)"] \arrow[d, bend left, "in(1)"] & C'_2 \arrow[l, bend left, "\overline{out(2)}"] \\
& C'_1 \arrow[u, bend left, "\overline{out(1)}"] &
\end{tikzcd}$$

infinite LTS

Behavioral Equivalence

Implementation

$$CM \stackrel{\text{def}}{=} coin.\overline{coffee}.CM$$

$$CS \stackrel{\text{def}}{=} \overline{pub}.\overline{coin}.coffee.CS$$

$$Uni \stackrel{\text{def}}{=} (CM \mid CS) \smallsetminus \{coin, coffee\}$$

Specification

$$Spec \stackrel{\text{def}}{=} \overline{pub}.Spec$$

Question

Are the processes *Uni* and *Spec* behaviorally equivalent?

$$Uni \equiv Spec$$

Goals

What should a reasonable behavioral equivalence satisfy?

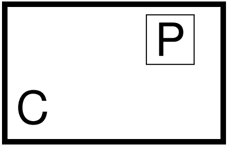
- abstract from states (consider only the behavior – actions)
- abstract from nondeterminism
- abstract from internal behavior

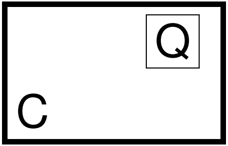
What else should a reasonable behavioral equivalence satisfy?

- reflexivity $P \equiv P$ for any process P
- transitivity $Spec_0 \equiv Spec_1 \equiv Spec_2 \equiv \cdots \equiv Impl$ implies that $Spec_0 \equiv Impl$
- symmetry $P \equiv Q$ iff $Q \equiv P$

– an equivalence relation

Congruence





$C(P)$ $C(Q)$

Congruence Property

$P \equiv Q$ implies that $C(P) \equiv C(Q)$

Example

$$\mathsf{CTM} \stackrel{\text{def}}{=} \mathsf{coin}.\big(\overline{\mathsf{coffee}}.\mathsf{CTM} + \overline{\mathsf{tea}}.\mathsf{CTM}\big)$$

$$\mathsf{CTM}' \stackrel{\text{def}}{=} \mathsf{coin}.\overline{\mathsf{coffee}}.\mathsf{CTM}' + \mathsf{coin}.\overline{\mathsf{tea}}.\mathsf{CTM}'$$

Should those two be equivalent?

No: Let $\mathsf{CA} \stackrel{\text{def}}{=} \overline{\mathsf{coin}}.\mathsf{coffee}.\mathsf{CA}$ and take the context

$$C(P) = (\mathsf{CA} \mid P) \setminus \{\mathsf{coin}, \mathsf{coffee}, \mathsf{tea}\}$$

Then $C(\mathsf{CTM})$ and $C(\mathsf{CTM}')$ have quite different behavior!

First Idea: Trace Equivalence

Let $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ be an LTS.

Trace Set for $s \in Proc$

$$Traces(s) = \{w \in Act^* \mid \exists s' \in Proc.\ s \xrightarrow{w} s'\}$$

Let $s \in Proc$ and $t \in Proc$.

Trace Equivalence

We say that s and t are trace equivalent ($s \equiv_T t$) if and only if $Traces(s) = Traces(t)$

– Not good enough: CTM and CTM′ are trace equivalent!

Second Idea: Black-Box Experiments

Experiment in CTM	Experiment in CTM′
$\boxed{\text{coin}}\ \overline{\text{tea}}\ \overline{\text{coffee}}$	$\boxed{\text{coin}}\ \overline{\text{tea}}\ \overline{\text{coffee}}$

Second Idea: Black-Box Experiments

Experiment in CTM	Experiment in CTM′
$\boxed{\text{coin}}$ $\overline{\text{tea}}$ $\overline{\text{coffee}}$	$\boxed{\text{coin}}$ $\overline{\text{tea}}$ $\overline{\text{coffee}}$
press coin	press coin
coin $\boxed{\overline{\text{tea}}}$ $\boxed{\overline{\text{coffee}}}$	coin $\boxed{\overline{\text{tea}}}$ $\overline{\text{coffee}}$ coin $\overline{\text{tea}}$ $\boxed{\overline{\text{coffee}}}$

Main Idea

Two processes are behaviorally equivalent if and only if an external observer cannot tell them apart.

Strong Bisimilarity

Let $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ be an LTS.

Strong Bisimulation

A binary relation $R \subseteq Proc \times Proc$ is a strong bisimulation iff whenever $(s, t) \in R$ then for each $a \in Act$:

- if $s \xrightarrow{a} s'$ then $t \xrightarrow{a} t'$ for some t' such that $(s', t') \in R$
- if $t \xrightarrow{a} t'$ then $s \xrightarrow{a} s'$ for some s' such that $(s', t') \in R$.

Strong Bisimilarity

Two processes $p_1, p_2 \in Proc$ are strongly bisimilar ($p_1 \sim p_2$) if and only if there exists a strong bisimulation R such that $(p_1, p_2) \in R$.

$$\sim = \bigcup \{R \mid R \text{ is a strong bisimulation}\}$$

Basic Properties

Theorem

$\sim$ *is an equivalence relation (reflexive, symmetric, transitive).*

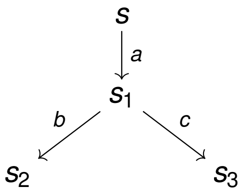
Theorem

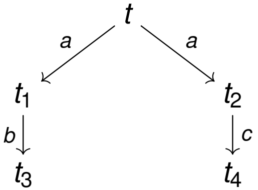
$\sim$ *is the largest strong bisimulation.*

Theorem

$s \sim t$ if and only if for each $a \in Act$:

- *if $s \xrightarrow{a} s'$ then $t \xrightarrow{a} t'$ for some t' such that $s' \sim t'$,*
- *if $t \xrightarrow{a} t'$ then $s \xrightarrow{a} s'$ for some s' such that $s' \sim t'$.*

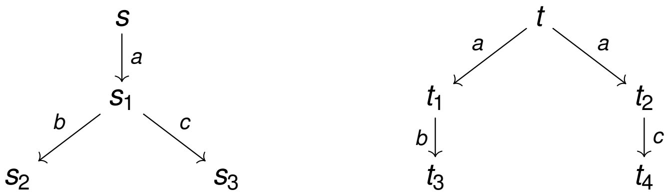




To prove that $s \not\sim t$:

- Enumerate all binary relations and show that none of them at the same time contains (s, t) and is a strong bisimulation. (Expensive: $2^{|Proc|^2}$ relations.)

How to Show Nonbisimilarity?



To prove that $s \not\sim t$:

- Enumerate all binary relations and show that none of them at the same time contains (s, t) and is a strong bisimulation. (Expensive: $2^{|Proc|^2}$ relations.) or
- Make certain observations which will enable to disqualify many bisimulation candidates in one step.
- Use game characterization of strong bisimilarity. (Less expensive)

Strong Bisimulation Game

Let $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ be an LTS and $s, t \in Proc$.

Define a two-player game of an 'attacker' and a 'defender' starting from s and t.

- The game is played in rounds and configurations of the game are pairs of states from $Proc \times Proc$.
- In every round exactly one configuration is called current. Initially the configuration (s, t) is the current one.

Intuition

The defender wants the show that s and t are strongly bisimilar while the attacker aims to prove the opposite.

Rules of the Bisimulation Game

Game Rules

In each round the players change the current configuration as follows:

1. the attacker chooses one of the processes in the current configuration and makes an $\xrightarrow{a}$-move for some $a \in Act$, and
2. the defender must respond by making an $\xrightarrow{a}$-move in the other process under the same action a.

The newly reached pair of processes becomes the current configuration. The game then continues by another round.

Result of the Game

- If one player cannot move, the other player wins.
- If the game is infinite, the defender wins.

Game Characterization of Strong Bisimilarity

Theorem

- States s and t are strongly bisimilar if and only if the defender has a *universal winning strategy* starting from the configuration (s, t).
- States s and t are not strongly bisimilar if and only if the attacker has a *universal winning strategy* starting from the configuration (s, t).

Remark

Bisimulation games can be used to prove both bisimilarity and nonbisimilarity of two processes. It very often provides elegant arguments for the negative case.