

# Semantics and Verification 2006

## Lecture 3

- value passing CCS
- behavioural equivalences
- strong bisimilarity and bisimulation games
- properties of strong bisimilarity

# Value Passing CCS

## Main Idea

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

$$\begin{array}{c} \overline{\text{pay}}(6).\text{Nil} \mid \overline{\text{pay}}(x).\overline{\text{save}}(x/2).\text{Nil} \mid \text{Bank}(100) \\ \downarrow \tau \\ \text{Nil} \mid \overline{\text{save}}(3).\text{Nil} \mid \text{Bank}(100) \\ \downarrow \tau \\ \text{Nil} \mid \text{Nil} \mid \text{Bank}(103) \end{array}$$

## Parametrized Process Constants

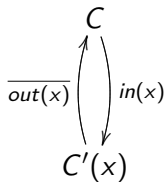
For example:  $\text{Bank}(\text{total}) \stackrel{\text{def}}{=} \text{save}(x).\text{Bank}(\text{total} + x)$ .

# Translation of Value Passing CCS to Standard CCS

## Value Passing CCS

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

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



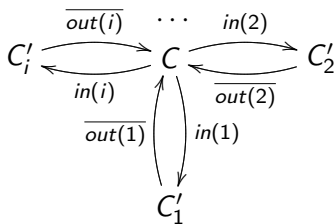
symbolic LTS

→

## Standard CCS

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

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



infinite LTS

## CCS Has Full Turing Power

### Fact

CCS can simulate a computation of any Turing machine.

### Remark

Hence CCS is as expressive as any other programming language but its use is to rather **describe** the behaviour of reactive systems than to perform specific calculations.

# Behavioural Equivalence

## Implementation

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

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

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

## Specification

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

## Question

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

$$Uni \equiv Spec$$

## Goals

What should a reasonable behavioural equivalence satisfy?

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

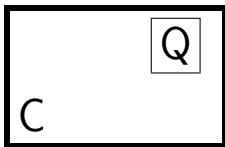
What else should a reasonable behavioural equivalence satisfy?

- **reflexivity**  $P \equiv P$  for any process  $P$
- **transitivity**  $Spec_0 \equiv Spec_1 \equiv Spec_2 \equiv \dots \equiv Impl$  gives that  
 $Spec_0 \equiv Impl$
- **symmetry**  $P \equiv Q$  iff  $Q \equiv P$

# Congruence



$C(P)$



$C(Q)$

## Congruence Property

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

# 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

$$Traces(s) = Traces(t)$$



## Black-Box Experiments

Experiment in A

coin  $\overline{tea}$   $\overline{coffee}$

press *coin*

*coin*  $\overline{tea}$   $\overline{coffee}$

Experiment in B

coin  $\overline{tea}$   $\overline{coffee}$

press *coin*

*coin*  $\overline{tea}$   $\overline{coffee}$

Experiment in B

coin  $\overline{tea}$   $\overline{coffee}$

press *coin*

*coin*  $\overline{tea}$   $\overline{coffee}$

### 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  $(p_1, p_2) \in R$ .

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

## Basic Properties of Strong Bisimilarity

### Theorem

$\sim$  is an equivalence (reflexive, symmetric and transitive)

### Theorem

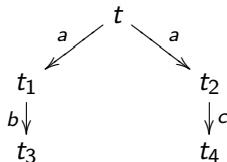
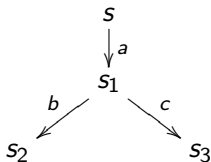
$\sim$  is the largest strong bisimulation

### Theorem

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

- 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  $s' \sim t'$ .

# 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.)
- Make certain **observations** which will enable to disqualify many bisimulation candidates in one step.
- Use **game characterization** of strong bisimilarity.

# Strong Bisimulation Game

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

We 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 to show that  $s$  and  $t$  are strongly bisimilar while the attacker aims to prove the opposite.

# Rules of the Bisimulation Games

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

# Strong Bisimilarity is a Congruence for CCS Operations

## Theorem

Let  $P$  and  $Q$  be CCS processes such that  $P \sim Q$ . Then

- $\alpha.P \sim \alpha.Q$  for each action  $\alpha \in \text{Act}$
- $P + R \sim Q + R$  and  $R + P \sim R + Q$  for each CCS process  $R$
- $P | R \sim Q | R$  and  $R | P \sim R | Q$  for each CCS process  $R$
- $P[f] \sim Q[f]$  for each relabelling function  $f$
- $P \setminus L \sim Q \setminus L$  for each set of labels  $L$ .



## Other Properties of Strong Bisimilarity

Following Properties Hold for any CCS Processes  $P$ ,  $Q$  and  $R$

- $P + Q \sim Q + P$
- $P | Q \sim Q | P$
- $P + Nil \sim P$
- $P | Nil \sim P$
- $(P + Q) + R \sim P + (Q + R)$
- $(P | Q) | R \sim P | (Q | R)$