

Modeling Concurrent and Probabilistic Systems

Lecture 4: Definition of Strong Bisimulation

Joost-Pieter Katoen Thomas Noll

Software Modeling and Verification Group
RWTH Aachen University
`noll@cs.rwth-aachen.de`

`http://www-i2.informatik.rwth-aachen.de/i2/mcps07/`

Winter Semester 2007/08

- 1 Repetition: Equivalence of CCS Processes
- 2 Definition of Strong Bisimulation

Repetition: Desired Properties of Equivalence

Wanted: a “feasible” (i.e., efficiently decidable) semantic equivalence between CCS processes which

- ① identifies processes whose **LTSs coincide**,
- ② **implies trace equivalence**, i.e., considers two processes equivalent only if both can execute the same actions sequences (formal definition later), and
- ③ is a **congruence**, i.e., allows to replace a subprocess by an equivalent counterpart without changing the overall semantics of the system (formal definition later).
- ④ is **deadlock sensitive**, i.e., if $P \cong Q$ and if P has a w -deadlock, then Q has a w -deadlock (and vice versa, by equivalence).

Formally: we are looking for a deadlock-sensitive congruence relation $\cong \subseteq \text{Proc} \times \text{Proc}$ such that

$$\text{LTS}(P) = \text{LTS}(Q) \implies P \cong Q \implies \text{Tr}(P) = \text{Tr}(Q)$$

Repetition: Trace Equivalence

Definition (Trace language)

For every $P \in \text{Prc}$, let

$$\text{Tr}(P) := \{w \in \text{Act}^* \mid \text{ex. } P' \in \text{Prc} \text{ such that } P \xrightarrow{w}^* P'\}$$

be the **trace language** of P .

$P, Q \in \text{Prc}$ are called **trace equivalent** if $\text{Tr}(P) = \text{Tr}(Q)$.

Example (One-place buffer)

$$B(\text{in}, \text{out}) = \text{in}.\overline{\text{out}}.B(\text{in}, \text{out})$$

$$\implies \text{Tr}(B) = (\text{in} \cdot \overline{\text{out}})^* \cdot (\text{in} + \varepsilon)$$

Repetition: CCS Congruences

Goal: replacing a subcomponent of a system by an equivalent process should yield an equivalent systems

$\implies$ modular system development

Definition (CCS congruence)

An equivalence relation $\cong \subseteq Prc \times Prc$ is said to be a **CCS congruence** if it is preserved by the CCS constructs; that is, if $P \cong Q$ then

$$\begin{aligned}\alpha.P &\cong \alpha.Q \\ P + R &\cong Q + R \\ R + P &\cong R + Q \\ P \parallel R &\cong Q \parallel R \\ R \parallel P &\cong R \parallel Q \\ \text{new } a P &\cong \text{new } a Q\end{aligned}$$

for every $\alpha \in Act$, $R \in Prc$, and $a \in N$.

Definition (Deadlock)

Let $P, Q \in \text{Prc}$ and $w \in \text{Act}^*$ such that $P \xrightarrow{w}^* Q$ and $Q \not\rightarrow$. Then Q is called a ***w*-deadlock** of P .

- Thus $P := a.b.\text{nil} + a.\text{nil}$ has an a -deadlock, in contrast to $Q := a.b.\text{nil}$.
- Such properties are important since it can be crucial that a certain communication is eventually possible.

- 1 Repetition: Equivalence of CCS Processes
- 2 Definition of Strong Bisimulation

Definition of Strong Bisimulation I

Observation: equivalence should be deadlock sensitive
 $\implies$ needs to take **branching structure** of processes into account

This is guaranteed by a definition according to the following scheme:

Bisimulation scheme

$P, Q \in \text{Proc}$ are equivalent iff, for every $\alpha \in \text{Act}$, every α -successor of P is equivalent to some α -successor of Q , and vice versa.

In the first version we will ignore the special function of the silent action τ ($\implies$ *weak bisimulation*)

Definition of Strong Bisimulation I

Observation: equivalence should be deadlock sensitive
 $\implies$ needs to take **branching structure** of processes into account

This is guaranteed by a definition according to the following scheme:

Bisimulation scheme

$P, Q \in \text{Proc}$ are equivalent iff, for every $\alpha \in \text{Act}$, every α -successor of P is equivalent to some α -successor of Q , and vice versa.

In the first version we will ignore the special function of the silent action τ ($\implies$ *weak bisimulation*)

Definition of Strong Bisimulation I

Observation: equivalence should be deadlock sensitive
 $\implies$ needs to take **branching structure** of processes into account

This is guaranteed by a definition according to the following scheme:

Bisimulation scheme

$P, Q \in \text{Proc}$ are equivalent iff, for every $\alpha \in \text{Act}$, every α -successor of P is equivalent to some α -successor of Q , and vice versa.

In the first version we will ignore the special function of the silent action τ ($\implies$ *weak bisimulation*)

Definition of Strong Bisimulation II

Definition 4.1 (Strong bisimulation)

A relation $\rho \subseteq Prc \times Prc$ is called a **strong bisimulation** if $P\rho Q$ implies, for every $\alpha \in Act$,

- ① $P \xrightarrow{\alpha} P' \implies \text{ex. } Q' \in Prc \text{ such that } Q \xrightarrow{\alpha} Q' \text{ and } P'\rho Q'$
- ② $Q \xrightarrow{\alpha} Q' \implies \text{ex. } P' \in Prc \text{ such that } P \xrightarrow{\alpha} P' \text{ and } P'\rho Q'$

$P, Q \in Prc$ are called **strongly bisimilar** (notation: $P \sim Q$) if there exists a strong bisimulation ρ such that $P\rho Q$.

Theorem 4.2

$\sim$ is an equivalence relation.

Proof.

on the board



Definition of Strong Bisimulation II

Definition 4.1 (Strong bisimulation)

A relation $\rho \subseteq Prc \times Prc$ is called a **strong bisimulation** if $P\rho Q$ implies, for every $\alpha \in Act$,

- ① $P \xrightarrow{\alpha} P' \implies \text{ex. } Q' \in Prc \text{ such that } Q \xrightarrow{\alpha} Q' \text{ and } P'\rho Q'$
- ② $Q \xrightarrow{\alpha} Q' \implies \text{ex. } P' \in Prc \text{ such that } P \xrightarrow{\alpha} P' \text{ and } P'\rho Q'$

$P, Q \in Prc$ are called **strongly bisimilar** (notation: $P \sim Q$) if there exists a strong bisimulation ρ such that $P\rho Q$.

Theorem 4.2

$\sim$ is an equivalence relation.

Proof.

on the board



Definition of Strong Bisimulation II

Definition 4.1 (Strong bisimulation)

A relation $\rho \subseteq Prc \times Prc$ is called a **strong bisimulation** if $P\rho Q$ implies, for every $\alpha \in Act$,

- ① $P \xrightarrow{\alpha} P' \implies \text{ex. } Q' \in Prc \text{ such that } Q \xrightarrow{\alpha} Q' \text{ and } P'\rho Q'$
- ② $Q \xrightarrow{\alpha} Q' \implies \text{ex. } P' \in Prc \text{ such that } P \xrightarrow{\alpha} P' \text{ and } P'\rho Q'$

$P, Q \in Prc$ are called **strongly bisimilar** (notation: $P \sim Q$) if there exists a strong bisimulation ρ such that $P\rho Q$.

Theorem 4.2

$\sim$ is an equivalence relation.

Proof.

on the board



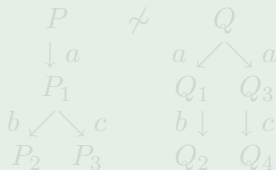
Example 4.3

(on the board)

1

$$\begin{array}{cc}
 P & Q_1 \\
 \circlearrowleft & a \downarrow \uparrow a \\
 a & Q_2
 \end{array}$$

2

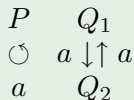


(remember: $Tr(P) = Tr(Q)$)

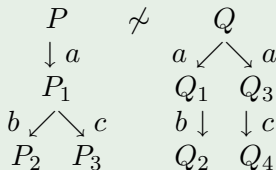
Example 4.3

(on the board)

①



②



(remember: $Tr(P) = Tr(Q)$)

Example 4.4

Binary semaphore

(controls exclusive access to two instances of a resource)

Sequential definition:

$$Sem_0(get, put) = get.Sem_1(get, put)$$

$$Sem_1(get, put) = get.Sem_2(get, put) + put.Sem_0(get, put)$$

$$Sem_2(get, put) = put.Sem_1(get, put)$$

Parallel definition:

$$S(get, put) = S_0(get, put) \parallel S_0(get, put)$$

$$S_0(get, put) = get.S_1(get, put)$$

$$S_1(get, put) = put.S_0(get, put)$$

Proposition: $Sem_0(get, put) \sim S(get, put)$ (see 3rd ex. sheet)

Example 4.5

Two-place buffer

Sequential definition:

$$B_0(in, out) = in.B_1(in, out)$$

$$B_1(in, out) = \overline{out}.B_0(in, out) + in.B_2(in, out)$$

$$B_2(in, out) = \overline{out}.B_1(in, out)$$

Parallel definition:

$$B_{\parallel}(in, out) = \text{new } com (B(in, com) \parallel B(com, out))$$

$$B(in, out) = in.\overline{out}.B(in, out)$$

Proposition: $B_0(in, out) \not\sim B_{\parallel}(in, out)$ (see 3rd ex. sheet)