Modeling Concurrent and Probabilistic Systems

Lecture 6: Properties 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/mcps09/

Summer Semester 2009

Outline

Repetition: Definition of Strong Bisimulation

Definition (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. $P \xrightarrow{\alpha} P' \implies$ ex. $Q' \in Prc$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \rho Q'$
2. $Q \xrightarrow{\alpha} Q' \implies$ ex. $P' \in Prc$ such that $P \xrightarrow{\alpha} P'$ 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

$\sim$ *is an equivalence relation.*

Proof.

on the board □

Repetition: Examples

Example

(on the board)

1.

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

2.

$$
\begin{array}{ccc}
\begin{array}{ccc}
 & P & \\
 & \downarrow a & \\
 & P_1 & \\
b \swarrow & & \searrow c \\
P_2 & & P_3
\end{array}
& \not\sim &
\begin{array}{ccc}
 & Q & \\
a \swarrow & & \searrow a \\
Q_1 & & Q_3 \\
b \downarrow & & \downarrow c \\
Q_2 & & Q_4
\end{array}
\end{array}
$$

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

Repetition: Bisimulation Implies Trace Equivalence

Definition (Trace language; repetition)

The trace language of $P \in Prc$ is given by

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

Theorem 6.4

For every $P, Q \in Prc$, $P \sim Q$ implies $Tr(P) = Tr(Q)$.

Proof.

- Assume that $P \sim Q$ but (w.l.o.g.) $w \in Tr(P) \setminus Tr(Q)$.
- Let $v \in Act^*$ be the longest prefix of w such that $v \in Tr(Q)$ (i.e., $w = v\alpha u$ for some $\alpha \in Act$ and $u \in Act^*$).
- Let $P', P'' \in Prc$ such that $P \xrightarrow{v} P' \xrightarrow{\alpha} P''$.
- Since $P \sim Q$ there exists $Q' \in Prc$ such that $Q \xrightarrow{v} Q'$ and $P' \sim Q'$ (by induction on $|v|$).
- But we have that $P' \xrightarrow{\alpha} P''$ whereas $Q' \not\xrightarrow{\alpha}$ ↯

Outline

Congruence Property of Strong Bisimulation I

The congruence proof employs the following lemma.

Lemma 6.1

For every $P, Q, R \in Prc$,

1. $P + Q \sim Q + P$
2. $P + (Q + R) \sim (P + Q) + R$
3. $P + \mathsf{nil} \sim P$
4. $P \parallel Q \sim Q \parallel P$
5. $P \parallel (Q \parallel R) \sim (P \parallel Q) \parallel R$
6. $P \parallel \mathsf{nil} \sim P$

Proof.

1. on the board
2. on the board
3. 3rd ex. sheet
4. on the board
5. 3rd ex. sheet
6. on the board

□

Congruence Property of Strong Bisimulation II

Definition (CCS congruence; repetition)

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

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

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

Theorem 6.2

$\sim$ is a CCS congruence.

Proof.

on the board □

Outline

Deadlock Sensitivity of Strong Bisimulation

Definition (Deadlock; repetition)

Let $P, Q \in \mathit{Prc}$ and $w \in \mathit{Act}^*$ such that $P \xrightarrow{w} Q$ and $Q \not\longrightarrow$. Then Q is called a w-deadlock of P.
An equivalence relation $\cong \subseteq \mathit{Prc} \times \mathit{Prc}$ is called deadlock sensitive if for every $P \cong Q$ such that P has a w-deadlock, Q also has a w-deadlock.

Theorem 6.3

$\sim$ is deadlock sensitive.

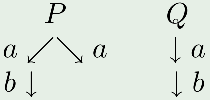
Proof.

on the board □

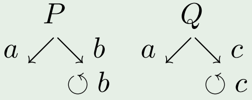
Outline

Remark: traces and deadlocks are independent in the following sense

Example 6.4



same traces
different deadlocks



different traces
same deadlocks

But: processes with finite trace sets and identical deadlocks are trace equivalent (since every trace is a prefix of some deadlock)