Modeling Concurrent and Probabilistic Systems

Lecture 5: 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: Deadlocks I

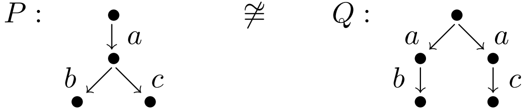
Definition (Deadlock)

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

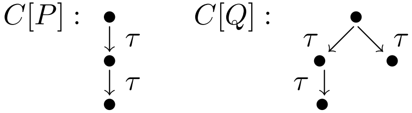
- Thus $P := a.b.\mathsf{nil} + a.\mathsf{nil}$ has an a-deadlock, in contrast to $Q := a.b.\mathsf{nil}$.
- Such properties are important since it can be crucial that a certain communication is eventually possible.
- We therefore extend our set of postulates: our semantic equivalence $\cong$ should
  1. identify processes with identical LTSs;
  2. imply trace equivalence;
  3. be a congruence; and
  4. be 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).

Repetition: Deadlocks II

The combination of congruence and deadlock sensitivity also excludes the following equivalence:



If $P \cong Q$, by congruence this equivalence should hold in every context. But $C[\cdot] := \mathsf{new}\, a, b, c\, (\overline{a}.\overline{b}.\mathsf{nil} \parallel \cdot)$ yields the following conflict:



no τ-deadlock $\qquad$ τ-deadlock

Remarks:

- Another motivation: elevator control with a = "call elevator", b = "choose 1st floor", c = "choose 2nd floor",
- P and Q are obviously trace equivalent

Outline

Definition of Strong Bisimulation I

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

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

Bisimulation scheme

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

- First version ignores special function of silent action τ ($\Longrightarrow$ *weak bisimulation*)
- Unidirectional version considered later ($\Longrightarrow$ *simulation*)

Definition of Strong Bisimulation I

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

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

Bisimulation scheme

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

- First version ignores special function of silent action τ ($\Longrightarrow$ *weak bisimulation*)
- Unidirectional version considered later ($\Longrightarrow$ *simulation*)

Definition of Strong Bisimulation I

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

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

Bisimulation scheme

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

- First version ignores special function of silent action τ ($\Longrightarrow$ *weak bisimulation*)
- Unidirectional version considered later ($\Longrightarrow$ *simulation*)

Definition of Strong Bisimulation II

Definition 5.1 (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. $P \xrightarrow{\alpha} P' \implies$ 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 5.2

$\sim$ is an equivalence relation.

Proof.

on the board □

Definition of Strong Bisimulation II

Definition 5.1 (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. $P \xrightarrow{\alpha} P' \implies$ 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 5.2

$\sim$ is an equivalence relation.

Proof.

on the board □

Definition of Strong Bisimulation II

Definition 5.1 (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. $P \xrightarrow{\alpha} P' \implies$ 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 5.2

$\sim$ *is an equivalence relation.*

Proof.

on the board □

Example 5.3

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

Example 5.3

(on the board)

1.

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

2.

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

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

Examples II

Example 5.4

Binary semaphore

(controls exclusive access to two instances of a resource)

Sequential definition:

$$\begin{aligned}
Sem_0(get, put) &= get.Sem_1(get, put) \\
Sem_1(get, put) &= get.Sem_2(get, put) + put.Sem_0(get, put) \\
Sem_2(get, put) &= put.Sem_1(get, put)
\end{aligned}$$

Parallel definition:

$$\begin{aligned}
S(get, put) &= S_0(get, put) \parallel S_0(get, put) \\
S_0(get, put) &= get.S_1(get, put) \\
S_1(get, put) &= put.S_0(get, put)
\end{aligned}$$

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

Examples III

Example 5.5

Two-place buffer

Sequential definition:

$$\begin{array}{rcl} B_0(in, out) & = & in.B_1(in, out) \\ B_1(in, out) & = & \overline{out}.B_0(in, out) + in.B_2(in, out) \\ B_2(in, out) & = & \overline{out}.B_1(in, out) \end{array}$$

Parallel definition:

$$\begin{array}{rcl} B_{\|}(in, out) & = & \mathsf{new}\; com\; (B(in, com) \parallel B(com, out)) \\ B(in, out) & = & in.\overline{out}.B(in, out) \end{array}$$

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

Outline

Properties of Strong Bisimulation

It remains to show that strong bisimulation has the required properties of a process equivalence:

1. Identification of processes with identical LTSs:
   since the definition of strong bisimulation directly relies on the transition relation, processes with identical transition trees are clearly strongly bisimilar
2. Implication of trace equivalence: following slides
3. CCS congruence: following slides
4. Deadlock sensitivity: following slides

Properties of Strong Bisimulation

It remains to show that strong bisimulation has the required properties of a process equivalence:

1. Identification of processes with identical LTSs:
   since the definition of strong bisimulation directly relies on the transition relation, processes with identical transition trees are clearly strongly bisimilar
2. Implication of trace equivalence: following slides
3. CCS congruence: following slides
4. Deadlock sensitivity: following slides

Strong Bisimulation Implies Trace Equivalence

Definition (Trace language; repetition)

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

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

Theorem 5.6

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

Proof.

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

□

Strong 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 5.6

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 $Q' \not\xrightarrow{\alpha}$ ↯

$\square$

Strong 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 5.6

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 $Q' \not\xrightarrow{\alpha}$ $\lightning$

□

Strong 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 5.6

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 $Q' \not\xrightarrow{\alpha}$ $\lightning$

□

Strong 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 5.6

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 $Q' \not\xrightarrow{\alpha}$ $\lightning$

□

Strong 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 5.6

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 $Q' \not\xrightarrow{\alpha}$ $\lightning$

□

Strong 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 5.6

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 $Q' \not\xrightarrow{\alpha}$ ↯

□