Modeling Concurrent and Probabilistic Systems

Lecture 7: Decidability of Strong Bisimulation & Definition of Strong Simulation

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 □

Congruence Property of Strong Bisimulation

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

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

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

Theorem 7.3

$\sim$ is a CCS congruence.

Proof.

on the board □

Deadlock Sensitivity of Strong Bisimulation

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.
An equivalence relation $\cong \subseteq Prc \times Prc$ is called deadlock sensitive if for every $P \cong Q$ such that P has a w-deadlock, Q also has a w-deadlock.

Theorem 7.4

$\sim$ is deadlock sensitive.

Proof.

on the board □

Outline

The Problem

We now show that the word problem for strong bisimulation

Problem (Word problem for strong bisimulation)

Given: $P, Q \in Prc$

Question: $P \sim Q$?

is decidable for finite-state processes (i.e., for those with $|S(P)|, |S(Q)| < \infty$ where $S(P) := \{P' \in Prc \mid P \longrightarrow P'\}$)

(in general it is undecidable – see 4th ex. sheet).

To this aim we give an algorithm which iteratively partitions the state set of an LTS such that the single blocks correspond to the $\sim$-equivalence classes.

The Problem

We now show that the word problem for strong bisimulation

Problem (Word problem for strong bisimulation)

Given: $P, Q \in Prc$

Question: $P \sim Q$?

is decidable for finite-state processes (i.e., for those with $|S(P)|, |S(Q)| < \infty$ where $S(P) := \{P' \in Prc \mid P \longrightarrow P'\}$)

(in general it is undecidable – see 4th ex. sheet).

To this aim we give an algorithm which iteratively partitions the state set of an LTS such that the single blocks correspond to the $\sim$-equivalence classes.

The Partitioning Algorithm I

Theorem 7.1 (Partitioning algorithm for $\sim$)

Input: *LTS* $(S, Act, \longrightarrow)$ *(S finite)*

Procedure:
1. *Start with initial partition* $\Pi := \{S\}$
2. *Let* $B \in \Pi$ *be a block and* $\alpha \in Act$ *an action*
3. *For every* $P \in B$*, let*
   $$\alpha(P) := \{C \in \Pi \mid ex.\ P' \in C \text{ with } P \xrightarrow{\alpha} P'\}$$
   be the set of P*'s* α*-successor blocks*
4. *Partition* $B = \bigcup_{i=1}^{k} B_i$ *such that*
   $$P, Q \in B_i \iff \alpha(P) = \alpha(Q) \text{ for every } \alpha \in Act$$
5. *Let* $\Pi := (\Pi \setminus \{B\}) \cup \{B_1, \ldots, B_k\}$
6. *Continue with (2) until* Π *becomes stable*

Output: *Partition* $\hat{\Pi}$ *of* S

Then, for every $P, Q \in S$*,*

$$P \sim Q \iff ex.\ B \in \hat{\Pi} \text{ with } P, Q \in B$$

The Partitioning Algorithm I

Theorem 7.1 (Partitioning algorithm for $\sim$)

Input: *LTS* $(S, Act, \longrightarrow)$ *(S finite)*

Procedure:

1. *Start with initial partition* $\Pi := \{S\}$
2. *Let* $B \in \Pi$ *be a block and* $\alpha \in Act$ *an action*
3. *For every* $P \in B$*, let*
   $$\alpha(P) := \{C \in \Pi \mid ex.\ P' \in C \text{ with } P \xrightarrow{\alpha} P'\}$$
   be the set of P's α*-successor blocks*
4. *Partition* $B = \bigcup_{i=1}^{k} B_i$ *such that*
   $$P, Q \in B_i \iff \alpha(P) = \alpha(Q) \text{ for every } \alpha \in Act$$
5. *Let* $\Pi := (\Pi \setminus \{B\}) \cup \{B_1, \ldots, B_k\}$
6. *Continue with (2) until* Π *becomes stable*

Output: *Partition* $\hat{\Pi}$ *of S*

Then, for every $P, Q \in S$,

$$P \sim Q \iff ex.\ B \in \hat{\Pi} \text{ with } P, Q \in B$$

The Partitioning Algorithm I

Theorem 7.1 (Partitioning algorithm for $\sim$)

Input: *LTS* $(S, Act, \longrightarrow)$ *(S finite)*

Procedure:
1. *Start with initial partition* $\Pi := \{S\}$
2. *Let* $B \in \Pi$ *be a block and* $\alpha \in Act$ *an action*
3. *For every* $P \in B$*, let*
$$\alpha(P) := \{C \in \Pi \mid \textit{ex. } P' \in C \textit{ with } P \xrightarrow{\alpha} P'\}$$
be the set of P*'s* α*-successor blocks*
4. *Partition* $B = \bigcup_{i=1}^{k} B_i$ *such that*
$$P, Q \in B_i \iff \alpha(P) = \alpha(Q) \textit{ for every } \alpha \in Act$$
5. *Let* $\Pi := (\Pi \setminus \{B\}) \cup \{B_1, \ldots, B_k\}$
6. *Continue with (2) until* Π *becomes stable*

Output: *Partition* $\hat{\Pi}$ *of* S

Then, for every $P, Q \in S$*,*
$$P \sim Q \iff \textit{ex. } B \in \hat{\Pi} \textit{ with } P, Q \in B$$

The Partitioning Algorithm I

Theorem 7.1 (Partitioning algorithm for $\sim$)

Input: *LTS* $(S, Act, \longrightarrow)$ *(S finite)*

Procedure:

1. *Start with initial partition* $\Pi := \{S\}$
2. *Let* $B \in \Pi$ *be a block and* $\alpha \in Act$ *an action*
3. *For every* $P \in B$*, let*
   $$\alpha(P) := \{C \in \Pi \mid ex.\ P' \in C \text{ with } P \xrightarrow{\alpha} P'\}$$
   be the set of P's α-successor blocks
4. *Partition* $B = \bigcup_{i=1}^{k} B_i$ *such that*
   $$P, Q \in B_i \iff \alpha(P) = \alpha(Q) \text{ for every } \alpha \in Act$$
5. *Let* $\Pi := (\Pi \setminus \{B\}) \cup \{B_1, \ldots, B_k\}$
6. *Continue with (2) until* Π *becomes stable*

Output: *Partition* $\hat{\Pi}$ *of* S

Then, for every $P, Q \in S$,

$$P \sim Q \iff ex.\ B \in \hat{\Pi} \text{ with } P, Q \in B$$

Theorem 7.1 (Partitioning algorithm for $\sim$)

Input: *LTS* $(S, Act, \longrightarrow)$ *(S finite)*

Procedure:

1. *Start with initial partition* $\Pi := \{S\}$
2. *Let* $B \in \Pi$ *be a block and* $\alpha \in Act$ *an action*
3. *For every* $P \in B$*, let*

   $$\alpha(P) := \{C \in \Pi \mid \text{ex. } P' \in C \text{ with } P \xrightarrow{\alpha} P'\}$$

   be the set of P's α*-successor blocks*
4. *Partition* $B = \bigcup_{i=1}^{k} B_i$ *such that*

   $$P, Q \in B_i \iff \alpha(P) = \alpha(Q) \text{ for every } \alpha \in Act$$

5. *Let* $\Pi := (\Pi \setminus \{B\}) \cup \{B_1, \ldots, B_k\}$
6. *Continue with (2) until* Π *becomes stable*

Output: *Partition* $\hat{\Pi}$ *of S*

Then, for every $P, Q \in S$,

$$P \sim Q \iff \text{ex. } B \in \hat{\Pi} \text{ with } P, Q \in B$$

The Partitioning Algorithm I

Theorem 7.1 (Partitioning algorithm for $\sim$)

Input: *LTS* $(S, Act, \longrightarrow)$ *(S finite)*

Procedure:

1. *Start with initial partition* $\Pi := \{S\}$
2. *Let* $B \in \Pi$ *be a block and* $\alpha \in Act$ *an action*
3. *For every* $P \in B$, *let*
   $$\alpha(P) := \{C \in \Pi \mid ex.\ P' \in C \text{ with } P \xrightarrow{\alpha} P'\}$$
   be the set of P's α-successor blocks
4. *Partition* $B = \bigcup_{i=1}^{k} B_i$ *such that*
   $$P, Q \in B_i \iff \alpha(P) = \alpha(Q) \text{ for every } \alpha \in Act$$
5. *Let* $\Pi := (\Pi \setminus \{B\}) \cup \{B_1, \ldots, B_k\}$
6. *Continue with (2) until* Π *becomes stable*

Output: *Partition* $\hat{\Pi}$ *of* S

Then, for every $P, Q \in S$,

$$P \sim Q \iff ex.\ B \in \hat{\Pi} \text{ with } P, Q \in B$$

The Partitioning Algorithm I

Theorem 7.1 (Partitioning algorithm for $\sim$)

Input: *LTS* $(S, Act, \longrightarrow)$ *(S finite)*

Procedure:

1. *Start with initial partition* $\Pi := \{S\}$
2. *Let* $B \in \Pi$ *be a block and* $\alpha \in Act$ *an action*
3. *For every* $P \in B$*, let*
   $$\alpha(P) := \{C \in \Pi \mid \textit{ex. } P' \in C \textit{ with } P \stackrel{\alpha}{\longrightarrow} P'\}$$
   be the set of P*'s* α*-successor blocks*
4. *Partition* $B = \bigcup_{i=1}^{k} B_i$ *such that*
   $$P, Q \in B_i \iff \alpha(P) = \alpha(Q) \textit{ for every } \alpha \in Act$$
5. *Let* $\Pi := (\Pi \setminus \{B\}) \cup \{B_1, \ldots, B_k\}$
6. *Continue with (2) until* Π *becomes stable*

Output: *Partition* $\hat{\Pi}$ *of* S

Then, for every $P, Q \in S$,

$$P \sim Q \iff \textit{ex. } B \in \hat{\Pi} \textit{ with } P, Q \in B$$

The Partitioning Algorithm I

Theorem 7.1 (Partitioning algorithm for $\sim$)

Input: *LTS* $(S, Act, \longrightarrow)$ *(S finite)*

Procedure:

1. *Start with initial partition* $\Pi := \{S\}$
2. *Let* $B \in \Pi$ *be a block and* $\alpha \in Act$ *an action*
3. *For every* $P \in B$, *let*
   $$\alpha(P) := \{C \in \Pi \mid ex.\ P' \in C \text{ with } P \xrightarrow{\alpha} P'\}$$
   be the set of P*'s* α*-successor blocks*
4. *Partition* $B = \bigcup_{i=1}^{k} B_i$ *such that*
   $$P, Q \in B_i \iff \alpha(P) = \alpha(Q) \text{ for every } \alpha \in Act$$
5. *Let* $\Pi := (\Pi \setminus \{B\}) \cup \{B_1, \ldots, B_k\}$
6. *Continue with (2) until* Π *becomes stable*

Output: *Partition* $\hat{\Pi}$ *of* S

Then, for every $P, Q \in S$,

$$P \sim Q \iff ex.\ B \in \hat{\Pi} \text{ with } P, Q \in B$$

The Partitioning Algorithm I

Theorem 7.1 (Partitioning algorithm for $\sim$)

Input: *LTS* $(S, Act, \longrightarrow)$ *(S finite)*

Procedure:
1. *Start with initial partition* $\Pi := \{S\}$
2. *Let* $B \in \Pi$ *be a block and* $\alpha \in Act$ *an action*
3. *For every* $P \in B$*, let*
$$\alpha(P) := \{C \in \Pi \mid \textit{ex. } P' \in C \textit{ with } P \xrightarrow{\alpha} P'\}$$
be the set of P's α-successor blocks
4. *Partition* $B = \bigcup_{i=1}^{k} B_i$ *such that*
$$P, Q \in B_i \iff \alpha(P) = \alpha(Q) \textit{ for every } \alpha \in Act$$
5. *Let* $\Pi := (\Pi \setminus \{B\}) \cup \{B_1, \ldots, B_k\}$
6. *Continue with (2) until* Π *becomes stable*

Output: *Partition* $\hat{\Pi}$ *of* S

Then, for every $P, Q \in S$,
$$P \sim Q \iff \textit{ex. } B \in \hat{\Pi} \textit{ with } P, Q \in B$$

The Partitioning Algorithm II

Remark: if states from two disjoint LTSs $(S_1, Act_1, \longrightarrow_1)$ and $(S_2, Act_2, \longrightarrow_2)$ (where $S_1 \cap S_2 = \emptyset$) are to be compared, their union $(S_1 \cup S_2, Act_1 \cup Act_2, \longrightarrow_1 \cup \longrightarrow_2)$ is chosen as input (here usually $Act_1 = Act_2$)

Example 7.2

Binary semaphore (on the board)

Proof.

(Theorem 7.1; on the board) □

The Partitioning Algorithm II

Remark: if states from two disjoint LTSs $(S_1, Act_1, \longrightarrow_1)$ and $(S_2, Act_2, \longrightarrow_2)$ (where $S_1 \cap S_2 = \emptyset$) are to be compared, their union $(S_1 \cup S_2, Act_1 \cup Act_2, \longrightarrow_1 \cup \longrightarrow_2)$ is chosen as input (here usually $Act_1 = Act_2$)

Example 7.2

Binary semaphore (on the board)

Proof.

(Theorem 7.1; on the board) □

The Partitioning Algorithm II

Remark: if states from two disjoint LTSs $(S_1, Act_1, \longrightarrow_1)$ and $(S_2, Act_2, \longrightarrow_2)$ (where $S_1 \cap S_2 = \emptyset$) are to be compared, their union $(S_1 \cup S_2, Act_1 \cup Act_2, \longrightarrow_1 \cup \longrightarrow_2)$ is chosen as input (here usually $Act_1 = Act_2$)

Example 7.2

Binary semaphore (on the board)

Proof.

(Theorem 7.1; on the board) □

Outline

Summary: Properties of Strong Bisimulation

Properties of strong bisimulation

1. $\sim$ is an equivalence relation
2. $LTS(P) = LTS(Q) \implies P \sim Q$
3. $P \sim Q \implies Tr(P) = Tr(Q)$
4. $\sim$ is a CCS congruence
5. $\sim$ is deadlock sensitive
6. $\sim$ is decidable for finite-state processes

Outline

Strong Simulation

Observation: sometimes, the concept of strong bisimulation is too strong (example: extending a system by new features)

Definition 7.3 (Strong simulation)

A relation $\rho \subseteq Prc \times Prc$ is called a strong simulation if, whenever $P \rho Q$ and $P \xrightarrow{\alpha} P'$, there exists $Q' \in Prc$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \rho Q'$.
We say that Q strongly simulates P if there exists a strong simulation ρ such that $P \rho Q$.

Thus: if Q strongly simulates P, then whatever transition path P takes, Q can match it by a path which retains all of P's options.

Example 7.4

Q strongly simulates P, but not vice versa

Strong Simulation

Observation: sometimes, the concept of strong bisimulation is too strong (example: extending a system by new features)

Definition 7.3 (Strong simulation)

A relation $\rho \subseteq Prc \times Prc$ is called a strong simulation if, whenever $P \rho Q$ and $P \xrightarrow{\alpha} P'$, there exists $Q' \in Prc$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \rho Q'$.
We say that Q strongly simulates P if there exists a strong simulation ρ such that $P \rho Q$.

Thus: if Q strongly simulates P, then whatever transition path P takes, Q can match it by a path which retains all of P's options.

Example 7.4

Q strongly simulates P, but not vice versa

Strong Simulation

Observation: sometimes, the concept of strong bisimulation is too strong (example: extending a system by new features)

Definition 7.3 (Strong simulation)

A relation $\rho \subseteq Prc \times Prc$ is called a strong simulation if, whenever $P \rho Q$ and $P \xrightarrow{\alpha} P'$, there exists $Q' \in Prc$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \rho Q'$.
We say that Q strongly simulates P if there exists a strong simulation ρ such that $P \rho Q$.

Thus: if Q strongly simulates P, then whatever transition path P takes, Q can match it by a path which retains all of P's options.

Example 7.4

Q strongly simulates P,
but not vice versa

Strong Simulation

Observation: sometimes, the concept of strong bisimulation is too strong (example: extending a system by new features)

Definition 7.3 (Strong simulation)

A relation $\rho \subseteq Prc \times Prc$ is called a strong simulation if, whenever $P \rho Q$ and $P \xrightarrow{\alpha} P'$, there exists $Q' \in Prc$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \rho Q'$.
We say that Q strongly simulates P if there exists a strong simulation ρ such that $P \rho Q$.

Thus: if Q strongly simulates P, then whatever transition path P takes, Q can match it by a path which retains all of P's options.

Example 7.4

$$
\begin{array}{ccc}
 & P & \\
a \swarrow & & \searrow a \\
P_1 & & P_3 \\
b \downarrow & & \downarrow c \\
P_2 & & P_4
\end{array}
\qquad
\begin{array}{ccc}
 & Q & \\
 & \downarrow a & \\
 & Q_1 & \\
b \swarrow & & \searrow c \\
Q_2 & & Q_3
\end{array}
$$

Q strongly simulates P, but not vice versa

Strong Simulation and Bisimulation

Corollary 7.5

If $P \sim Q$, then Q strongly simulates P, and P strongly simulates Q.

Proof.

A strong bisimulation $\rho \subseteq Prc \times Prc$ for $P \sim Q$ is a strong simulation for both directions. □

Caveat: the converse does generally not hold!

Example 7.6

$$\begin{array}{ccc} & P & \\ a \swarrow & & \searrow a \\ P_1 & & P_3 \\ b \downarrow & & \\ P_2 & & \end{array} \qquad \begin{array}{c} Q \\ \downarrow a \\ Q_1 \\ \downarrow b \\ Q_2 \end{array}$$

Q simulates P and vice versa, but $P \not\sim Q$

Strong Simulation and Bisimulation

Corollary 7.5

If $P \sim Q$, then Q strongly simulates P, and P strongly simulates Q.

Proof.

A strong bisimulation $\rho \subseteq Prc \times Prc$ for $P \sim Q$ is a strong simulation for both directions. □

Caveat: the converse does generally not hold!

Example 7.6

Q simulates P and vice versa, but $P \not\sim Q$

Corollary 7.5

If $P \sim Q$, then Q strongly simulates P, and P strongly simulates Q.

Proof.

A strong bisimulation $\rho \subseteq Prc \times Prc$ for $P \sim Q$ is a strong simulation for both directions. □

Caveat: the converse does generally not hold!

Example 7.6

$$P \xrightarrow{a} P_1,\quad P \xrightarrow{a} P_3,\quad P_1 \xrightarrow{b} P_2$$

$$Q \xrightarrow{a} Q_1,\quad Q_1 \xrightarrow{b} Q_2$$

Q simulates P and vice versa, but $P \not\sim Q$