Modeling Concurrent and Probabilistic Systems

Lecture 6: Decidability 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

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 $P \rho Q$.

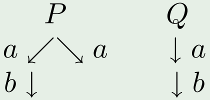
Theorem

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*

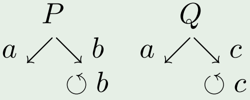
Outline

Remark: traces and deadlocks are independent in the following sense

Example 6.1



same traces
different deadlocks

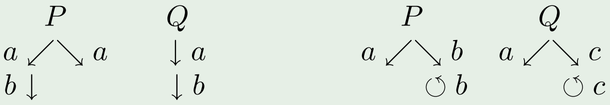


different traces
same deadlocks

But: if all traces are finite, then processes with identical deadlocks are trace equivalent (since every trace is a prefix of some deadlock)

Remark: traces and deadlocks are independent in the following sense

Example 6.1



same traces
different deadlocks

different traces
same deadlocks

But: if all traces are finite, then processes with identical deadlocks are trace equivalent (since every trace is a prefix of some deadlock)

Outline

The Problem

We now show that the word problem for strong bisimulation

Problem

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

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 6.2 (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 *is 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 6.2 (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 *is 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 6.2 (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 *is 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 6.2 (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 *is 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 6.2 (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 *is 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 6.2 (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 *is 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 6.2 (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 *is stable*

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

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

Theorem 6.2 (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 *is 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 6.2 (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 *is 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 6.3

Binary semaphore (on the board)

Proof.

(Theorem 6.2; 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 6.3

Binary semaphore (on the board)

Proof.

(Theorem 6.2; 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 6.3

Binary semaphore (on the board)

Proof.

(Theorem 6.2; on the board) □