Concurrency Theory

Winter Semester 2015/16

Lecture 12: Strong Bisimulation

Joost-Pieter Katoen and Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

http://moves.rwth-aachen.de/teaching/ws-1516/ct/

Trace Equivalence

- Trace equivalence is a possible behavioural equivalence, is a congruence, but does not preserve deadlocks.
- Main problem:

$$\alpha.(P + Q) \equiv \alpha.P + \alpha.Q,$$

whereas their deadlock behaviour in a context can differ.
- Solution: consider finer behavioural equivalences such that:

$$\alpha.(P + Q) \not\equiv \alpha.P + \alpha.Q$$

- Our (serious) attempt today: Milner's strong bisimulation.

Robin Milner (1934–2010)

Software Modeling
and Verification Chair

RWTH AACHEN
UNIVERSITY

Bisimulation

Rationale

Observation

In order for a behavioural equivalence to be deadlock sensitive, it has to take the branching structure of processes into account.

This is achieved by an equivalence that is defined according to the scheme:

Bisimulation scheme

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

Three versions will be considered in this course:

1. Strong bisimulation: ignore the special function of τ-actions
2. Weak bisimulation: treat τ-actions as invisible
3. Simulation relations: unidirectional versions of bisimulation

Concurrency Theory
Winter Semester 2015/16
Lecture 12: Strong Bisimulation

Software Modeling
and Verification Chair

RWTH AACHEN UNIVERSITY

Bisimulation

Strong Bisimulation I

A binary relation $\rho \subseteq Prc \times Prc$ is a strong bisimulation whenever for every $(P, Q) \in \rho$ and $\alpha \in Act$:

1. if $P \xrightarrow{\alpha} P'$, then there exists $Q' \in Prc$ such that $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in \rho$, and
2. if $Q \xrightarrow{\alpha} Q'$, then there exists $P' \in Prc$ such that $P \xrightarrow{\alpha} P'$ and $(P', Q') \in \rho$.

Definition 12.2 (Strong bisimilarity)

Processes P and Q are strongly bisimilar, denoted $P \sim Q$, iff there is a strong bisimulation ρ with $(P, Q) \in \rho$. Thus,

$$\sim = \bigcup \{\rho \mid \rho \text{ is a strong bisimulation}\}.$$

Relation $\sim$ is called strong bisimilarity.

Concurrency Theory
Winter Semester 2015/16
Lecture 12: Strong Bisimulation

Software Modeling
and Verification Chair

RWTHAACHEN
UNIVERSITY

Strong Bisimulation II

$$P \xrightarrow{\alpha} P'$$

$$\rho \qquad \text{can be completed to} \qquad \rho \qquad \rho$$

$$Q \qquad\qquad\qquad\qquad\qquad\qquad\qquad P \xrightarrow{\alpha} P'$$

$$Q \xrightarrow{\alpha} Q'$$

and

$$P \qquad\qquad\qquad\qquad\qquad\qquad\qquad P \xrightarrow{\alpha} P'$$

$$\rho \qquad \text{can be completed to} \qquad \rho \qquad \rho$$

$$Q \xrightarrow{\alpha} Q' \qquad\qquad\qquad\qquad Q \xrightarrow{\alpha} Q'$$

Software Modeling
and Verification Chair

RWTHAACHEN
UNIVERSITY

Bisimulation

Examples

Example 12.3 (A first example)

Claim: $P \sim Q$ where

$$P = a.P_1 + a.P_2 \qquad Q = a.Q_1 \quad \text{Proof:}$$
$$P_1 = b.P_2 \qquad Q_1 = b.Q_1$$
$$P_2 = b.P_2$$

$\rho = \{(P, Q), (P_1, Q_1), (P_2, Q_1)\}$ is a strong bisimulation

Example 12.4 (Relating a finite to an infinite-state process)

Claim: $P_0 \sim Q$ where $P_i = a.P_{i+1}$ for $i \in \mathbb{N}$ and $Q = a.Q$.

Proof: $\rho = \{(P_i, Q) \mid i \in \mathbb{N}\}$ is a strong bisimulation.

Example 12.5 (Counterexample; cf. Example 11.9)

Show on board that $CTM \not\sim CTM'$ where

$$CTM = coin.\left(\overline{coffee}.CTM + \overline{tea}.CTM\right)$$
$$CTM' = coin.\overline{coffee}.CTM' + coin.\overline{tea}.CTM'.$$

Properties of Strong Bisimilarity

Lemma 12.6 (Properties of $\sim$)

1. $\sim$ *is an equivalence relation (i.e., reflexive, symmetric, and transitive)*
2. $\sim$ *is the coarsest strong bisimulation*

Proof.

on the board ☐

Software Modeling
and Verification Chair

RWTH AACHEN UNIVERSITY

Bisimulation and Trace Equivalence

Bisimulation on Paths

Lemma 12.7 (Bisimulation on paths)

Whenever we have:

$$P_0 \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} P_2 \xrightarrow{\alpha_3} P_3 \xrightarrow{\alpha_4} P_4 \ldots \ldots$$

$$\rho$$

$$Q_0$$

this can be completed to

$$P_0 \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} P_2 \xrightarrow{\alpha_3} P_3 \xrightarrow{\alpha_4} P_4 \ldots \ldots$$

$$\rho \qquad \rho \qquad \rho \qquad \rho \qquad \rho$$

$$Q_0 \xrightarrow{\alpha_1} Q_1 \xrightarrow{\alpha_2} Q_2 \xrightarrow{\alpha_3} Q_3 \xrightarrow{\alpha_4} Q_4 \ldots \ldots$$

Proof.

by induction on the length of the path ☐

Bisimulation and Trace Equivalence

Strong Bisimulation vs. Trace Equivalence

Theorem 12.8

$P \sim Q$ implies that P and Q are trace equivalent. The reverse does generally not hold.

Proof.

The implication from left to right follows from the previous slide.

Consider the other direction.
Take $P = a.P_1$ with $P_1 = b.\text{nil} + c.\text{nil}$ and $Q = a.b.nil + a.c.\text{nil}$.
Then: $Tr(P) = \{\epsilon, a, ab, ac\} = Tr(Q)$.
Thus, P and Q are trace equivalent.
But: $P \not\sim Q$, as there is no state in the LTS of Q that is bisimilar to P_1.
Why? No state in Q can perform both b and c. □

Software Modeling
and Verification Chair

RWTH AACHEN
UNIVERSITY

Bisimulation and Trace Equivalence

Deterministic Transition Systems

Definition 12.9 (Determinism)

$P \in Prc$ is deterministic whenever for every of its states s it holds:
$$\left(s \xrightarrow{\alpha} t \text{ and } s \xrightarrow{\alpha} u \right) \quad \text{implies} \quad t = u.$$

Theorem 12.10 (Determinism implies coincidence of $\sim$ and trace equivalence) (Park)

For deterministic P and Q: $P \sim Q$ iff $Tr(P) = Tr(Q)$.

Proof.

Left as an exercise. In fact, for deterministic processes, trace equivalence, complete trace, failure trace, and ready trace equivalence all coincide. □

Software Modeling
and Verification Chair

RWTH AACHEN UNIVERSITY

Congruence

Theorem 12.11 (CCS congruence property of $\sim$)

Strong bisimilarity $\sim$ is a CCS congruence, that is, whenever $P, Q \in Prc$ such that $P \sim Q$,

$$
\begin{aligned}
\alpha.P &\sim \alpha.Q &&\text{for every action } \alpha \\
P + R &\sim Q + R &&\text{for every process } R \\
P \parallel R &\sim Q \parallel R &&\text{for every process } R \\
P \setminus L &\sim Q \setminus L &&\text{for every set } L \subseteq A \\
P[f] &\sim Q[f] &&\text{for every relabelling } f : A \to A
\end{aligned}
$$

Proof.

- for $\parallel$: on the board
- for other CCS operators: left as an exercise

Software Modeling
and Verification Chair

RWTH AACHEN UNIVERSITY

Congruence and Deadlock Sensitivity

Deadlock Sensitivity of $\sim$

Definition (Deadlock; cf. Definition 11.5)

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.

Definition (Deadlock sensitivity; cf. Definition 11.7)

Relation $\equiv \subseteq Prc \times Prc$ is deadlock sensitive whenever:

$$P \equiv Q \text{ implies } \left(\forall w \in Act^*. P \text{ has a } w\text{-deadlock iff } Q \text{ has a } w\text{-deadlock}\right).$$

Theorem 12.12

$\sim$ is deadlock sensitive.

Proof.

on the board $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad$ $\square$

Software Modeling
and Verification Chair

RWTH AACHEN UNIVERSITY

Two Buffers

Example 12.13 (One-place buffer)
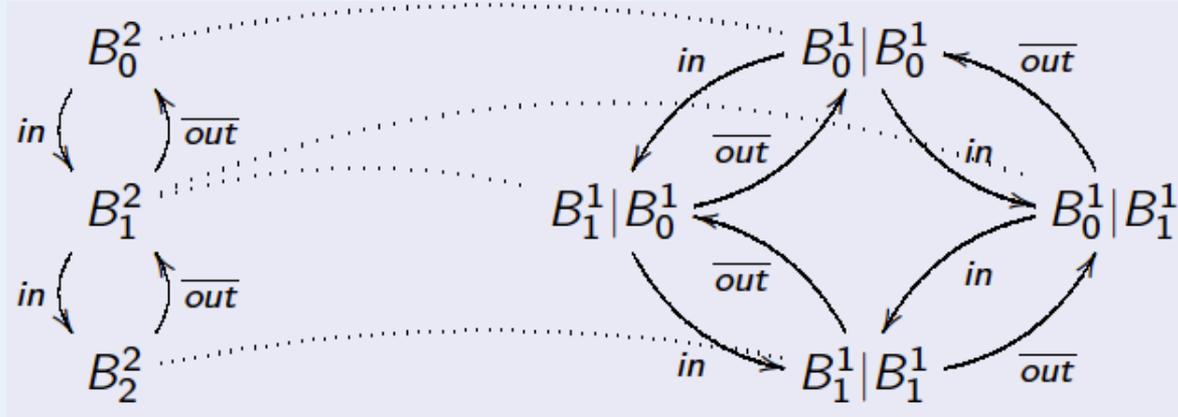
$$B_0^1 = in.B_1^1$$
$$B_1^1 = \overline{out}.B_0^1.$$

Example 12.14 (Two-place buffer)

$$B_0^2 = in.B_1^2$$
$$B_1^2 = in.B_2^2 + \overline{out}.B_0^2$$
$$B_2^2 = \overline{out}.B_1^2.$$

$B_0^2 \sim B_0^1 \parallel B_0^1$

Software Modeling
and Verification Chair

RWTH AACHEN UNIVERSITY

Buffers Revisited

Semaphores: A Generalisation

Let S_i^n stand for a semaphore for n resources i of which are taken:

$$
\begin{aligned}
S_0^n &= get.S_1^n \\
S_i^n &= get.S_{i+1}^n + put.S_{i-1}^n \quad \text{for } 0 < i < n \\
S_n^n &= put.S_{n-1}^n
\end{aligned}
$$

This process is strongly bisimilar to n parallel binary semaphores:

Lemma 12.16

For every $n \in \mathbb{N}_+$, we have: $S_0^n \sim \underbrace{S_0^1 \parallel \cdots \parallel S_0^1}_{n \text{ times}}.$

Concurrency Theory
Winter Semester 2015/16
Lecture 12: Strong Bisimulation

Software Modeling
and Verification Chair

RWTH AACHEN UNIVERSITY

Buffers Revisited

Semaphores II

> **Lemma**
>
> *For every $n \in \mathbb{N}_+$, we have: $S_0^n \sim \underbrace{S_0^1 \parallel \cdots \parallel S_0^1}_{n \text{ times}}$.*
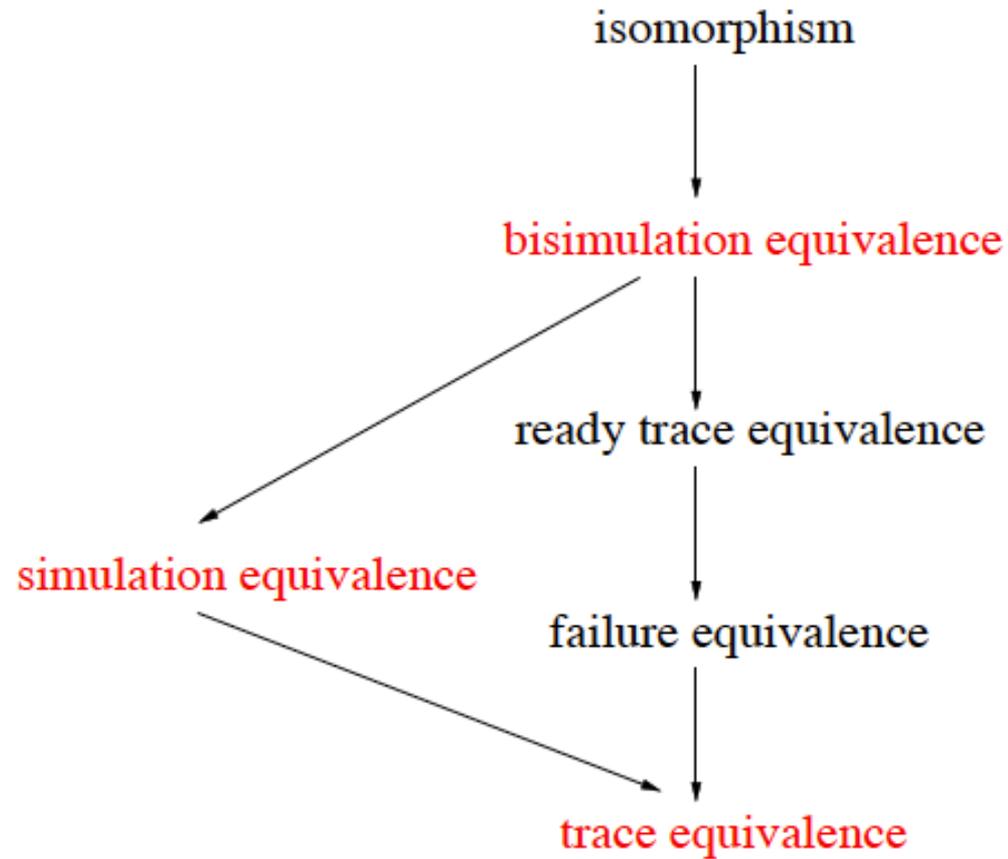
> **Proof.**
>
> Consider the following binary relation where $i_1, i_2, \ldots, i_n \in \{0, 1\}$:
>
> $$\rho = \left\{ \left(S_i^n, S_{i_1}^1 \parallel \cdots \parallel S_{i_n}^1 \right) \,\middle|\, \sum_{j=1}^{n} i_j = i \right\}$$
>
> Then: ρ is a strong bisimulation and $\left(S_0^n, \underbrace{S_0^1 \parallel \cdots \parallel S_0^1}_{n \text{ times}} \right) \in \rho$. $\qquad\square$

Software Modeling
and Verification Chair

RWTH AACHEN
UNIVERSITY

Overview of Some Behavioural Equivalences

Concurrency Theory
Winter Semester 2015/16
Lecture 12: Strong Bisimulation

Software Modeling
and Verification Chair

RWTH AACHEN UNIVERSITY

Epilogue

Summary

- Strong bisimulation of processes is based on mutually mimicking each other
- Strong bisimilarity $\sim$:
  1. is the largest strong bisimulation
  2. is an equivalence
  3. is a CCS congruence
  4. is strictly finer than trace equivalence
  5. is deadlock sensitive

Concurrency Theory
Winter Semester 2015/16
Lecture 12: Strong Bisimulation

**Software Modeling
and Verification Chair**

RWTH AACHEN
UNIVERSITY