Model Checking

Lecture #15: Bisimulation Quotienting

[Baier & Katoen, Chapter 7.2–7.6]

Joost-Pieter Katoen

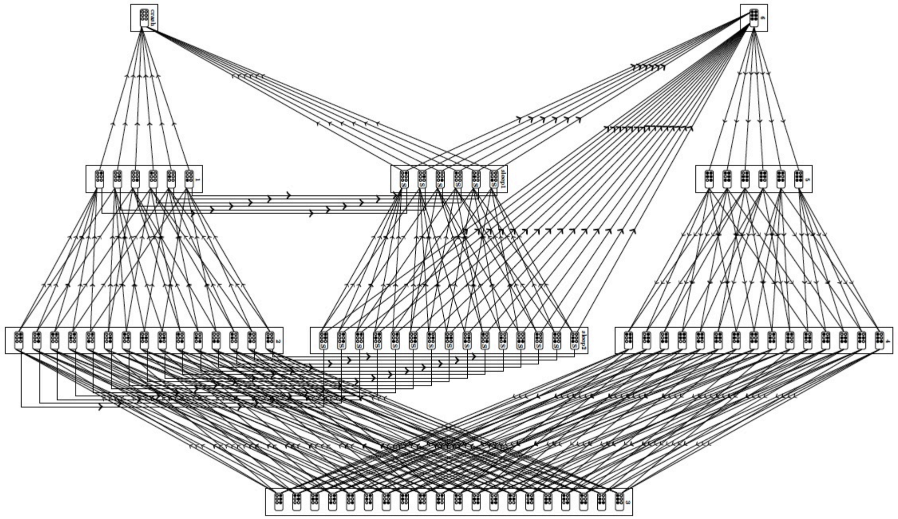
Software Modeling and Verification Group

Model Checking Course, RWTH Aachen, WiSe 2019/2020

Overview

1. Bisimulation Equivalence
2. Quotient Transition System
3. Bisimulation Quotienting
4. Simulation Pre-Order
5. Checking Simulation Pre-order

State Spaces Can Be Gigantic

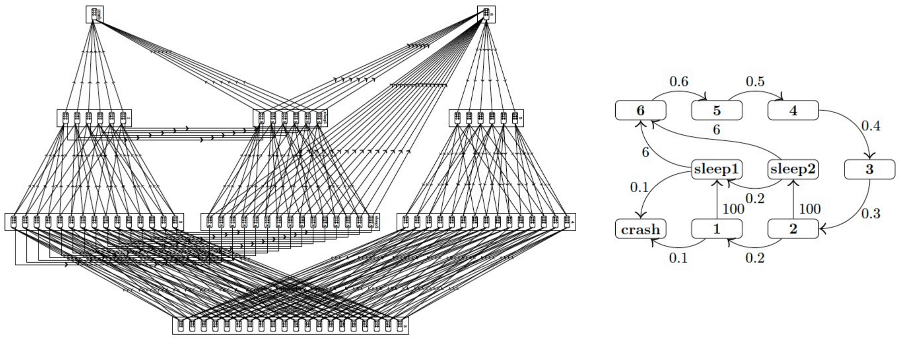


A model of the Hubble telescope

Treating Gigantic Models?

- Use compact data structures
- Make models smaller prior to (or: during) model checking
- Try to make them even smaller
- If possible, try to obtain the smallest possible model
- While preserving the properties of interest
- Do this all algorithmically and possibly fast

Abstraction



Gigantic versus smallest

Is a crash state reachable?	✔	✔
Is a failure repaired on time?	✖	✖

Abstraction

Reduce (a huge) TS to (a small) $\widehat{TS}$ prior or during model checking
Relevant issues:

- What is the formal relationship between TS and $\widehat{TS}$?
- Can $\widehat{TS}$ be obtained algorithmically and efficiently?
- Which logical fragment (of LTL, CTL, CTL*) is preserved?
- And in what sense?
  - "strong" preservation: positive and negative results carry over
  - "weak" preservation: only positive results carry over
  - "match": logic equivalence coincides with formal relation

Overview

Bisimulation

Definition: bisimulation relation

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP, L_i)$, i=1, 2, be transition systems. The symmetric relation $\mathfrak{R} \subseteq (S_1 \times S_2 \cup S_2 \times S_1)$ is a bisimulation for (TS_1, TS_2) whenever:

1. for all initial states $s_1 \in I_1$. $(s_1, s_2) \in \mathfrak{R}$ for some $s_2 \in I_2$
2. for all states $(s_1, s_2) \in \mathfrak{R}$ it holds:

   2.1 $L_1(s_1) = L_2(s_2)$, and

   2.2 $s_1' \in Post(s_1)$ implies $(s_1', s_2') \in \mathfrak{R}$ for some $s_2' \in Post(s_2)$.

Visually

$$\begin{array}{ccc} s_1 & \rightarrow & s_1' \\ \mathfrak{R} & & \\ s_2 & & \end{array} \quad \text{can be completed to} \quad \begin{array}{ccc} s_1 & \rightarrow & s_1' \\ \mathfrak{R} & & \mathfrak{R} \\ s_2 & \rightarrow & s_2' \end{array}$$

and by symmetry

$$\begin{array}{ccc} s_1 & & \\ \mathfrak{R} & & \\ s_2 & \rightarrow & s_2' \end{array} \quad \text{can be completed to} \quad \begin{array}{ccc} s_1 & \rightarrow & s_1' \\ \mathfrak{R} & & \mathfrak{R} \\ s_2 & \rightarrow & s_2' \end{array}$$

Bisimulation Equivalence

Definition: bisimulation equivalence

TS_1 and TS_2 are bisimulation equivalent (short: bisimilar), denoted $TS_1 \sim TS_2$, if there exists a bisimulation for (TS_1, TS_2). That is:

$$\sim \;=\; \bigcup \{ \mathfrak{R} \mid \mathfrak{R} \text{ is a bisimulation on } (TS_1, TS_2) \}.$$

Bisimilarity ($\sim$) is an equivalence relation.

Overview

Bisimulation on States

Definition: bisimulation/bisimilarity on states

Symmetric relation $\mathfrak{R} \subseteq S \times S$ is a bisimulation on TS (with state space S) if for any $(s_1, s_2) \in \mathfrak{R}$:

1. $L(s_1) = L(s_2)$
2. $s_1' \in Post(s_1)$ then $(s_1', s_2') \in \mathfrak{R}$ for some $s_2' \in Post(s_2)$.

The states s_1 and s_2 are bisimilar, denoted $s_1 \sim_{TS} s_2$, if $(s_1, s_2) \in \mathfrak{R}$ for some bisimulation $\mathfrak{R}$ for TS.

$s_1 \sim_{TS} s_2$ if and only if $TS_{s_1} \sim TS_{s_2}$ where TS_{s_i} denotes the transition system TS in which s_i is the only initial state.

Coarsest Bisimulation

The relation $\sim_{TS}$ is a bisimulation, an equivalence, and the coarsest bisimulation for TS.

Proof.

□

Quotient Transition System

Definition: quotient transition system

For $TS = (S, Act, \rightarrow, I, AP, L)$ and bisimulation $\sim_{TS} \subseteq S \times S$ on TS, let the quotient transition system

$$TS/\sim_{TS} \; = \; (S', \{\tau\}, \rightarrow', I', AP, L'), \quad \text{the quotient of } TS \text{ under } \sim_{TS}$$

where

- $S' = S/\sim_{TS} \; = \; \{ [s]_\sim \mid s \in S \}$ with $[s]_\sim \; = \; \{ s' \in S \mid s \sim_{TS} s' \}$
- $\rightarrow'$ is defined by: $\dfrac{s \xrightarrow{\alpha} s'}{[s]_\sim \xrightarrow{\tau}{}' [s']_\sim}$
- $I' = \{ [s]_\sim \mid s \in I \}$
- $L'([s]_\sim) = L(s)$.

Property

For every transition system TS it holds: $TS \sim TS/\sim_{TS}$.

Proof.

□

Example

(Simplified) Lamport's Bakery Algorithm

Thread 1:

```
        ......
        while true {
          ......
n1 :      x1 := x2 + 1;
w1 :      wait until(x2 = 0 || x1 < x2 ) {
c1 :        ...critical section...}
          x1 := 0;
          ......
        }
```

Thread 2:

```
        ......
        while true {
          ......
n2 :      x2 := x1 + 1;
w2 :      wait until(x1 = 0 || x2 < x1 ) {
c2 :        ...critical section...}
          x2 := 0;
          ......
        }
```

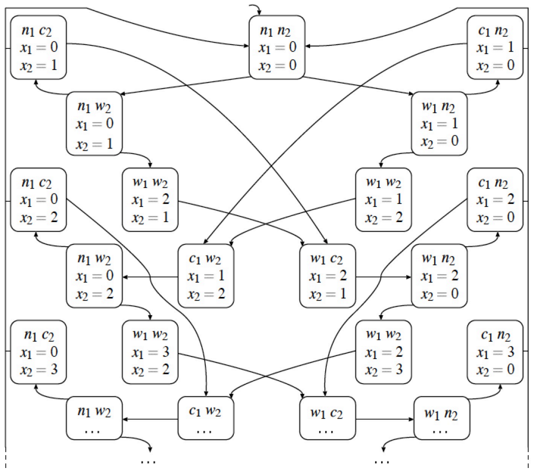
This algorithm can be applied to arbitrarily many processes

Example Bakery Algorithm Run

thread P_1	thread P_2	x_1	x_2	effect
n_1	n_2	0	0	P_1 requests access to critical section
w_1	n_2	1	0	P_2 requests access to critical section
w_1	w_2	1	2	P_1 enters the critical section
c_1	w_2	1	2	P_1 leaves the critical section
n_1	w_2	0	2	P_1 requests access to critical section
w_1	w_2	3	2	P_2 enters the critical section
w_1	c_2	3	2	P_2 leaves the critical section
w_1	n_2	3	0	P_2 requests access to critical section
w_1	w_2	3	4	P_2 enters the critical section
...	...	..	..	...

Counters may grow unboundedly large.

Bakery Algorithm Transition System



Infinite state space due to possible unbounded increase of counters

Bisimulation Relation

Let function f map a reachable state of TS_{Bak} onto a state in TS_{Bak}^{abs}

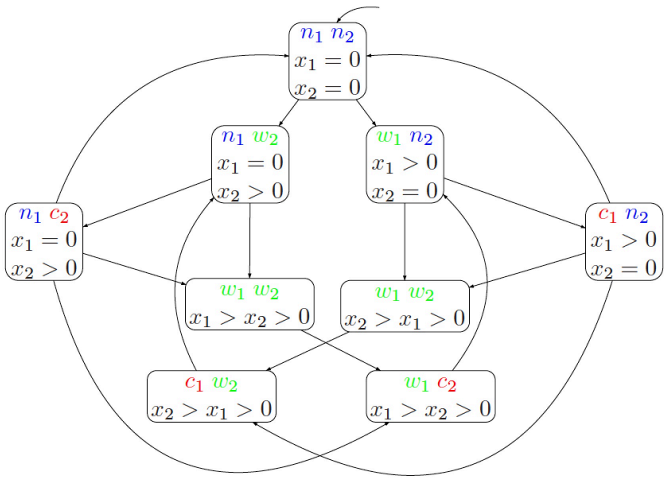
Let $s = \langle \ell_1, \ell_2, x_1 = b_1, x_2 = b_2 \rangle \in TS_{Bak}$ with $\ell_i \in \{ n_i, w_i, c_i \}$ and $b_i \in \mathbb{N}$

Then:

$$f(s) = \begin{cases} \langle \ell_1, \ell_2, x_1 = 0, x_2 = 0 \rangle & \text{if } b_1 = b_2 = 0 \\ \langle \ell_1, \ell_2, x_1 = 0, x_2 > 0 \rangle & \text{if } b_1 = 0 \text{ and } b_2 > 0 \\ \langle \ell_1, \ell_2, x_1 > 0, x_2 = 0 \rangle & \text{if } b_1 > 0 \text{ and } b_2 = 0 \\ \langle \ell_1, \ell_2, x_1 > x_2 > 0 \rangle & \text{if } b_1 > b_2 > 0 \\ \langle \ell_1, \ell_2, x_2 > x_1 > 0 \rangle & \text{if } b_2 > b_1 > 0 \end{cases}$$

It follows: $\mathfrak{R} = \{ (s, f(s)) \mid s \in S \}$ is a bisimulation for $(TS_{Bak}, TS_{Bak}^{abs})$ for any subset of $AP = \{ noncrit_i, wait_i, crit_i \mid i = 1, 2 \}$.

Quotient of Bakery Algorithm



$$TS_{Bak}^{abs} = TS_{Bak}/\sim \quad \text{for} \quad AP = \{ noncrit_i, wait_i, crit_i \mid i = 1, 2 \}$$

Overview

Partitions

- A partition $\Pi = \{ B_1, \ldots, B_k \}$ of S satisfies:
  - B_i is non-empty; B_i is called a block
  - $B_i \cap B_j = \varnothing$ for all i, j with $i \neq j$
  - $B_1 \cup \ldots \cup B_k = S$
- $C \subseteq S$ is a super-block of partition Π of S if
  $C = B_{i_1} \cup \ldots \cup B_{i_m} \quad$ for $B_{i_j} \in \Pi$ for $0 < j \leq m$
- Partition Π (of S) is finer than partition Π' (of S) if:

$$\forall B \in \Pi. \, \big(\exists B' \in \Pi'. \, B \subseteq B' \big)$$

  - each block of Π' equals the union of a set of blocks in Π
- Π is strictly finer than Π' if it is finer than Π' and $\Pi \neq \Pi'$

Partitions and Equivalences

- $\mathfrak{R}$ is an equivalence on $S \quad \Rightarrow \quad S/\mathfrak{R}$ is a partition of S
- Partition $\Pi = \{ B_1, \ldots, B_k \}$ of S induces the equivalence relation

$$\mathfrak{R}_\Pi = \{ (s, t) \mid \exists B_i \in \Pi. \, s \in B_i \wedge t \in B_i \}$$

  where it holds: $S/\mathfrak{R}_\Pi = \Pi$.

There is a one-to-one relationship between partitions and equivalences.

Partition Refinement

from now on, we assume that TS is finite

- Iteratively compute a partition of S
- Initially: Π_0 equals $\Pi_{AP} = \{ (s,t) \in S \times S \mid L(s) = L(t) \}$
- Repeat until no change: $\boxed{\Pi_{i+1} := Refine(\Pi_i)}$
  loop invariant: Π_i is coarser than $S/\sim$ and finer than $\{ S \}$
- Return Π_i
  - termination is ensured:

$$S \times S \supseteq \mathfrak{R}_{\Pi_0} \supsetneq \mathfrak{R}_{\Pi_1} \supsetneq \mathfrak{R}_{\Pi_2} \supsetneq \ldots \supsetneq \mathfrak{R}_{\Pi_i} = \sim_{TS}$$

  - time complexity: maximally $|S|$ iterations needed

Theorem

$S/\sim$ is the coarsest partition Π of S such that:

1. Π is finer than the initial partition Π_{AP}, and
2. for all $B, C \in \Pi$ it holds[1]:

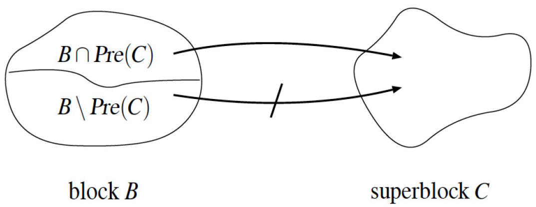
$$B \cap Pre(C) = \varnothing \text{ or } B \subseteq Pre(C).$$

Proof. □

[1] In fact, this also holds for all $B \in \Pi$ and all super-blocks C of Π.

Refinement Operator

- Let: $Refine(\Pi, C) = \bigcup_{B \in \Pi} Refine(B, C)$ for C a super-block of Π where
- $Refine(B, C) = \{ B \cap Pre(C),\ B \setminus Pre(C) \} \setminus \{\varnothing\}$



- Basic properties:
  - for Π finer than Π_{AP} and coarser than $S/\sim$:

    $Refine(\Pi, C)$ is finer than Π and $Refine(\Pi, C)$ is coarser than $S/\sim$

  - Π is strictly coarser than $S/\sim$ if and only if there exists a splitter for Π

Splitters

- Let Π be a partition of S and C a super-block of Π
- C is a splitter of Π if for some $B \in \Pi$:

$$B \cap Pre(C) \neq \varnothing \quad \text{and} \quad B \setminus Pre(C) \neq \varnothing$$

- Block B is stable wrt. C if

$$B \cap Pre(C) = \varnothing \quad \text{and} \quad B \setminus Pre(C) = \varnothing$$

- Π is stable w.r.t. C if every $B \in \Pi$ is stable wrt. C

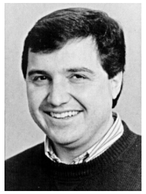
Algorithm Skeleton

Input: finite transition system *TS* over *AP* with state space S
Output: bisimulation quotient space $S/\!\sim$

$\Pi := \Pi_{AP}$;
while there exists a splitter for Π **do**
  choose a splitter C for Π;
  $\Pi :=$ *Refine*(Π, C);   (* *Refine*(Π, C) is strictly finer than Π *)
od
return Π

Splitter Selection

Scott Smolka (1954 –)



Paris Kanellakis (1953 – †1995)

Robert A. Paige (†1999)

Robert E. Tarjan (1948 –)

Which Splitter to Take?

How to determine a splitter for partition Π_{i+1}?

1. Simple strategy: $O(|S|\cdot M)$
   use any block of Π_i as splitter candidate
2. Advanced strategy: $O(\log|S|\cdot M)$
   use only "smaller" blocks of Π_i as splitter candidates
   and apply "a ternary" refinement

Advanced Selection Strategy

- Not necessary to refine with respect to all blocks $C \in \Pi_{old}$

⇒ Consider only the "smaller" subblocks of a previous refinement

- Step i: refine C' into $C_1 = C' \cap Pre(D)$ and $C_2 = C' \setminus Pre(D)$
- Step $i{+}1$: use the *smallest* $C \in \{ C_1, C_2 \}$ as splitter
  - let C be such that $|C| \leq |C'|/2$, thus $|C| \leq |C' \setminus C|$
  - combine the refinement steps with respect to C and $C' \setminus C$
- *Refine*$(\Pi, C, C' \setminus C)$ = *Refine*(*Refine*(Π, C), $C' \setminus C$) where $|C| \leq |C' \setminus C|$
  the decomposed blocks are stable with respect to C and $C' \setminus C$

The Ternary Refinement Operator

Let: $Refine(\Pi, C, C' \setminus C) = \bigcup_{B \in \Pi} Refine(B, C, C' \setminus C)$

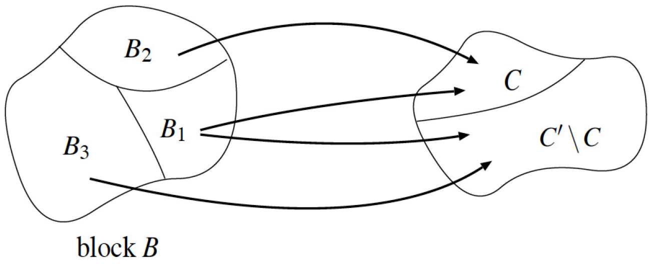
where $Refine(B, C, C' \setminus C) = \{ B_1, B_2, B_3 \} \setminus \{ \varnothing \}$ with:

$B_1 = B \cap Pre(C) \cap Pre(C' \setminus C)$ to both C and $C \setminus C'$

$B_2 = (B \cap Pre(C)) \setminus Pre(C' \setminus C)$ only to C

$B_3 = (B \cap Pre(C' \setminus C)) \setminus Pre(C)$ only to $C' \setminus C$

$\Rightarrow$ blocks B_1, B_2, B_3 are stable with respect to C and $C' \setminus C$



Quotienting Algorithm

Input: finite transition system TS with state space S
Output: bisimulation quotient space $S/\!\sim$

$\Pi_{old} := \{ S \}$;
$\Pi := Refine(\Pi_{AP}, S)$;

(* loop invariant: Π is coarser than $S/\!\sim$ and finer than Π_{AP} and Π_{old}, *)
(* and Π is stable with respect to any block in Π_{old} *)

repeat
choose block $C' \in \Pi_{old} \setminus \Pi$ and block $C \in \Pi$ with $C \subseteq C'$ and $|C| \leqslant \frac{|C'|}{2}$;
$\Pi := Refine(\Pi, C, C' \setminus C)$;
$\Pi_{old} := \Pi_{old} \setminus \{ C' \} \cup \{ C, C' \setminus C \}$;
until $\Pi = \Pi_{old}$
return Π

Complexity

The bisimulation quotient of finite transition system TS can be computed in $O(N \cdot \log M)$ where N and M are the number of states and transitions in TS respectively.

Checking bisimilarity is PTIME-complete.

Proof.

Reduction from the direct circuit value problem. Outside the scope of this lecture. □

Overview

Simulation Relation

Definition: simulation relation

Relation $\mathfrak{R} \subseteq S \times S$ is a simulation relation on TS if for any $(s_1, s_2) \in \mathfrak{R}$:

- $L(s_1) = L(s_2)$, and
- if $s_1' \in Post(s_1)$ then $(s_1', s_2') \in \mathfrak{R}$ for some $s_2' \in Post(s_2)$.

State s_2 simulates s_1, written $s_1 \preceq_{TS} s_2$ if $(s_1, s_2) \in \mathfrak{R}$ for some simulation relation $\mathfrak{R}$ on TS.

$TS_1 \preceq TS_2$ iff $\forall s_1 \in I_1. \exists s_2 \in I_2. s_1 \preceq_{TS_1 \oplus TS_2} s_2$.

$\preceq_{TS}$ is a preorder and the coarsest simulation for TS.

Visually

s_1	$\rightarrow$	s_1'		s_1	$\rightarrow$	s_1'
$\mathfrak{R}$			can be completed to	$\mathfrak{R}$		$\mathfrak{R}$
s_2				s_2	$\rightarrow$	s_2'

but not necessarily:

s_1				s_1	$\rightarrow$	s_1'
$\mathfrak{R}$			can be completed to	$\mathfrak{R}$		$\mathfrak{R}$
s_2	$\rightarrow$	s_2'		s_2	$\rightarrow$	s_2'

Abstraction Function

Definition: abstraction function

$f : S \rightarrow \widehat{S}$ is an abstraction function if $f(s) = f(s') \Rightarrow L(s) = L(s')$.

S are "concrete" states and $\widehat{S}$ are "abstract" states, mostly $|\widehat{S}| < |S|$

Abstraction functions are useful for:

- data abstraction: abstract from values of program or control variables

  f : concrete data domain $\rightarrow$ abstract data domain

- predicate abstraction: use predicates over the program variables

  f : state $\rightarrow$ valuations of the predicates

- localization reduction: program variables are visible or invisible

  f : all variables $\rightarrow$ visible variables

Abstract Transition System

Definition: abstract transition system

For $TS = (S, Act, \rightarrow, I, AP, L)$ and abstraction function $f : S \rightarrow \widehat{S}$ let:

$$TS_f = (\widehat{S}, Act, \rightarrow_f, I_f, AP, L_f), \quad \text{the abstraction of } TS \text{ under } f$$

where

- $\rightarrow_f$ is defined by: $\dfrac{s \xrightarrow{\alpha} s'}{f(s) \xrightarrow{\alpha}_f f(s')}$
- $I_f = \{ f(s) \mid s \in I \}$ and $L_f(f(s)) = L(s)$.

The relation $\mathfrak{R} = \{ (s, f(s)) \mid s \in S \}$ is a simulation for (TS, TS_f).

Proof.

By checking all conditions of a simulation relation. Straightforward. □

Example

Simulation Equivalence

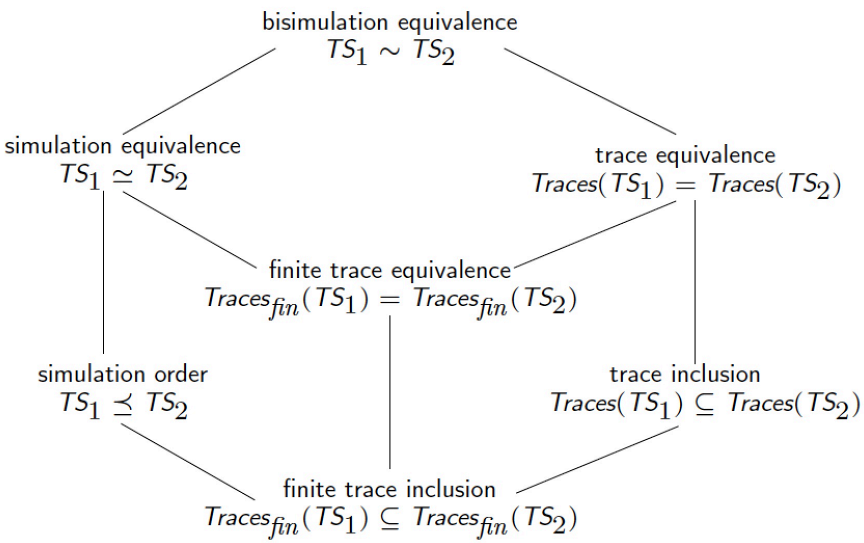
Definition: simulation equivalence

Transition systems TS_1 and TS_2 are simulation equivalent, denoted $TS_1 \simeq TS_2$ if $TS_1 \preceq TS_2$ and $TS_2 \preceq TS_1$.

1. Bisimilarity implies simulation equivalence; not the converse.
2. Simulation equivalence implies trace equivalence; not the converse.
3. For AP-deterministic[2] transition systems, simulation, bisimulation and trace equivalence coincide.

[2] TS is AP-deterministic if all initial states are labelled differently, and this also applies to all direct successors of any state in TS.

Overview



Logical Characterisation

- Negation of formulas is problematic as $\preceq_{TS}$ is not symmetric
- Let **L** be a fragment of CTL* which is closed under negation
- And assume **L** weakly matches $\preceq_{TS}$, that is:
  $s_1 \preceq_{TS} s_2$ iff for all state formulae Φ of **L**: $s_2 \models \Phi \implies s_1 \models \Phi$.
- Let $s_1 \preceq_{TS} s_2$. Then, for any state formula Φ of **L**:
  $$s_1 \models \Phi \implies s_1 \not\models \neg\Phi \implies s_2 \not\models \neg\Phi \implies s_2 \models \Phi.$$
- Hence, $s_2 \preceq_{TS} s_1$ which requires $\preceq_{TS}$ to be symmetric. Contradiction.

Universal Fragment of CTL*

Definition: universal fragment of CTL*

∀CTL* state-formulas are formed according to:

$$\Phi ::= \text{true} \;\Big|\; \text{false} \;\Big|\; a \;\Big|\; \neg a \;\Big|\; \Phi_1 \wedge \Phi_2 \;\Big|\; \Phi_1 \vee \Phi_2 \;\Big|\; \forall\varphi$$

where $a \in AP$ and φ is a path-formula. ∀CTL* path-formulas are formed according to:

$$\varphi ::= \Phi \;\Big|\; \bigcirc\varphi \;\Big|\; \varphi_1 \wedge \varphi_2 \;\Big|\; \varphi_1 \vee \varphi_2 \;\Big|\; \varphi_1 \,\mathsf{U}\, \varphi_2 \;\Big|\; \varphi_1 \,\mathsf{R}\, \varphi_2$$

where Φ is a state-formula, and φ, φ_1 and φ_2 are path-formulas.

∀CTL does not contain (general) negation and no existential path quantifier

Universal CTL* Contains LTL

For every LTL formula there exists an equivalent ∀CTL* formula.

Simulation and CTL

Theorem: Simulation equivalence, CTL and and CTL*

Let TS be a finitely branching[3] transition system and s, s' states in TS. The following statements are equivalent:

1. $s \preceq_{TS} s'$
2. for any ∀CTL*-formula Φ: $s' \models \Phi$ implies $s \models \Phi$
3. for any ∀CTL-formula Φ: $s' \models \Phi$ implies $s \models \Phi$
4. for any $\forall\text{CTL}\backslash_{\mathsf{U},\,\mathsf{R}}$-formula Φ: $s' \models \Phi$ implies $s \models \Phi$

Proof.

Along similar lines as the proof for the corresponding theorem for bisimilarity and CTL*, CTL and CTL⁻-equivalence. □

[3]This means that every state has only finitely many direct successors.

Overview

Algorithm Skeleton

Input: finite transition system *TS* over *AP* with state space S
Output: simulation order $\preceq_{TS}$

$\mathcal{R} := \{ (s_1, s_2) \mid L(s_1) = L(s_2) \};$

while $\mathcal{R}$ is not a simulation **do**
  let $(s_1, s_2) \in \mathcal{R}$ such that $s_1 \to s_1'$ and $\forall s_2'. s_2 \to s_2'$ implies $(s_1', s_2') \notin \mathcal{R}$;
  $\mathcal{R} := \mathcal{R} \setminus \{ (s_1, s_2) \};$
od
return $\mathcal{R}$

The number of iterations is bounded above by $|S|^2$, since:

$$S \times S \supseteq \mathfrak{R}_0 \supsetneq \mathfrak{R}_1 \supsetneq \mathfrak{R}_2 \supsetneq \ldots \supsetneq \mathfrak{R}_n = \preceq_{TS}$$

Algorithm

for all $s_1 \in S$ **do**
  $Sim(s_1) := \{ s_2 \in S \mid L(s_1) = L(s_2) \};$ (* initialization *)
od

while $\exists (s_1, s_2) \in S \times Sim(s_1). \exists s_1' \in Post(s_1)$ with $Post(s_2) \cap Sim(s_1') = \varnothing$ **do**
  choose such a pair of states (s_1, s_2); (* $s_1 \not\preceq_{TS} s_2$ *)
  $Sim(s_1) := Sim(s_1) \setminus \{ s_2 \};$
od
(* $Sim(s) = Sim_{TS}(s)$ for any s *)
return $\{ (s_1, s_2) \mid s_2 \in Sim(s_1) \}$

$Sim_{\mathfrak{R}}(s) = \{ s' \mid (s, s') \in \mathfrak{R} \}$, the upward closure of s under $\mathfrak{R}$

$$\varnothing \supseteq Sim_{\mathfrak{R}_0}(s) \supseteq Sim_{\mathfrak{R}_1}(s) \supseteq \ldots \supseteq Sim_{\mathfrak{R}_n}(s) = Sim_{\preceq_{TS}}(s)$$

Time complexity

The time complexity of computing $\prec_{TS}$ is $O(M \cdot N^2)$.

Proof.
In the worst case, there are N^2 iterations as their are N^2 pairs of states. For each pair of states in the worst case all transitions have to be examined. □

The best known algorithm[4] has complexity $O(M \cdot N)$. It removes several pairs in each iteration at a time and uses efficient data structures for the sets $Sim_{\mathfrak{R}}(s)$.

[4] Due to Henzinger, Henzinger and Kopke.

Next Lecture

Thursday December 19, 10:30