Simulation Quotienting

Lecture #3 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: `katoen@cs.rwth-aachen.de`

April 22, 2009

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

Current state of affairs

formal relation	trace equivalence	bisimulation
complexity	PSPACE-complete	PTIME
logical fragment	LTL	CTL^*
preservation	strong	strong match

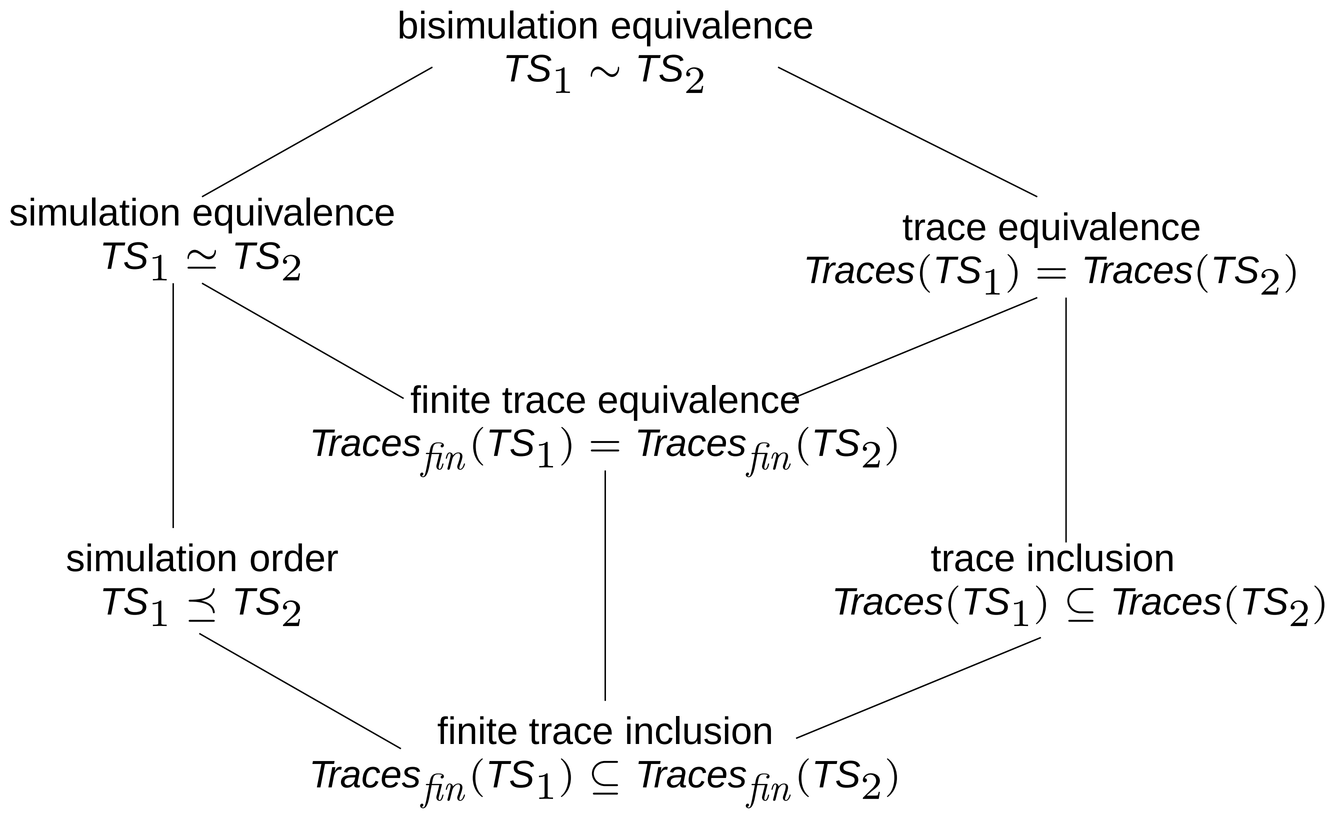
bisimulation is strictly finer than trace equivalence

Outlook of today's lecture (1)

formal relation	trace equivalence	bisimulation	simulation
complexity	PSPACE-complete	PTIME	PTIME
logical fragment	LTL	CTL^*	$\forall CTL^*$
preservation	strong	strong match	weak match

bisimulation is strictly finer than simulation equivalence

Outlook of today’s lecture (2)



Simulation order

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

- $L(s_1) = L(s_2)$
- if $s_1' \in \mathit{Post}(s_1)$ then there exists an $s_2' \in \mathit{Post}(s_2)$ with $(s_1', s_2') \in \mathcal{R}$

s_2 *simulates* s_1, denoted $s_1 \preceq_{TS} s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some simulation $\mathcal{R}$ on *TS*

Simulation order

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

but not necessarily:

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

Simulation order

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

- $L(s_1) = L(s_2)$
- if $s_1' \in \mathit{Post}(s_1)$ then there exists an $s_2' \in \mathit{Post}(s_2)$ with $(s_1', s_2') \in \mathcal{R}$

s_2 *simulates* s_1, $s_1 \preceq_{\mathit{TS}} s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some simulation $\mathcal{R}$ on *TS*

Facts: $\preceq_{\mathit{TS}}$ is a preorder and the coarsest simulation for *TS*

Simulation on paths

Whenever we have:

$$\begin{array}{ccccccccc} s_0 & \longrightarrow & s_1 & \longrightarrow & s_2 & \longrightarrow & s_3 & \longrightarrow & s_4 \ldots\ldots \\ \mathcal{R} & & & & & & & & \\ t_0 & & & & & & & & \end{array}$$

for simulation relation $\mathcal{R}$, then this can be completed to:

$$\begin{array}{ccccccccc} s_0 & \longrightarrow & s_1 & \longrightarrow & s_2 & \longrightarrow & s_3 & \longrightarrow & s_4 \ldots\ldots \\ \mathcal{R} & & \mathcal{R} & & \mathcal{R} & & \mathcal{R} & & \mathcal{R} \\ t_0 & \longrightarrow & t_1 & \longrightarrow & t_2 & \longrightarrow & t_3 & \longrightarrow & t_4 \ldots\ldots \end{array}$$

proof: by induction on the length of a path

Simulation of transition systems

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

Abstraction function

- $f : S \longrightarrow \widehat{S}$ is an *abstraction function* if $f(s) = f(s') \;\Rightarrow\; L(s) = L(s')$
  - S is a set of concrete states and $\widehat{S}$ a set of abstract states, i.e. $|\widehat{S}| \ll |S|$
- Abstraction functions are useful for:
  - data abstraction: abstract from values of program or control variables

    $$f \;:\; \text{concrete data domain} \longrightarrow \text{abstract data domain}$$

  - predicate abstraction: use predicates over the program variables

    $$f \;:\; \text{state} \longrightarrow \text{valuations of the predicates}$$

  - localization reduction: partition program variables into visible and invisible

    $$f \;:\; \text{all variables} \longrightarrow \text{visible variables}$$

Abstract transition system

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

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

where

- $\rightarrow_f$ is defined by: $$\frac{s \xrightarrow{\alpha} s'}{f(s) \xrightarrow{\alpha}_f f(s')}$$
- $I_f = \{ f(s) \mid s \in I \}$
- $L_f(f(s)) = L(s)$; for $s \in \widehat{S} \setminus f(S)$, labeling is undefined

Abstract transition system

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

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

where

- $\rightarrow_f$ is defined by: $$\frac{s \xrightarrow{\alpha} s'}{f(s) \xrightarrow{\alpha}_f f(s')}$$
- $I_f = \{\, f(s) \mid s \in I \,\}$
- $L_f(f(s)) = L(s)$; for $s \in \widehat{S} \setminus f(S)$, labeling is undefined

$\mathcal{R} = \{\, (s, f(s)) \mid s \in S \,\}$ *is a simulation for* (TS, TS_f)

Example: program abstraction

Simulation equivalence

TS_1 and TS_2 are *simulation equivalent*, denoted $TS_1 \simeq TS_2$,
if $TS_1 \preceq TS_2$ and $TS_2 \preceq TS_1$

Simulation quotient

For $\mathit{TS} = (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ and simulation equivalence $\simeq \subseteq S \times S$ let

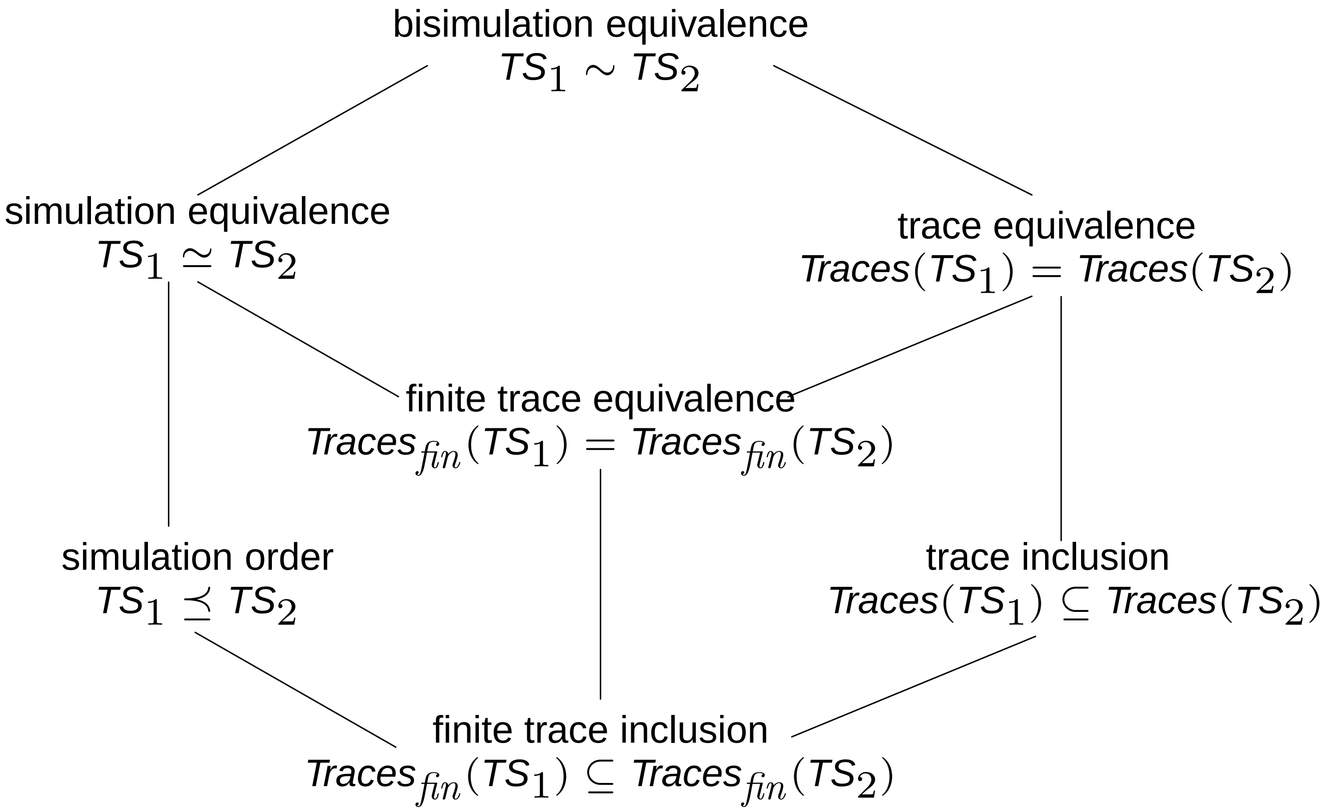
$$\mathit{TS}/\!\simeq \; = \; (S', \{\, \tau \,\}, \rightarrow', I', \mathit{AP}, L'), \qquad \text{the } \textit{quotient} \text{ of } \mathit{TS} \text{ under } \simeq$$

where

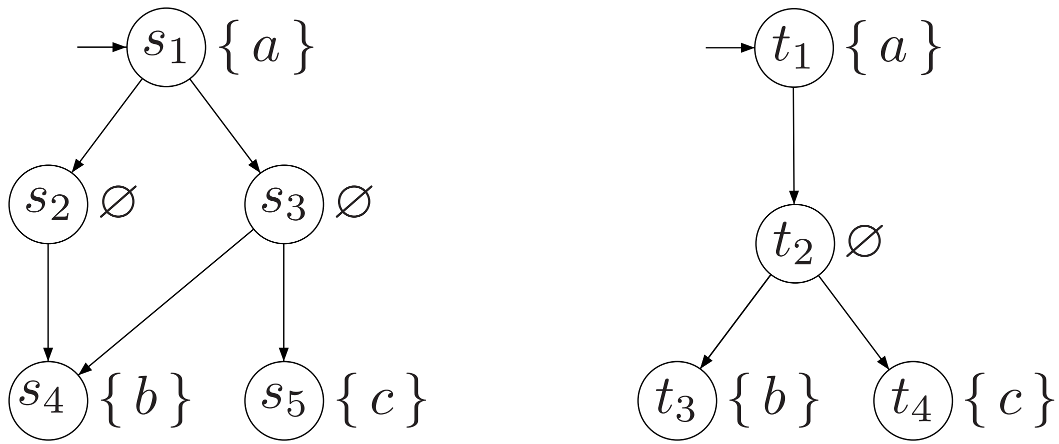
- $S' = S/\!\simeq = \{\, [s]_{\simeq} \mid s \in S \,\}$ and $I' = \{\, [s]_{\simeq} \mid s \in I \,\}$
- $\rightarrow'$ is defined by: $$\frac{s \xrightarrow{\alpha} s'}{[s]_{\simeq} \xrightarrow{\tau}{}' [s']_{\simeq}}$$
- $L'([s]_{\simeq}) = L(s)$

$\mathit{TS} \simeq \mathit{TS}/\!\simeq$; proof on blackboard

Trace, bisimulation, and simulation equivalence



Similar but not bisimilar



$TS_{left} \simeq TS_{right}$ but $TS_{left} \not\sim TS_{right}$

Simulation vs. trace equivalence

For transition systems TS_1 and TS_2 over AP:

- $TS_1 \simeq TS_2$ implies $Traces_{fin}(TS_1) = Traces_{fin}(TS_2)$
- If TS_1 and TS_2 do not have terminal states:

$$TS_1 \preceq TS_2 \quad \text{implies} \quad Traces(TS_1) \subseteq Traces(TS_2)$$

- If TS_1 and TS_2 are AP-deterministic:

$$TS_1 \simeq TS_2 \quad \text{iff} \quad Traces(TS_1) = Traces(TS_2) \quad \text{iff} \quad TS_1 \sim TS_2$$

TS is *AP*-deterministic if all initial states are labeled differently,
and this also applies to all direct successors of any state in *TS*

Simulation and safety properties

- $\mathit{TS}_1 \preceq \mathit{TS}_2$ implies $\mathit{Traces}_{fin}(\mathit{TS}_1) \subseteq \mathit{Traces}_{fin}(\mathit{TS}_2)$
- For safety LT-property P_{safe} and TS_1, TS_2 without terminal states:

$$\mathit{TS}_1 \preceq \mathit{TS}_2 \quad \text{implies} \quad (\mathit{TS}_2 \models P_{safe} \quad \text{implies} \quad \mathit{TS}_1 \models P_{safe})$$

LT property is a safety property if its violation can be shown by a finite trace

Logical characterization of $\preceq_{TS}$

- Negation of formulas is problematic as $\preceq_{TS}$ is not symmetric
- Let $\mathbf{L}$ be a fragment of CTL* which is closed under negation
- And assume $\mathbf{L}$ weakly matches $\preceq_{TS}$, that is:

$$s_1 \preceq_{TS} s_2 \quad \text{iff} \quad \text{for all state formulae } \Phi \text{ of } \mathbf{L}\text{: } s_2 \models \Phi \implies s_1 \models \Phi.$$

- Let $s_1 \preceq_{TS} s_2$. Then, for any state formula Φ of $\mathbf{L}$:

$$s_1 \models \Phi \implies s_1 \not\models \neg\Phi \implies s_2 \not\models \neg\Phi \implies s_2 \models \Phi.$$

- Hence, $s_2 \preceq_{TS} s_1$ which requires $\preceq_{TS}$ to be symmetric

Universal fragment of CTL*

$\forall$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

$\forall$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

in $\forall$CTL, the only path operators are $\bigcirc \Phi$, $\Phi_1 \, \mathsf{U} \, \Phi_2$ and $\Phi_1 \, \mathsf{R} \, \Phi_2$

Universal CTL* contains LTL

For every LTL formula there exists an equivalent $\forall$CTL* formula

Simulation order and $\forall$CTL*

Let *TS* be a finite transition system (without terminal states) and s, s' states in *TS*.

The following statements are equivalent:

(1) $s \preceq_{TS} s'$

(2) for all $\forall$CTL*-formulas Φ: $s' \models \Phi$ implies $s \models \Phi$

(3) for all $\forall$CTL-formulas Φ: $s' \models \Phi$ implies $s \models \Phi$

proof is carried out in three steps: (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (1)

Proof

Distinguishing nonsimilar transition systems

Existential fragment of CTL*

$\exists$CTL* *state-formulas* are formed according to:

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

where $a \in AP$ and φ is a path-formula

$\exists$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

in $\exists$CTL, the only path operators are $\bigcirc\Phi$, $\Phi_1 \,\mathsf{U}\, \Phi_2$ and $\Phi_1 \,\mathsf{R}\, \Phi_2$

Simulation order and $\exists$CTL*

Let *TS* be a finite transition system (without terminal states) and s, s' states in *TS*.

The following statements are equivalent:

(1) $s \preceq_{TS} s'$

(2) for all $\exists$CTL*-formulas Φ: $s \models \Phi$ implies $s' \models \Phi$

(3) for all $\exists$CTL-formulas Φ: $s \models \Phi$ implies $s' \models \Phi$

$\simeq$, $\forall$CTL*, and $\exists$CTL* equivalence

For finite transition system *TS* without terminal states:

$$\simeq_{TS} \;=\; \equiv_{\forall\text{CTL}^*} \;=\; \equiv_{\forall\text{CTL}} \;=\; \equiv_{\exists\text{CTL}^*} \;=\; \equiv_{\exists\text{CTL}}$$

But how to compute the quotient under $\simeq_{TS}$?

Basic fixpoint characterization

Consider the function $\mathcal{G} : 2^{S \times S} \longrightarrow 2^{S \times S}$:

$$\begin{array}{rcl} \mathcal{G}(\mathcal{R}) & = & \{ \quad (s,t) \mid L(s) = L(t) \;\wedge\; \forall s' \in S. \\ & & \quad \big(s \xrightarrow{\alpha} s' \;\Rightarrow\; \exists t' \in S.\, t \xrightarrow{\alpha} t' \;\wedge\; (s',t') \in \mathcal{R} \big) \\ & & \} \end{array}$$

$\preceq_{TS} = \mathcal{G}(\preceq_{TS})$ and for any $\mathcal{R}$ such that $\mathcal{G}(\mathcal{R}) = \mathcal{R}$ it holds $\mathcal{R} \subseteq \preceq_{TS}$

How to compute the fixpoint of $\mathcal{G}$?

Let $\mathit{TS} = (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ be an *image-finite* transition system

Then:

$$\preceq_{\mathit{TS}} \;=\; \bigcap_{i=0}^{\infty} \preceq_i$$

where $\preceq_i$ is defined by:

$$\begin{aligned} \preceq_0 \;&=\; \{\, (s,t) \in S \times S \mid L(s) = L(t) \,\} \\ \preceq_{i+1} \;&=\; \mathcal{G}(\preceq_i) \end{aligned}$$

this constitutes the basis for the algorithms to follow

Skeleton for simulation preorder checking

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**
  pick $(s_1, s_2) \in \mathcal{R}$ such that $s_1 \longrightarrow s_1'$, but for all s_2' with $s_2 \longrightarrow s_2'$ and $(s_1', s_2') \notin \mathcal{R}$;
  $\mathcal{R} := \mathcal{R} \setminus \{ (s_1, s_2) \};$
od
return $\mathcal{R}$

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

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

Algorithm to compute $\preceq$ (1)

Input: finite transition system *TS* over *AP* with state space S

Output: simulation order $\preceq_{TS}$

---

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 $\{ (s_1, s_2) \mid s_2 \in Sim(s_1) \}$

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

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

Time complexity

For *TS* $= (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ with $M \geqslant |S|$, the $\#$ edges in *TS*:

Time complexity of computing $\prec_{TS}$ is $\mathcal{O}\left(M \cdot |S|^3\right)$

in each iteration a single pair is deleted; can we do better?

Proof

First Observation

$$\begin{array}{ccc} s_1 & \longrightarrow & s_1' \\ \mathcal{R} & & \mathcal{R} \\ s_2 & \longrightarrow & s_2' \end{array}$$

- Assume: s_2' is the *only* successor of s_2 related to s_1' $(*)$
  - $Sim_{\mathcal{R}}(s_1') \cap Post(s_2) = \{ s_2' \}$ where $Sim_{\mathcal{R}}(s_1') = \{ s \in S \mid (s_1', s) \in \mathcal{R} \}$
- Removing (s_1', s_2') from $\mathcal{R}$ implies that $s_1 \not\preceq s_2$

$\Rightarrow$ (s_1, s_2) can thus also safely be removed from $\mathcal{R}$

- This applies to *all* direct predecessors of s_2' satisfying $(*)$

Algorithm to compute $\preceq$ (2)

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

---

for all $s_1 \in S$ **do**
  $Sim_{old}(s_1) := S$;
  $Sim(s_1) := \{ s_2 \in S \mid L(s_1) = L(s_2) \}$;
od
while ($\exists s \in S$ with $Sim_{old}(s) \neq Sim(s)$) **do**
  choose s_1' such that $Sim_{old}(s_1') \neq Sim(s_1')$;
  $Remove := Pre\big(Sim_{old}(s_1')\big) \setminus Pre\big(Sim(s_1')\big)$;   (* predecessors that $\not\preceq s_1'$ *)
  for all $s_1 \in Pre(s_1')$ **do**
    $Sim(s_1) := Sim(s_1) \setminus Remove$;
  od
  $Sim_{old}(s_1') := Sim(s_1')$;
od
return $\{ (s_1, s_2) \mid s_2 \in Sim(s_1) \}$

Implementation details

- Introduce for any state s_1' the set *Remove*(s_1')
  - contains all states s_2 to be removed from *Sim*(s_1) for $s_1 \in$ *Pre*(s_1'):

$$\textit{Remove}(s_1') \;=\; \textit{Pre}(\textit{Sim}_{old}(s_1')) \setminus \textit{Pre}(\textit{Sim}(s_1'))$$

  ⇒ the sets *Sim*$_{old}$ are superfluous
  ⇒ termination condition: *Remove*$(s_1') = \varnothing$ for all $s_1' \in S$
  - adapt the sets *Remove* on modifying *Sim*(s_1)

- Let $s_2 \in$ *Remove*(s_1') and $s_1 \in$ *Pre*(s_1')
  - then $s_1 \longrightarrow s_1'$ but no transition $s_2 \longrightarrow s_2'$ with $s_2' \in$ *Sim*(s_1')
  - then $s_1 \not\preceq s_2$, so s_2 can be removed from *Sim*(s_1):

  ⇒ extend *Remove*(s_1) with $s \in$ *Pre*(s_2) and *Post*$(s) \cap$ *Sim*$(s_1) = \varnothing$

Algorithm to compute $\preceq$ (3)

for all $s_1 \in S$ **do**
  $\mathit{Sim}(s_1) := \{ s_2 \in S \mid L(s_1) = L(s_2) \}$; (* initialization *)
  $\mathit{Remove}(s_1) := S \setminus \mathit{Pre}(\mathit{Sim}(s_1))$;
od
(* loop invariant: $\mathit{Remove}(s_1') = \mathit{Pre}\left(\mathit{Sim}_{old}(s_1')\right) \setminus \mathit{Pre}\left(\mathit{Sim}(s_1')\right)$ *)
while $(\exists s_1' \in S$ with $\mathit{Remove}(s_1') \neq \varnothing)$ **do**
  choose s_1' such that $\mathit{Remove}(s_1') \neq \varnothing$;
  for all $s_2 \in \mathit{Remove}(s_1')$ **do**
    for all $s_1 \in \mathit{Pre}(s_1')$ **do**
      if $s_2 \in \mathit{Sim}(s_1)$ **then**
        $\mathit{Sim}(s_1) := \mathit{Sim}(s_1) \setminus \{ s_2 \}$; (* $s_2 \in \mathit{Sim}_{old}(s_1) \setminus \mathit{Sim}(s_1)$ *)
        for all $s \in \mathit{Pre}(s_2)$ with $\mathit{Post}(s) \cap \mathit{Sim}(s_1) = \varnothing$ **do**
          (* $s \in \mathit{Pre}\left(\mathit{Sim}_{old}(s_1)\right) \setminus \mathit{Pre}(\mathit{Sim}(s_1))$ *)
          $\mathit{Remove}(s_1) := \mathit{Remove}(s_1) \cup \{ s \}$;
        od
      fi
    od
  od
  $\mathit{Remove}(s_1') := \varnothing$; (* $\mathit{Sim}_{old}(s_1') := \mathit{Sim}(s_1')$ *)
od
return $\{ (s_1, s_2) \mid s_2 \in \mathit{Sim}(s_1) \}$

Time complexity

For *TS* $= (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ with $M \geqslant |S|$, the $\#$ edges in *TS*:

Time complexity of computing $\prec_{TS}$ is $\mathcal{O}\big(|S| \cdot |\mathit{AP}| + M \cdot |S|\big)$

Proof

Summary

formal relation	trace equivalence	bisimulation	simulation
complexity	PSPACE-complete	$\mathcal{O}(M \cdot \log \lvert S \rvert)$	$\mathcal{O}(M \cdot \lvert S \rvert)$
logical fragment	LTL	CTL*	$\forall$CTL*
preservation	strong	strong match	weak match