Simulation Quotienting

Lecture #4 + #5 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

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

November 2+6, 2006

Simulation order

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP, L_i)$, $i=1,2$, be transition systems.

A *simulation* for (TS_1, TS_2) is a binary relation $\mathcal{R} \subseteq S_1 \times S_2$ such that:

1. $\forall s_1 \in I_1 \, \exists s_2 \in I_2. \, (s_1, s_2) \in \mathcal{R}$
2. for all states $s_1 \in S_1$, $s_2 \in S_2$ with $(s_1, s_2) \in \mathcal{R}$ it holds:
   (a) $L_1(s_1) = L_2(s_2)$
   (b) if $s_1' \in Post(s_1)$ then there exists $s_2' \in Post(s_2)$ with $(s_1', s_2') \in \mathcal{R}$

$TS_1 \preceq TS_2$ iff there exists a simulation $\mathcal{R}$ for (TS_1, TS_2)

Simulation order

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

but <u>not</u> necessarily:

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

Example

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

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

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 transition system

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

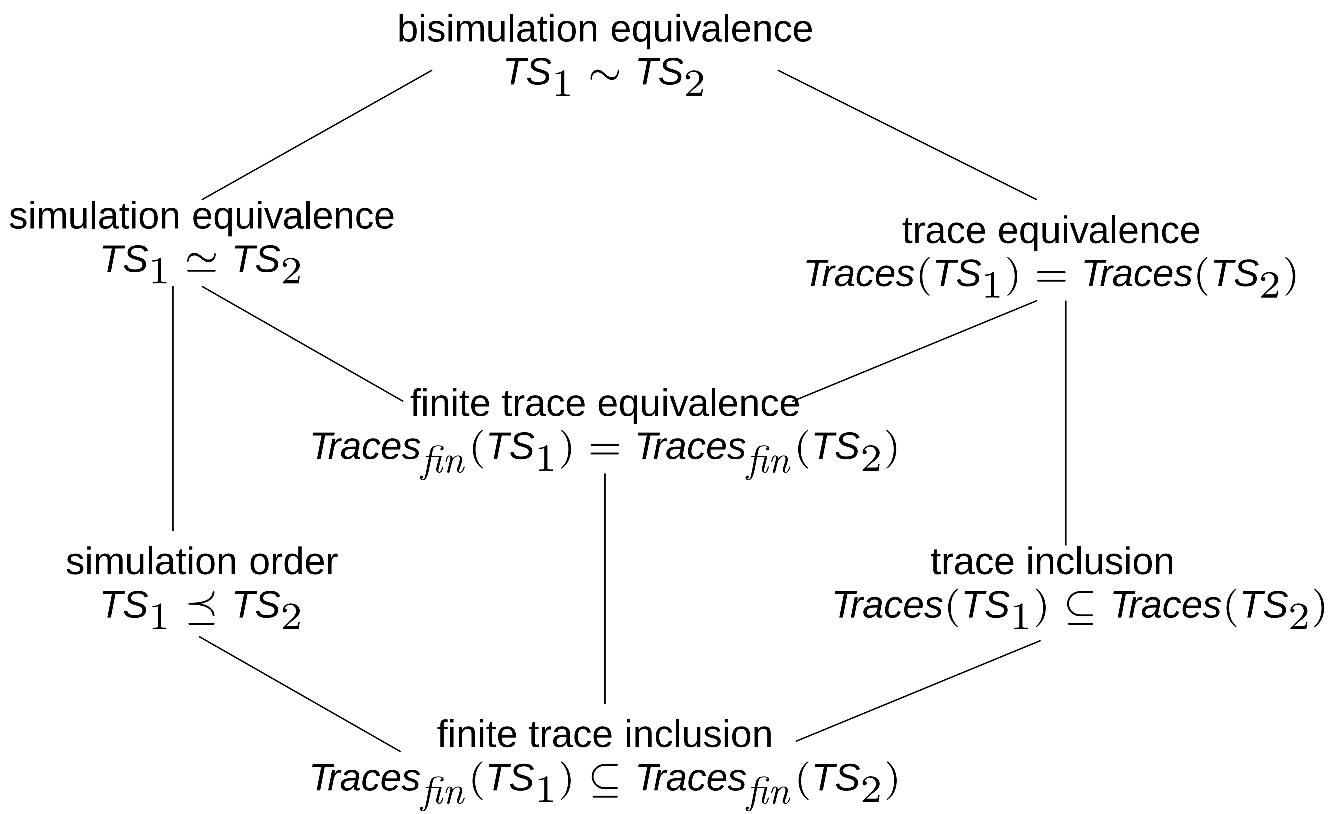
$$\mathit{TS}/\simeq \; = \; (S', \{\tau\}, \rightarrow', I', \mathit{AP}, L'), \qquad \text{the } \textit{quotient} \text{ of } \mathit{TS} \text{ under } \simeq$$

where

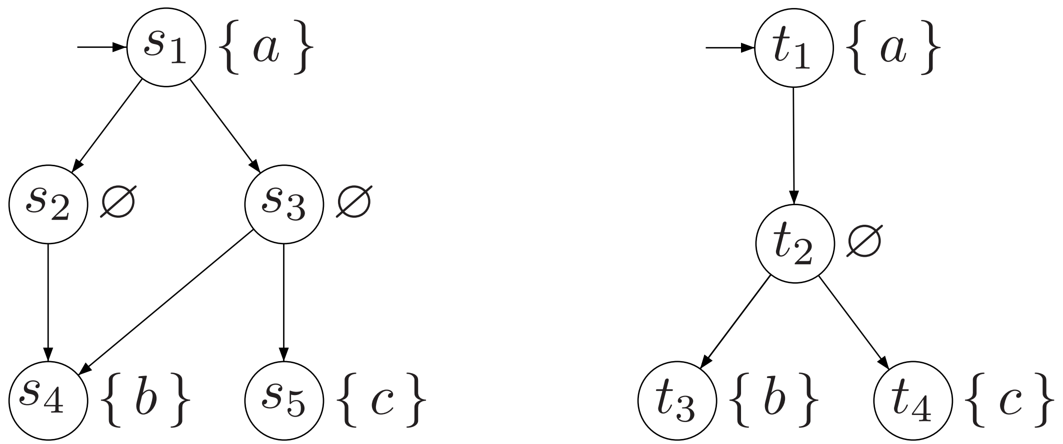
- $S' = S/\simeq = \{ [s]_\simeq \mid s \in S \}$ and $I' = \{ [s]_\simeq \mid s \in I \}$
- $\rightarrow'$ is defined by: $$\frac{s \xrightarrow{\alpha} s'}{[s]_\simeq \xrightarrow{\tau}' [s']_\simeq}$$
- $L'([s]_\simeq) = L(s)$

lemma: $\mathit{TS} \;\simeq\; \mathit{TS}/\simeq$; proof not straightforward!

Trace, bisimulation and simulation equivalence



Similar but not bisimilar



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

Terminal states and determinism

For transition systems TS_1 and TS_2 over AP:

- If TS_1 has no terminal states:

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

- If TS_1 is AP-deterministic:

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

- $TS = (S, Act, \rightarrow, I, AP, L)$ is *AP-deterministic* if:

  1. for $A \subseteq AP$: $|\, I \cap \{ s \mid L(s) = A \}| \leqslant 1$, and
  2. $s \xrightarrow{\alpha} s'$ and $s \xrightarrow{\alpha} s''$ and $L(s') = L(s'')$ implies $s' = s''$

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)

Example

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:

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

Basic fixpoint characterization

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

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

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

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

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

Then:

$$\preceq \;=\; \bigcap_{i=0}^{\infty} \preceq_i$$

where $\preceq_i$ is defined by:

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

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**
  choose $(s_1, s_2) \in \mathcal{R}$ such that $s_1 \longrightarrow s_1'$, but for all s_2' with $s_2 \longrightarrow s_2'$ and $(s_2, s_2') \not\in \mathcal{R}$;
  $\mathcal{R} := \mathcal{R} \setminus \{ (s_1, s_2) \}$
od
return $\mathcal{R}$

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

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

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

$$Sim_{\mathcal{R}}(s) = \{ s' \mid (s, s') \in \mathcal{R} \}$$

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

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 $\mathcal{O}\big(M \cdot |S|^3\big)$

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' $(*)$
  - $\mathit{Sim}_{\mathcal{R}}(s_1') \cap \mathit{Post}(s_2) = \{ s_2' \}$ where $\mathit{Sim}_{\mathcal{R}}(s) = \{ s' \in S \mid (s, s') \in \mathcal{R} \}$
- Remove (s_1', s_2') from $\mathcal{R}$ implies that $s_1 \not\preceq s_2$

$\Rightarrow$ (s_1, s_2) can thus also 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 $Sim_{old}(s_1') \neq Sim(s_1')$;
  $Remove := Pre\big(Sim_{old}(s_1')\big) \setminus Pre(Sim(s_1'))$;
  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):

$$\mathit{Remove}(s'_1) \;=\; \mathit{Pre}(\mathit{Sim}_{old}(s'_1)) \setminus \mathit{Pre}(\mathit{Sim}(s'_1))$$

$\Rightarrow$ the sets Sim_{old} are superfluous

$\Rightarrow$ loop condition: $\mathit{Remove}(s'_1) \neq \varnothing$

- Let $s_2 \in \mathit{Remove}(s'_1)$ and $s_1 \in \mathit{Pre}(s'_1)$
  - then $s_1 \longrightarrow s'_1$ but no transition $s_2 \longrightarrow s'_2$ with $s'_2 \in \mathit{Sim}(s'_1)$
  - then $s_1 \not\preceq s_2$, so s_2 can be removed from *Sim*(s_1):

$\Rightarrow$ extend $\mathit{Remove}(s_1)$ with $s \in \mathit{Pre}(s_2)$ and $\mathit{Post}(s) \cap \mathit{Sim}(s_1) = \varnothing$

Algorithm to compute $\preceq$ (3)

```
for all s1 ∈ S do
  Sim(s1) := { s2 ∈ S | L(s1) = L(s2) };                          (* initialization *)
  Remove(s1) := S \ Pre(Sim(s1));
od
        (* loop invariant: Remove(s1') = Pre (Sim_old(s1')) \ Pre (Sim(s1')) *)
while (∃s1' ∈ S with Remove(s1') ≠ ∅) do
  choose s1' such that Remove(s1') ≠ ∅;
  for all s2 ∈ Remove(s1') do
    for all s1 ∈ Pre(s1') do
      if s2 ∈ Sim(s1) then
        Sim(s1) := Sim(s1) \ { s2 };                (* s2 ∈ Sim_old(s1) \ Sim(s1) *)
        for all s ∈ Pre(s2) with Post(s) ∩ Sim(s1) = ∅ do
                                       (* s ∈ Pre (Sim_old(s1)) \ Pre(Sim(s1)) *)
          Remove(s1) := Remove(s1) ∪ { s };
        od
      fi
    od
  od
  Remove(s1') := ∅;                                 (* Sim_old(s1') := Sim(s1') *)
od
return { (s1, s2) | s2 ∈ Sim(s1) }
```

Time complexity

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

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

Proof

Checking trace equivalence

Let TS_1 and TS_2 be finite transition systems over AP. Then:

1. The problem whether

$$\mathit{Traces}_{\mathit{fin}}(TS_1) = \mathit{Traces}_{\mathit{fin}}(TS_2) \quad \text{is PSPACE-complete}$$

2. The problem whether

$$\mathit{Traces}(TS_1) = \mathit{Traces}(TS_2) \quad \text{is PSPACE-complete}$$

Proof

Overview implementation relations

	bisimulation equivalence	simulation order	trace equivalence
preservation of temporal-logical properties	CTL^* CTL	$\forall CTL^*/\exists CTL^*$ $\forall CTL/\exists CTL$	LTL (LT properties)
checking equivalence	PTIME	PTIME	PSPACE-complete
graph minimization	PTIME $\mathcal{O}(M \log \lvert S \rvert)$	PTIME $\mathcal{O}(M \cdot \lvert S \rvert)$	—