Simulation Preorder

Lecture #24 of Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

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

July 3, 2007

Overview Lecture #24

$\Rightarrow$ Simulation Order

- Simulation Equivalence
- Comparing Trace Equivalence, Bisimulation and Simulation
- Universal Fragment of CTL*

Simulation order

Let $\mathit{TS}_i = (S_i, \mathit{Act}_i, \rightarrow_i, I_i, \mathit{AP}, L_i)$, $i{=}1,2$, be transition systems.

A *simulation* for $(\mathit{TS}_1, \mathit{TS}_2)$ 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 $(s_1, s_2) \in \mathcal{R}$ it holds:
   (a) $L_1(s_1) = L_2(s_2)$
   (b) if $s_1' \in \mathit{Post}(s_1)$ then there exists $s_2' \in \mathit{Post}(s_2)$ with $(s_1', s_2') \in \mathcal{R}$

$\mathit{TS}_1 \preceq \mathit{TS}_2$ iff there exists a simulation $\mathcal{R}$ for $(\mathit{TS}_1, \mathit{TS}_2)$

Simulation order

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

but <u>not</u> necessarily:

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

Example

The use of simulations

- As a notion of correctness for *refinement*
  - $TS \preceq TS'$ whenever TS is obtained by deleting transitions from TS'
  - e.g., nondeterminism is resolved by choosing one alternative
- As a notion of correctness for *abstraction*
  - abstract from concrete values of certain program or control variables
  - use instead abstract values or ignore their value completely
  - used in e.g., software model checking of `C` and `Java`
  - formalised by an abstraction function f that maps s onto its abstraction $f(s)$

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 order on paths

Whenever we have:

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

this can be completed to

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

the proof of this fact is by induction on the length of the path

note that a finite path may be simulated by a prefix of an infinite path!

Simulation is a pre-order

$\preceq$ is a preorder, i.e., reflexive and transitive

Overview Lecture #24

- Simulation Order

⇒ Simulation Equivalence

- Comparing Trace Equivalence, Bisimulation and Simulation

- Universal Fragment of CTL^*

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 order on states

A *simulation* for $TS = (S, Act, \rightarrow, I, AP, L)$ is a binary relation $\mathcal{R} \subseteq S \times S$ such that for all $(s_1, s_2) \in \mathcal{R}$:

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

s_1 is *simulated by* s_2, denoted by $s_1 \preceq_{TS} s_2$,
if there exists a simulation $\mathcal{R}$ for TS with $(s_1, s_2) \in \mathcal{R}$

$s_1 \preceq_{TS} s_2$ if and only if $TS_{s_1} \preceq TS_{s_2}$

$s_1 \simeq_{TS} s_2$ if and only if $s_1 \preceq_{TS} s_2$ and $s_2 \preceq_{TS} s_1$

Simulation quotient transition system

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

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

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: $TS \; \simeq \; TS/\simeq$; proof not straightforward!

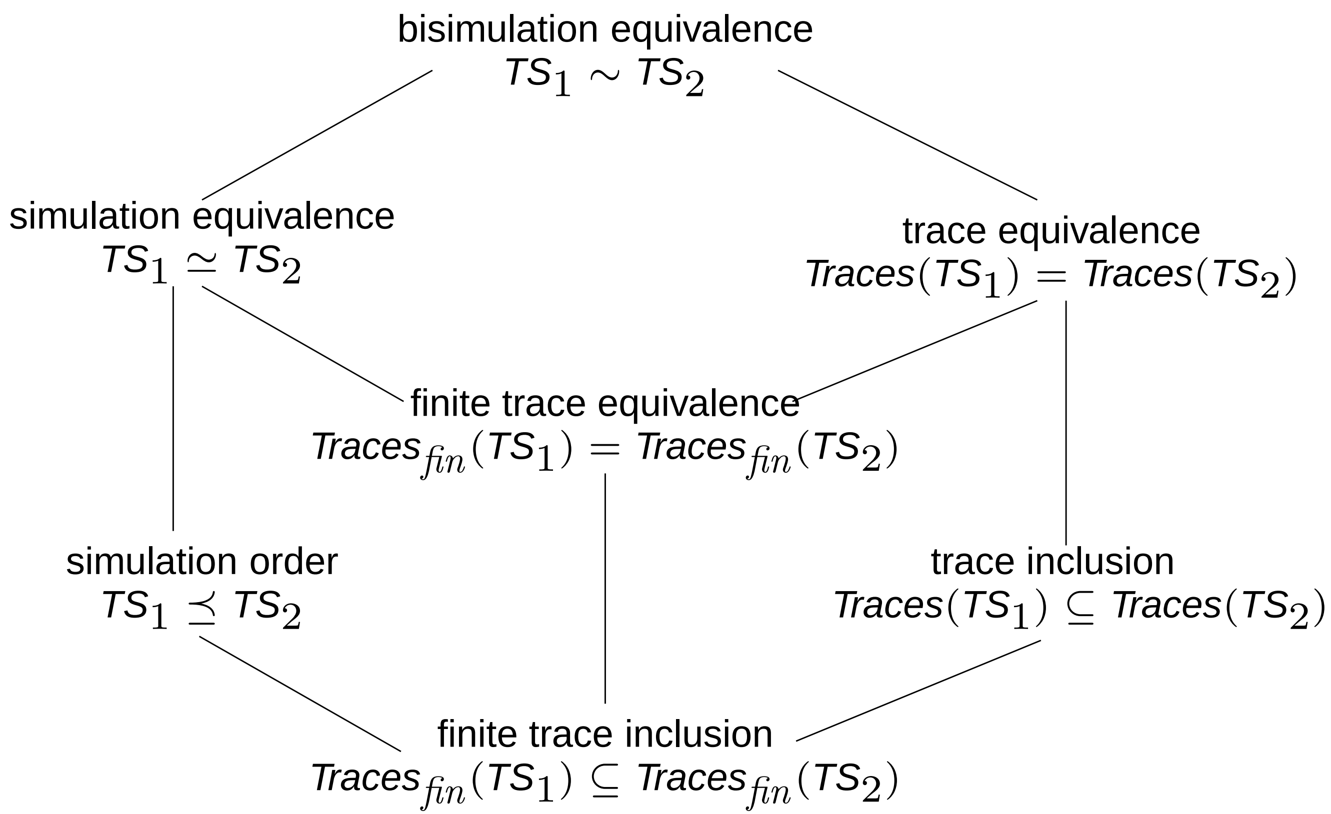
Overview Lecture #24

- Simulation Order
- Simulation Equivalence

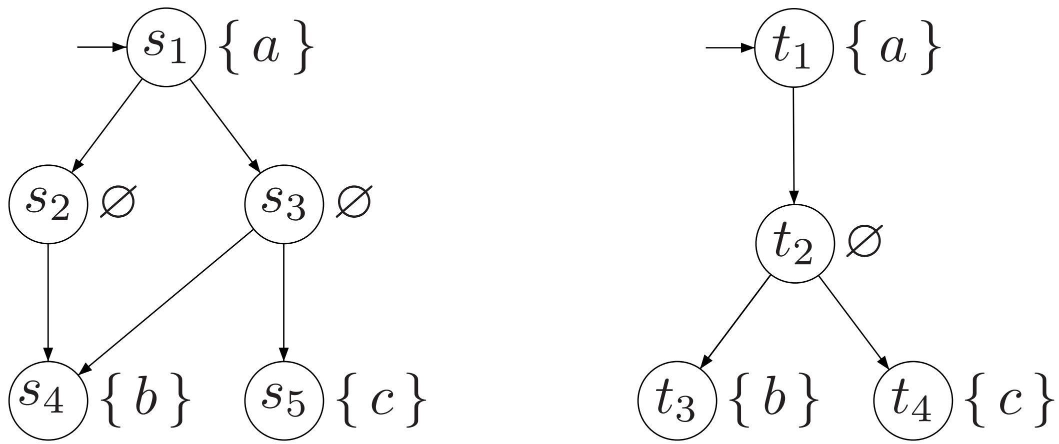
$\Rightarrow$ Comparing Trace Equivalence, Bisimulation and Simulation

- Universal Fragment of CTL*

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 $s' = s''$

Overview Lecture #24

- Simulation Order
- Simulation Equivalence
- Comparing Trace Equivalence, Bisimulation and Simulation

$\Rightarrow$ Universal Fragment of CTL*

Universal fragment of CTL*

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

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

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

$\forall$CTL* *path-formulas* are formed according to:

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

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:

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

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 $\mathcal{O}(M \cdot \lvert S \rvert)$	—