Verification

Lecture 18

Bernd Finkbeiner

UNIVERSITÄT
DES
SAARLANDES

Plan for today

- CTL*
- Bisimulation
  - Computing bisimulation quotients
- Simulation

Bisimulation vs. CTL* and CTL equivalence

Let TS be a finite state graph and s, s' states in TS

The following statements are equivalent:

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

(2) s and s' are CTL-equivalent, i.e., $s \equiv_{CTL} s'$

(3) s and s' are CTL*-equivalent, i.e., $s \equiv_{CTL^*} s'$

this is proven in three steps: $\equiv_{CTL} \subseteq \sim \subseteq \equiv_{CTL^*} \subseteq \equiv_{CTL}$

important: equivalence is also obtained for any sub-logic containing $\neg$, $\wedge$ and X

The importance of this result

- CTL and CTL* equivalence coincide
  - despite the fact that CTL* is more expressive than CTL
- Bisimilar transition systems preserve the same CTL* formulas
  - and thus the same LTL formulas (and LT properties)
- Non-bisimilarity can be shown by a single CTL (or CTL*) formula
  - $TS_1 \models \Phi$ and $TS_2 \not\models \Phi$ implies $TS_1 \not\sim TS_2$
- You even do not need to use an until-operator!
- To check $TS \models \Phi$, it suffices to check $TS/\sim \models \Phi$

Computing bisimulation quotients

Computing bisimulation quotients

A partition $\Pi = \{B_1, \ldots, B_k\}$ of S is a set of nonempty ($B_i \neq \emptyset$) and pairwise disjoint blocks B_i that decompose S ($S = \biguplus_{i=1,\ldots k} B_i$).
A partition defines an equivalence relation $\sim$
($(q, q') \in \sim \Leftrightarrow \exists B_i \in \Pi. \, q, q' \in B_i$).
Likewise, an equivalence relation $\sim$ defines a partition $\Pi = S/\sim$.
A nonempty union $C = \biguplus_{i \in I} B_i$ of blocks is called a superblock.

A block B_i of a partition Π is called stable w.r.t. a set B if either
$B_i \cap Pre(B) = \emptyset$, or $B_i \subseteq Pre(B)$.

$(Pre(B) = \{q \in S \mid Post(q) \cap B \neq \emptyset\})$

A partition Π is called stable w.r.t. a set B if all blocks of Π are.

Lemma 1. A partition Π with consistently labeled blocks is stable with respect to all of its (super)blocks if, and only if, it defines a bisimulation relation.

Partition refinement

For two partitions $\Pi = \{B_1, \ldots, B_k\}$ and $\Pi' = \{B'_1, \ldots, B'_j\}$ of S, we say that Π is finer than Π' iff every block of Π' is a superblock of Π.

For a given partition $\Pi = \{B_1, \ldots, B_k\}$, we call a (super)block C of Π a splitter of a block B_i / the partition Π if B_i / Π is not stable w.r.t. C.

Refine(B_i, C) denotes $\{B_i\}$ if B_i is stable w.r.t. C, and $\{B_i \cap Pre(C), B_i \setminus Pre(C)\}$ if C is a splitter of C.

Refine$(\Pi, C) = \biguplus_{i=1,\ldots,k}$Refine$(B_i, C)$.

Lemma 2. Refine(Π, C) is finer than Π.

An algorithm for bisimulation quotienting

Input: Transition system $(S, Act, \rightarrow, I, AP, L)$
Output: Bisimulation quotient

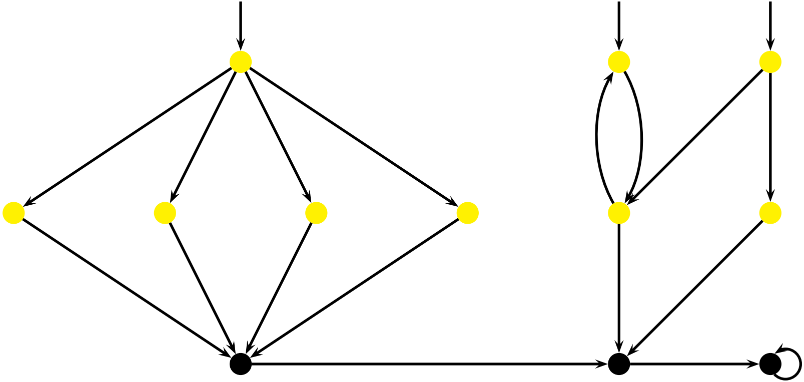
1. $\Pi = S/\sim_{AP}$ $\quad (q, q') \in \sim_{AP} \Leftrightarrow L(q) = L(q')$
2. while some block $B \in \Pi$ is a splitter of Π loop invariant: Π is coarser than $S/\sim_{TS}$
   - 2.1 pick a block B that is a splitter of Π
   - 2.2 $\Pi = \text{Refine}(\Pi, B)$
3. return Π

Example

1. $\Pi = S/{\sim_{AP}}$
2. while some block $B \in \Pi$ is a splitter of Π
   2.1 pick a block B that is a splitter of Π
   2.2 $\Pi = \text{Refine}(\Pi, B)$
3. return Π

$(q, q') \in {\sim_{AP}} \Leftrightarrow L(q) = L(q')$

loop invariant: Π is coarser than $S/{\sim_{TS}}$

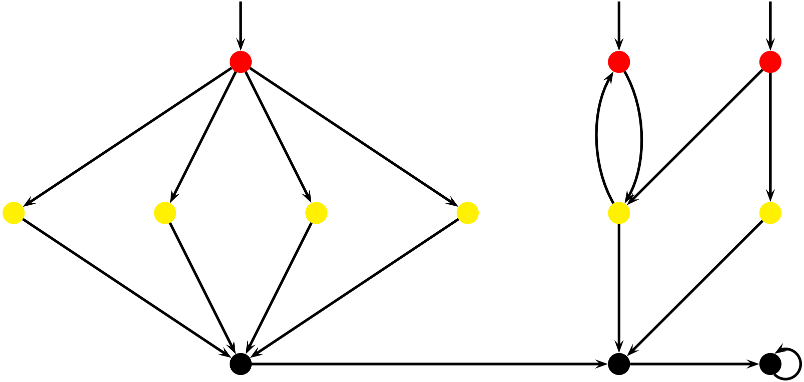


Example

1. $\Pi = S/{\sim_{AP}}$
2. while some block $B \in \Pi$ is a splitter of Π
   2.1 pick a block B that is a splitter of Π
   2.2 $\Pi = \text{Refine}(\Pi, B)$
3. return Π

$(q, q') \in {\sim_{AP}} \Leftrightarrow L(q) = L(q')$

loop invariant: Π is coarser than $S/{\sim_{TS}}$

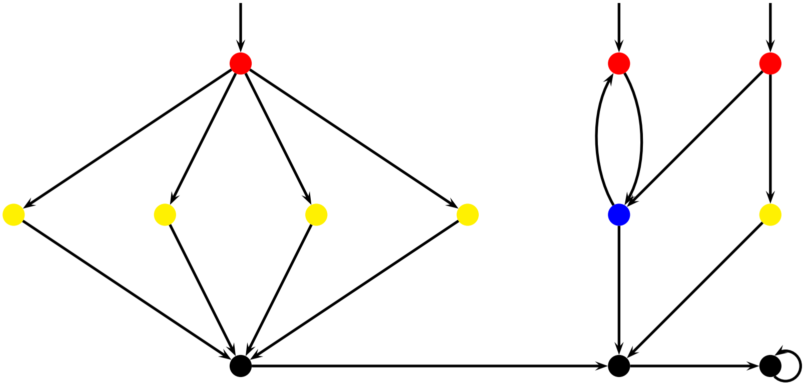


Example

1. $\Pi = S/{\sim_{AP}}$
2. while some block $B \in \Pi$ is a splitter of Π
   - 2.1 pick a block B that is a splitter of Π
   - 2.2 $\Pi = \text{Refine}(\Pi, B)$
3. return Π

$(q, q') \in \sim_{AP} \Leftrightarrow L(q) = L(q')$

loop invariant: Π is coarser than $S/{\sim_{TS}}$



Example

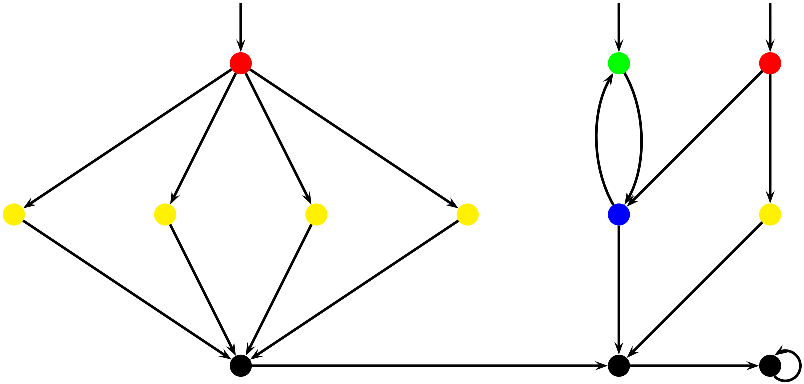
1. $\Pi = S/\sim_{AP}$

$(q, q') \in \sim_{AP} \Leftrightarrow L(q) = L(q')$

2. while some block $B \in \Pi$ is a splitter of Π

loop invariant: Π is coarser than $S/\sim_{TS}$

   2.1 pick a block B that is a splitter of Π
   2.2 $\Pi = \text{Refine}(\Pi, B)$
3. return Π

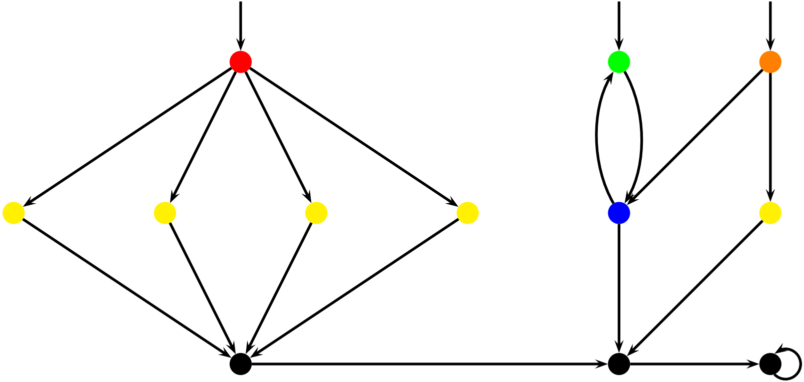


Example

1. $\Pi = S/\!\sim_{AP}$
2. while some block $B \in \Pi$ is a splitter of Π
   2.1 pick a block B that is a splitter of Π
   2.2 $\Pi = \text{Refine}(\Pi, B)$
3. return Π

$(q, q') \in \sim_{AP} \Leftrightarrow L(q) = L(q')$

loop invariant: Π is coarser than $S/\!\sim_{TS}$



Correctness and termination

1. $\Pi = S/\sim_{AP}$ $\quad (q, q') \in \sim_{AP} \Leftrightarrow L(q) = L(q')$
2. while some block $B \in \Pi$ is a splitter of Π $\quad$ loop invariant: Π is coarser than $S/\sim_{TS}$
   2.1 pick a block B that is a splitter of Π
   2.2 $\Pi = \text{Refine}(\Pi, B)$
3. return Π

Lemma 3. The algorithm terminates.
Lemma 4. The loop invariant holds initially.
Lemma 5. The loop invariant is preserved.
Theorem. The algorithm returns the quotient $S/\sim_{TS}$ of the coarsest bisimulation $\sim_{TS}$.

Simulation

Simulation order

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP, L_i)$, i=1, 2,
be two transition systems over AP.
A simulation for (TS_1, TS_2) is a binary relation $\mathcal{R} \subseteq S_1 \times S_2$ such that:

1. $\forall q_1 \in I_1 \; \exists q_2 \in I_2 . \; (q_1, q_2) \in \mathcal{R}$
2. for all $(q_1, q_2) \in \mathcal{R}$ it holds:

   2.1 $L_1(q_1) = L_2(q_2)$

   2.2 if $q_1' \in Post(q_1)$
   then there exists $q_2' \in Post(q_2)$ with $(q_1', q_2') \in \mathcal{R}$

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

Simulation order

$$\begin{array}{ccc} q_1 & \rightarrow & q_1' \\ \mathcal{R} & & \\ q_2 & & \end{array} \quad \text{can be completed to} \quad \begin{array}{ccc} q_1 & \rightarrow & q_1' \\ \mathcal{R} & & \mathcal{R} \\ q_2 & \rightarrow & q_2' \end{array}$$

but <u>not</u> necessarily:

$$\begin{array}{ccc} q_1 & & \\ \mathcal{R} & & \\ q_2 & \rightarrow & q_2' \end{array} \quad \text{can be completed to} \quad \begin{array}{ccc} q_1 & \rightarrow & q_1' \\ \mathcal{R} & & \mathcal{R} \\ q_2 & \rightarrow & q_2' \end{array}$$

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`
  - formalized by an abstraction function f that maps s onto its abstraction $f(s)$

Abstraction function

- $f : S \to \widehat{S}$ is an abstraction function if
  $f(q) = f(q') \;\Rightarrow\; L(q) = L(q')$
  - 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

    $$f : \text{concrete data domain} \to \text{abstract data domain}$$

  - predicate abstraction: use predicates over the program variables

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

  - localization reduction: partition program variables into visible and invisible

    $$f : \text{all variables} \to \text{visible variables}$$

Abstract transition system

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

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

where

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

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

Simulation order on paths

Whenever we have:

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

this can be completed to

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

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

Simulation is a pre-order

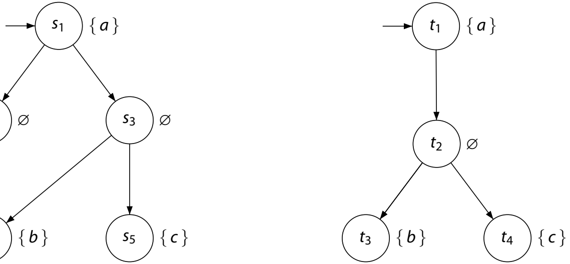
$\preceq$ is a preorder, i.e., reflexive and transitive

Simulation equivalence

TS_1 and TS_2 are simulation equivalent, denoted $TS_1 \simeq TS_2$,

if $TS_1 \preceq TS_2$ and $TS_2 \preceq TS_1$

Similar but not bisimilar



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

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 $(q_1, q_2) \in \mathcal{R}$:

1. $L(q_1) = L(q_2)$
2. if $q_1' \in Post(q_1)$
   then there exists an $q_2' \in Post(q_2)$ with $(q_1', q_2') \in \mathcal{R}$

q_1 is simulated by q_2, denoted by $q_1 \preceq_{TS} q_2$,
if there exists a simulation $\mathcal{R}$ for TS with $(q_1, q_2) \in \mathcal{R}$

$$q_1 \preceq_{TS} q_2 \quad \text{if and only if} \quad TS_{q_1} \preceq TS_{q_2}$$

$$q_1 \simeq_{TS} q_2 \quad \text{if and only if} \quad q_1 \preceq_{TS} q_2 \text{ and } q_2 \preceq_{TS} q_1$$

Simulation quotient

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

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

where

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

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