Bisimulation Minimization

Lecture #3 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

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

October 29, 2006

Implementation relations

- A *binary relation* on transition systems
  - when does a transition systems correctly implements another?
- Important for system *synthesis*
  - stepwise *refinement* of a system specification TS into an "implementation" TS'
- Important for system *analysis*
  - use the implementation relation as a means for *abstraction*
  - replace $TS \models \varphi$ by $TS' \models \varphi$ where $|\, TS' \,| \ll |\, TS \,|$ such that

$$TS \models \varphi \text{ iff } TS' \models \varphi \quad \text{or} \quad TS' \models \varphi \;\Rightarrow\; TS \models \varphi$$

$\Rightarrow$ Focus on state-based *bisimulation* and *simulation*

- definition, logical characterization, and quotienting algorithms
- variants that allow for stuttering of (internal) steps

Bisimulation

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

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

1. $\forall s_1 \in I_1 \, \exists s_2 \in I_2 . \, (s_1, s_2) \in \mathcal{R}$ and $\forall s_2 \in I_2 \, \exists s_1 \in I_1 . \, (s_1, s_2) \in \mathcal{R}$

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 $(s_1', s_2') \in \mathcal{R}$

   (c) if $s_2' \in Post(s_2)$ then there exists $s_1' \in Post(s_1)$ with $(s_1', s_2') \in \mathcal{R}$

TS_1 and TS_2 are bisimilar, denoted $TS_1 \sim TS_2$, if there exists a bisimulation for (TS_1, TS_2)

Bisimulation equivalence

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

and

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

Bisimulation on states

A *bisimulation* on $TS = (S, Act, \rightarrow, I, AP, L)$ is an equivalence relation $\mathcal{R}$ on 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 $(s_1', s_2') \in \mathcal{R}$
3. if $s_2' \in Post(s_2)$ then there exists an $s_1' \in Post(s_1)$ with $(s_1', s_2') \in \mathcal{R}$

s_1 and s_2 are *bisimilar*, denoted $s_1 \sim_{TS} s_2$,
if there exists a bisimulation $\mathcal{R}$ for TS with $(s_1, s_2) \in \mathcal{R}$

$s_1 \sim_{TS} s_2$ if and only if $TS_{s_1} \sim TS_{s_2}$

Quotient transition system

For $TS = (S, Act, \rightarrow, I, AP, L)$ and bisimulation $\sim \subseteq S \times S$ let

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

where

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

The Bakery algorithm

Process 1:

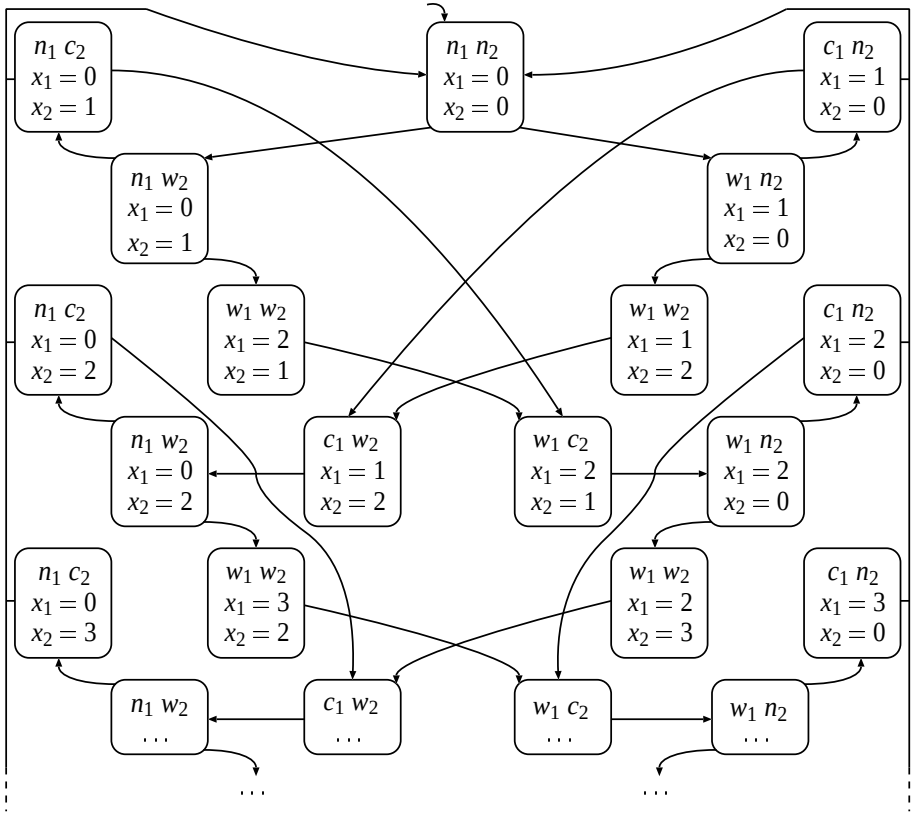
```
        ......
        while true {
            ......
n1 :        x1 := x2 + 1;
w1 :        wait until(x2 = 0 ||x1 < x2 ) {
c1 :            ... critical section ...}
            x1 := 0;
            ......
        }
```

Process 2:

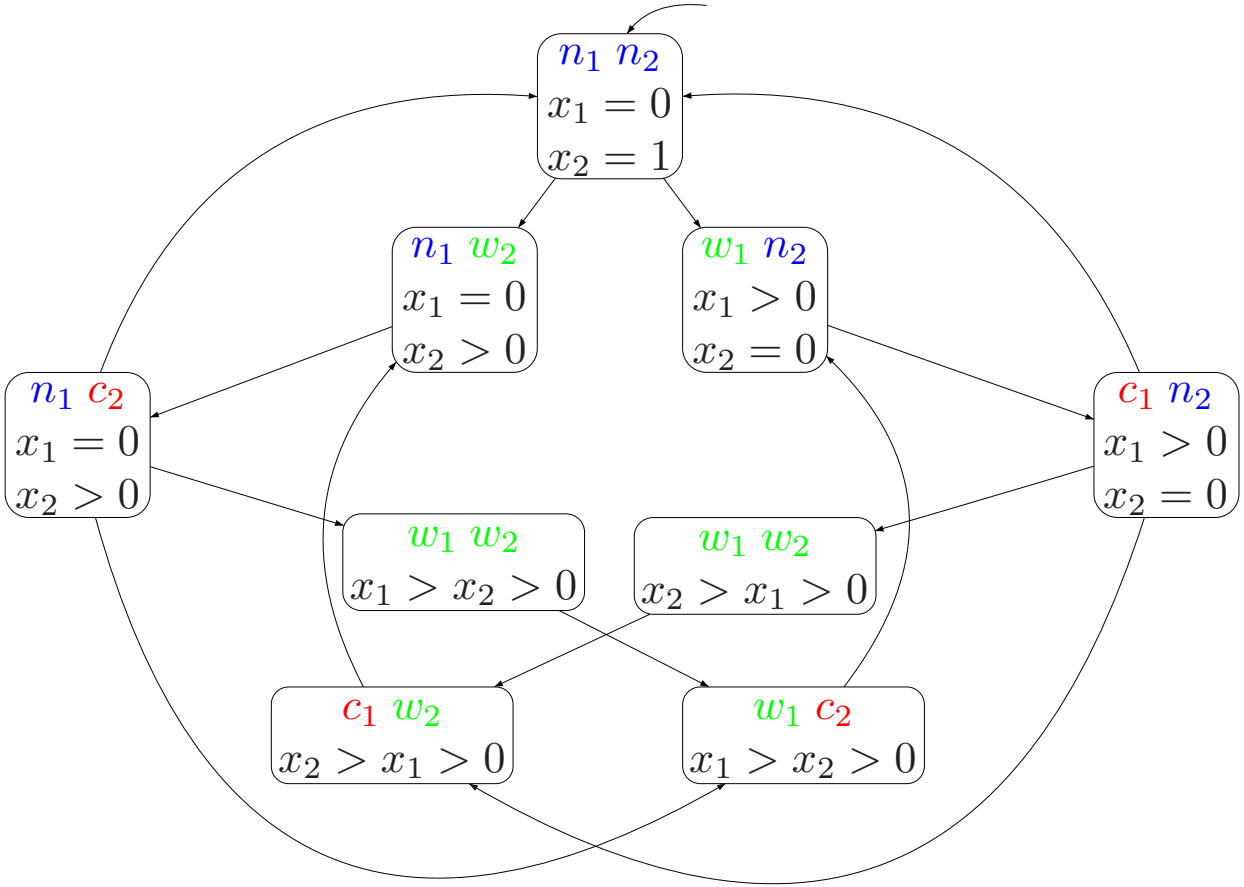
```
        ......
        while true {
            ......
n2 :        x2 := x1 + 1;
w2 :        wait until(x1 = 0 || x2 < x1) {
c2 :            ... critical section ...}
            x2 := 0;
            ......
        }
```

this algorithm can be applied to an arbitrary number of processes

Infinite transition system



Bisimulation quotient



Data abstraction

Bisimulation on paths

Whenever for bisimulation $\mathcal{R}$ 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:

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

proof: by induction on index i of state s_i

Bisimulation vs. trace equivalence

- Bisimulation is finer than trace equivalence:

$$TS_1 \sim TS_2 \quad \text{implies} \quad \mathit{Traces}(TS_1) = \mathit{Traces}(TS_2)$$

- As trace-equivalent systems satisfy the same LT-properties:

$TS_1 \sim TS_2$ implies $TS_1 \models \varphi$ iff $TS_2 \models \varphi$ for any LTL formula φ

- To check $TS \models \varphi$, it suffices to check $TS/\!\sim \models \varphi$
  - recall that $TS/\!\sim$ may be much smaller than TS!
  - this does not only apply to any LTL-formula, but to any formula in CTL*

Syntax of CTL*

CTL* *state-formulas* are formed according to:

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

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

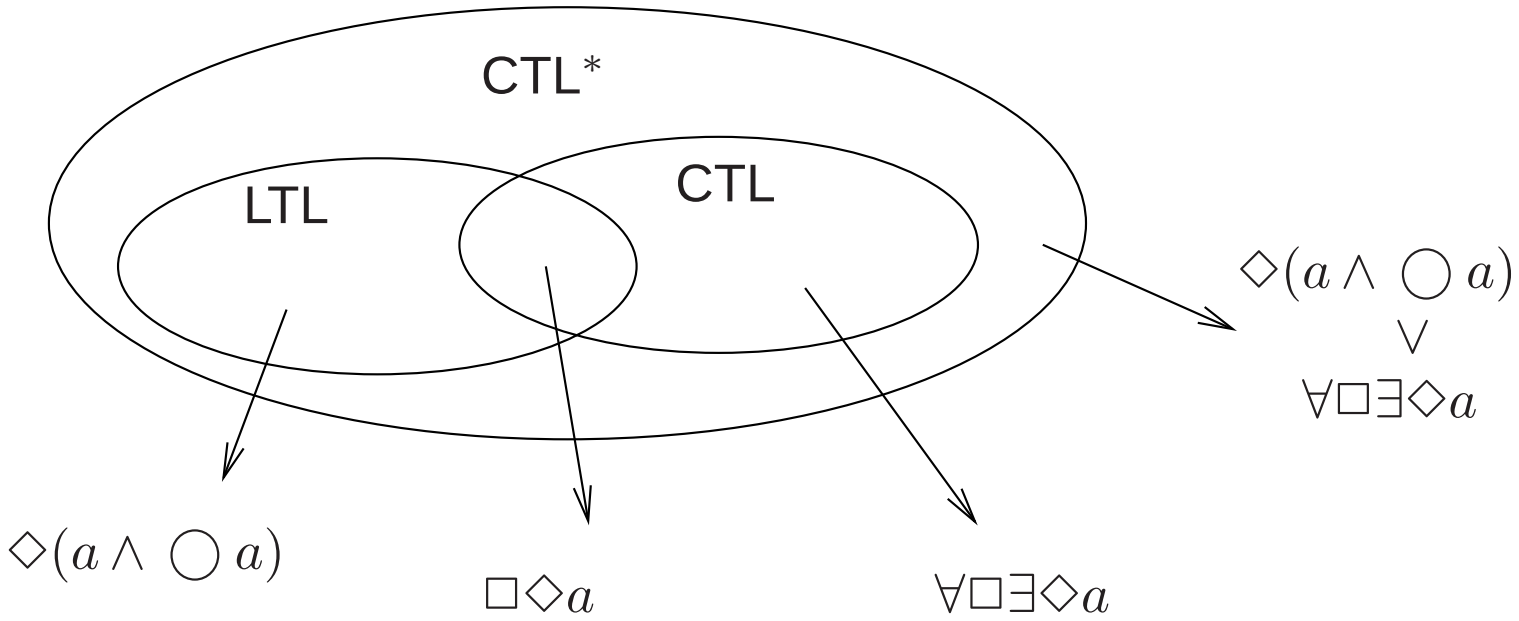
CTL* *path-formulas* are formed according to the grammar:

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

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

in CTL*: $\forall\varphi \;=\; \neg\exists\neg\varphi$. This does not hold in CTL!

Relationship between LTL, CTL and CTL*



CTL* equivalence

States s_1 and s_2 in *TS* (over *AP*) are CTL*-equivalent:

$s_1 \equiv_{\text{CTL}^*} s_2$ if and only if $(s_1 \models \Phi$ iff $s_2 \models \Phi)$

for all CTL* state formulas over *AP*

$TS_1 \equiv_{\text{CTL}^*} TS_2$ if and only if $(TS_1 \models \Phi$ iff $TS_2 \models \Phi)$

for any sublogic of CTL*, logical equivalence is defined analogously

Bisimulation vs. CTL* and CTL equivalence

Let *TS* be a *finite* transition system (without terminal states) 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., $s \equiv_{\text{CTL}} s'$

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

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

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

Example

Bisimulation vs. CTL*-equivalence

For any transition systems TS and TS' (over AP) without terminal states:

$TS \sim TS'$ if and only if $TS \equiv_{\mathrm{CTL}} TS'$ if and only if $TS \equiv_{\mathrm{CTL}^*} TS'$

$\Rightarrow$ prior to model-check Φ, it is safe to first minimize TS wrt. $\sim$

how to obtain such bisimulation quotients?

Basic fixpoint characterization

Consider the function $\mathcal{F} : 2^{S \times S} \to 2^{S \times S}$:

$$
\begin{aligned}
\mathcal{F}(\mathcal{R}) \quad = \quad \{ \quad & (s,t) \mid L(s) = L(t) \ \wedge \ \forall s' \in S. \\
& (s \to s' \ \Rightarrow \ \exists t' \in S. \, t \to t' \ \wedge \ (s',t') \in \mathcal{R}) \ \wedge \\
& (t \to s' \ \Rightarrow \ \exists u' \in S. \, s \to u' \ \wedge \ (s',u') \in R) \ \wedge \\
\} &
\end{aligned}
$$

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

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

For *finite* transition system $TS = (S, Act, \rightarrow, I, AP, L)$:

$$\sim_{TS} \;=\; \bigcap_{i=0}^{\infty} \sim_i \qquad \text{that is:} \quad s \sim_{TS} s' \text{ iff } s \sim_i s' \text{ for all } i \geqslant 0$$

where $\sim_i$ is defined by:

$$\begin{aligned} \sim_0 &= \{\, (s,t) \in S \times S \mid L(s) = L(t) \,\} \\ \sim_{i+1} &= \mathcal{F}(\sim_i) \end{aligned}$$

this constitutes the basis for the algorithms to follow

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

$$C = B_{i_1} \cup \ldots \cup B_{i_l} \quad \text{for } B_{i_j} \in \Pi \text{ for } 0 < j \leqslant l$$

- Partition Π is *finer than* partition Π' if:

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

$\Rightarrow$ each block of Π' equals the disjoint 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

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

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

- $S/\mathcal{R}_\Pi \ = \ \Pi$

$\Rightarrow$ there is a one-to-one relationship between partitions and equivalences

Skeleton for bisimulation checking

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: $\boxed{\Pi_{i+1} := \mathit{Refine}(\Pi_i)}$
  - loop invariant: Π_i is coarser than $S/\sim$ and finer than $\{ S \}$
- Return Π_i
  - termination: $\mathcal{R}_{\Pi_0} \supsetneq \mathcal{R}_{\Pi_1} \supsetneq \mathcal{R}_{\Pi_2} \supsetneq \ldots \supsetneq \mathcal{R}_{\Pi_i} = \sim_{TS}$
  - time complexity: maximally $|S|$ iterations needed

this is a partition-refinement algorithm

Computing the initial partition Π_{AP}

- Main idea: construct a *decision tree* of height k for $AP = \{ a_1, \ldots, a_k \}$
- Node at depth $i < k$ of the tree: $a_i \in L(s)$ or $a_i \notin L(s)$?
- Leaf v represents equally labeled states:
  - $s \in states(v)$ if and only if decision path for $L(s)$ leads from root to v
- Decision tree is created step-by-step
  - new nodes are created when a state is encountered with a new labeling
- Time complexity $\Theta(|S| \cdot |AP|)$
  - a single tree traversal is needed for each state

Example

Theorem

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

(i) Π is finer than the initial partition Π_{AP}, and

(ii) $B \cap \mathit{Pre}(C) = \varnothing$ or $B \subseteq \mathit{Pre}(C)$ for all $B, C \in \Pi$

If (ii) holds for Π, then it holds for all super-blocks C of Π

- No state in B has a direct successor in C, or
- All states in B have a direct successor in C

Proof

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

For *finite* transition system $TS = (S, Act, \rightarrow, I, AP, L)$:

$$\sim \quad = \quad \bigcap_{i=0}^{\infty} \sim_i$$

where $\sim_i$ is defined by:

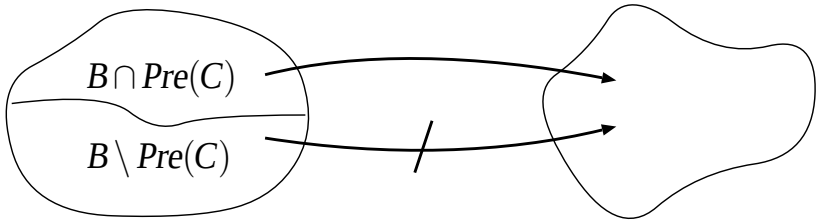
$$\begin{aligned} \sim_0 &= \{\, (s,t) \in S \times S \mid L(s) = L(t) \,\} \\ \sim_{i+1} &= \sim_i \cap \{(s,t) \mid \forall C \in S/\!\sim_i \,.\, s \in \mathit{Pre}(C) \text{ iff } t \in \mathit{Pre}(C)\} \end{aligned}$$

the block C is called a splitter

each relation $\sim_i$ is an equivalence relation

The refinement operator

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



block B superblock C

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

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

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

A partition-refinement algorithm

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

$\Pi := \Pi_{AP}$;
$\Pi_{old} := \{ S \}$; (* Π_{old} is the "previous" partition *)
(* loop invariant: Π is coarser than $S/\sim$ and finer than Π_{AP} and Π_{old} *)
repeat
  $\Pi_{old} := \Pi$;
  for all $C \in \Pi_{old}$ **do**
    $\Pi :=$ *Refine*(Π, C);
  od
until $\Pi = \Pi_{old}$
return Π

Example

Time complexity

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

The partition-refinement algorithm to compute $TS/\sim$
has a worst-case time complexity in $\mathcal{O}\big(|S| \cdot (|AP| + M)\big)$

Proof

An efficiency improvement

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

$\Rightarrow$ Consider only the “smaller” subblocks of a previous refinement

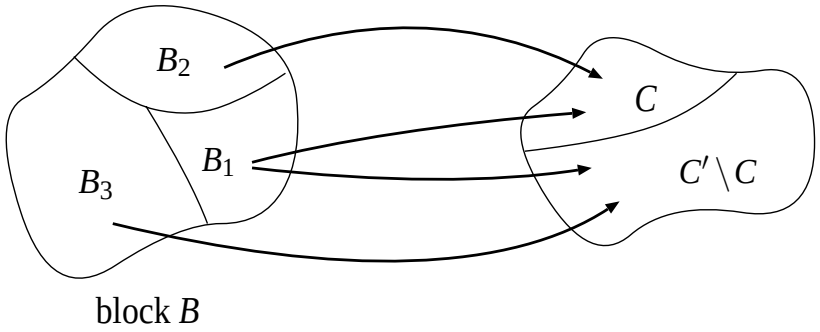
- Step i: refine C' into $C_1 = C' \cap \mathit{Pre}(D)$ and $C_2 = C' \setminus \mathit{Pre}(D)$
- Step $i{+}1$: use the *smallest* $C \in \{ C_1, C_2 \}$ as splitter candidate
  - let C be such that $|C| \leqslant |C'|/2$, thus $|C| \leqslant |C' \setminus C|$
  - combine the refinement steps with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

The new refinement operator

- Let: $\mathit{Refine}(\Pi, C, C' \setminus C) = \bigcup_{B \in \Pi} \mathit{Refine}(B, C, C' \setminus C)$
  - where $\mathit{Refine}(B, C, C' \setminus C) = \{ B_1, B_2, B_3 \} \setminus \{ \varnothing \}$ with:

$$
\begin{aligned}
B_1 &= B \cap \mathit{Pre}(C) \cap \mathit{Pre}(C' \setminus C) && \text{to both } C \text{ and } C \setminus C' \\
B_2 &= (B \cap \mathit{Pre}(C)) \setminus \mathit{Pre}(C' \setminus C) && \text{only to } C \\
B_3 &= (B \cap \mathit{Pre}(C' \setminus C)) \setminus \mathit{Pre}(C) && \text{only to } C' \setminus C
\end{aligned}
$$

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



Improved partition-refinement algorithm

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

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

(* 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} *)

while $\Pi \neq \Pi_{old}$ **do**
  choose block $C' \in \Pi_{old} \setminus \Pi$ and block $C \in \Pi$ with $C \subseteq C'$ and $|C| \leqslant \frac{|C'|}{2}$;
  $\Pi := \mathit{Refine}(\Pi, C, C' \setminus C);$
  $\Pi_{old} := \Pi_{old} \setminus \{ C' \} \cup \{ C, C' \setminus C \};$
od
return Π

Example

Implementation details

Time complexity

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

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

Proof