Bisimulation Quotienting

Lecture #2 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

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

April 17, 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

Summary of lecture #1

formal relation	trace equivalence
complexity	PSPACE-complete
logical fragment	LTL
preservation	strong

Outlook of today's lecture

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

Bisimulation

$\mathcal{R} \subseteq S \times S$ is a *bisimulation* 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 $(s_1', s_2') \in \mathcal{R}$
- if $s_2' \in \mathit{Post}(s_2)$ then there exists an $s_1' \in \mathit{Post}(s_1)$ with $(s_1', s_2') \in \mathcal{R}$

s_1 and s_2 are *bisimilar*, $s_1 \sim_{\mathit{TS}} s_2$, if $(s_1, s_2) \in \mathcal{R}$ for some bisimulation $\mathcal{R}$ for *TS*

Bisimulation

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

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 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

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

proof: by induction on the length of a path

Bisimulation of transition systems

$$\mathit{TS}_1 \sim \mathit{TS}_2 \quad \text{iff} \quad \forall s_1 \in I_1.\, \exists s_2 \in I_2.\, s_1 \sim_{\mathit{TS}} s_2$$
$$\wedge\ \forall s_2 \in I_2.\, \exists s_1 \in I_1.\, s_1 \sim_{\mathit{TS}} s_2$$

$\sim$ vs. trace equivalence

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

bisimilar transition systems thus satisfy the same LT properties!

Quotient transition system

Let $TS = (S, Act, \rightarrow, I, AP, L)$ and bisimulation $\mathcal{R} \subseteq S \times S$ be an *equivalence*

The *quotient* of TS under $\mathcal{R}$ is defined by:

$$TS/\mathcal{R} = (S', \{ \tau \}, \rightarrow', I', AP, L')$$

where

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

note that $TS \sim TS/\mathcal{R}$ Why?

Coarsest bisimulation

$\sim_{TS}$ is a bisimulation, an equivalence,

and the coarsest bisimulation for *TS*

The quotient under $\sim_{TS}$ is the smallest
under any bisimulation relation

The simplified bakery algorithm

Process 1:

$\ldots\ldots$

while true $\{$

$\ldots\ldots$

n_1 : $x_1 := x_2 + 1;$

w_1 : $\textbf{wait until}(x_2 = 0 \,||\, x_1 < x_2)\ \{$

c_1 : $\ldots$ critical section $\ldots\}$

$x_1 := 0;$

$\ldots\ldots$

$\}$

Process 2:

$\ldots\ldots$

while true $\{$

$\ldots\ldots$

n_2 : $x_2 := x_1 + 1;$

w_2 : $\textbf{wait until}(x_1 = 0 \,||\, x_2 < x_1)\ \{$

c_2 : $\ldots$ critical section $\ldots\}$

$x_2 := 0;$

$\ldots\ldots$

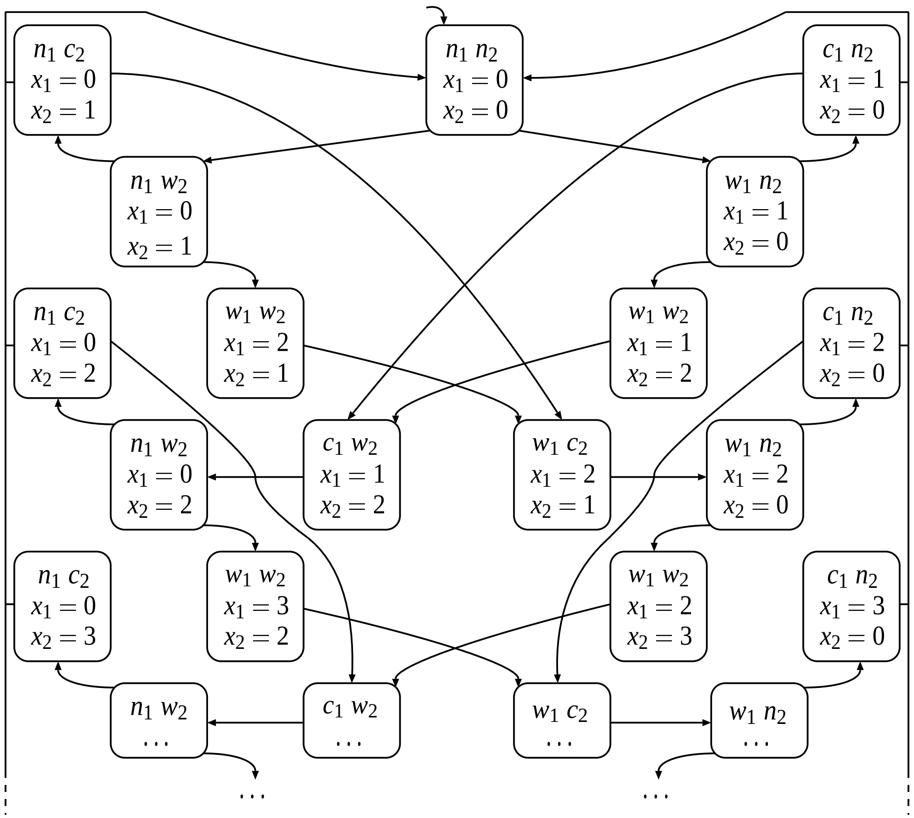
$\}$

this algorithm can be applied to arbitrarily many processes

Example path fragment

process P_1	process P_2	x_1	x_2	effect
n_1	n_2	0	0	P_1 requests access to critical section
w_1	n_2	1	0	P_2 requests access to critical section
w_1	w_2	1	2	P_1 enters the critical section
c_1	w_2	1	2	P_1 leaves the critical section
n_1	w_2	0	2	P_1 requests access to critical section
w_1	w_2	3	2	P_2 enters the critical section
w_1	c_2	3	2	P_2 leaves the critical section
w_1	n_2	3	0	P_2 requests access to critical section
w_1	w_2	3	4	P_2 enters the critical section
$\ldots$	$\ldots$	..	..	$\ldots$

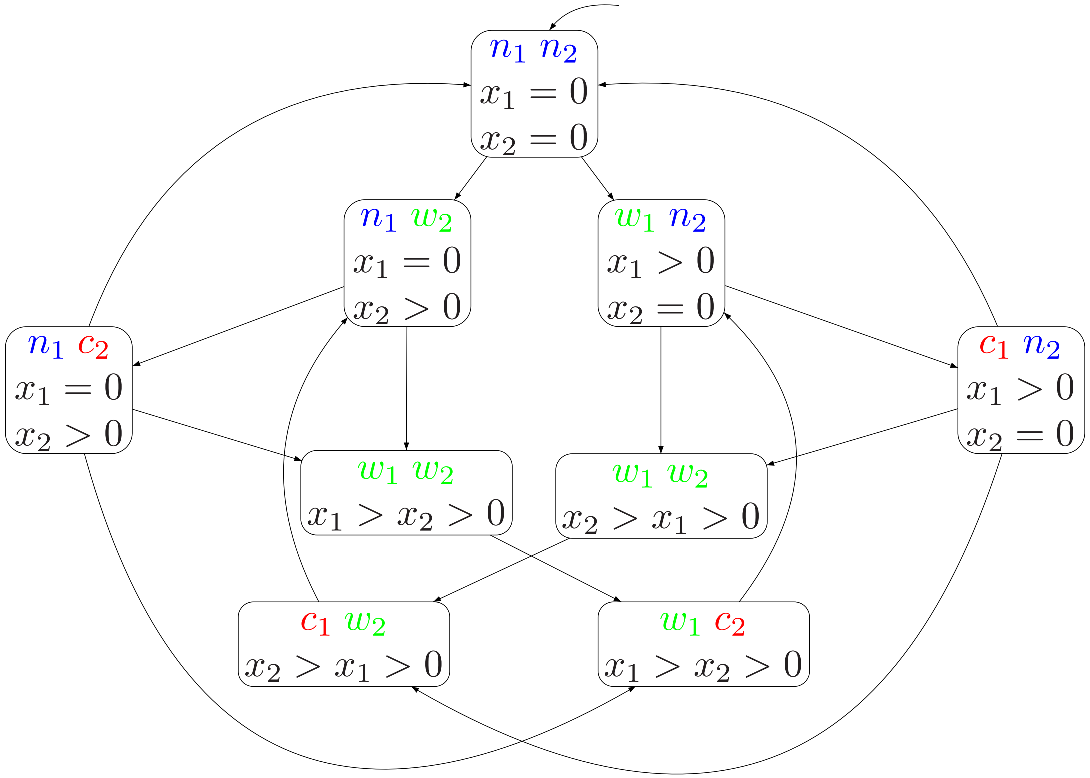
Bakery algorithm as transition system



infinite state space due to possible unbounded increase of counters

Bisimulation

Bisimulation quotient



$$TS_{Bak}^{abs} \;=\; TS_{Bak}/\mathcal{R} \quad \text{for} \quad AP = \{\, crit_1, crit_2, wait_1, wait_2 \,\}$$

Preservation of properties

- $TS_{Bak}^{abs} \models \varphi$ with, e.g.,:
  - $\Box(\neg crit_1 \lor \neg crit_2)$ and $(\Box\Diamond wait_1 \Rightarrow \Box\Diamond crit_1) \land (\Box\Diamond wait_2 \Rightarrow \Box\Diamond crit_2)$
- Since $TS_{Bak}^{abs} \sim TS_{Bak}$, it follows $Traces(TS_{Bak}^{abs}) = Traces(TS_{Bak})$
- Since $Traces(TS_{Bak}^{abs}) = Traces(TS_{Bak})$, it follows $TS_{Bak} \models \varphi$
- We thus have $Traces(TS_{Bak}^{abs}) = Traces(TS_{Bak})$

Syntax of CTL*

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

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

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

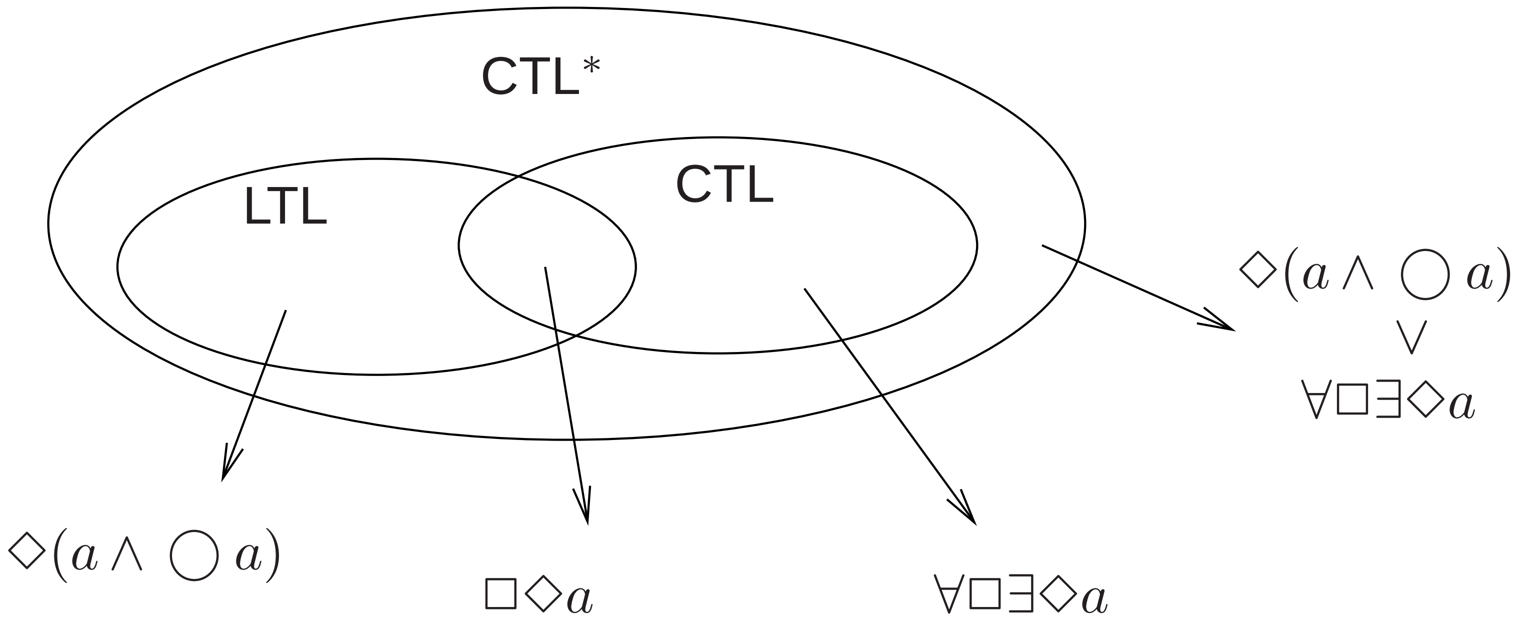
CTL* *path-formulas* are formed according to the grammar:

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

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:

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

for all CTL* state formulas over *AP*

$$\mathit{TS}_1 \equiv_{\text{CTL}^*} \mathit{TS}_2 \quad \text{if and only if} \quad (\mathit{TS}_1 \models \Phi \text{ iff } \mathit{TS}_2 \models \Phi)$$

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

$\Rightarrow$ 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 $\mathcal{F} : 2^{S \times S} \rightarrow 2^{S \times S}$:

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

$\sim_{TS} = \mathcal{F}(\sim_{TS})$ 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 $\mathit{TS} = (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$:

$$\sim_{\mathit{TS}} \;=\; \bigcap_{i=0}^{\infty} \sim_i$$ that is: $s \sim_{\mathit{TS}} s'$ iff $s \sim_i s'$ 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

  $C = B_{i_1} \cup \ldots \cup B_{i_l}$ for $B_{i_j} \in \Pi$ 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: $S \times S \supseteq \mathcal{R}_{\Pi_0} \supsetneqq \mathcal{R}_{\Pi_1} \supsetneqq \mathcal{R}_{\Pi_2} \supsetneqq \ldots \supsetneqq \mathcal{R}_{\Pi_i} = \sim_{TS}$
  - time complexity: maximally $|\, S \,|$ iterations needed (why?)

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

Lemma

1. $S/\sim$ 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 $B, C \in \Pi$

i.e., either no or all states in B have a direct successor in C

2. If (ii) holds for Π, then it holds for all $B \in \Pi$ and all superblocks C of Π

Proof

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

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

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

where $\sim_i$ is defined by:

$$\sim_0 \;=\; \{ (s,t) \in S \times S \mid L(s) = L(t) \}$$

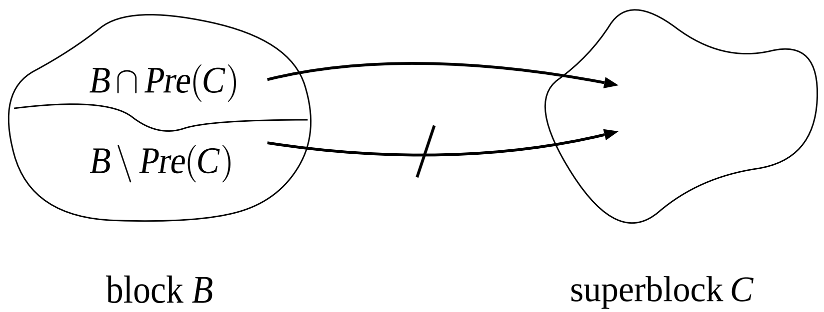
$$\sim_{i+1} \;=\; \sim_i \cap \{(s,t) \mid \forall C \in S/\sim_i .\, s \in \mathit{Pre}(C) \text{ iff } t \in \mathit{Pre}(C)\}$$

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 $\mathit{Refine}(B, C) = \left\{ B \cap \mathit{Pre}(C),\ B \setminus \mathit{Pre}(C) \right\} \setminus \{\varnothing\}$



- 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

Splitters

- Let Π be a partition of S and C a superblock of Π
- C is a splitter of Π if for some $B \in \Pi$:

$$B \cap \mathit{Pre}(C) \neq \varnothing \ \wedge \ B \setminus \mathit{Pre}(C) \neq \varnothing$$

- Block B is stable wrt. C if

$$B \cap \mathit{Pre}(C) = \varnothing \ \wedge \ B \setminus \mathit{Pre}(C) = \varnothing$$

- Π is stable wrt. C if any $B \in \Pi$ is stable wrt. C

Algorithm skeleton

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

$\Pi := \Pi_{AP}$;
while there exists a splitter for Π **do**
  choose a splitter C for Π;
  $\Pi :=$ *Refine*(Π, C); (* *Refine*(Π, C) is strictly finer than Π *)
od
return Π

Example

Which splitter to take?

How to determine a splitter for partition Π_{i+1}?

1. Simple strategy: $\mathcal{O}(|S| \cdot M)$

use any block of Π_i as splitter candidate

2. Advanced strategy: $\mathcal{O}(\log |S| \cdot M)$

use only "smaller" blocks of Π_i as splitter candidates

and apply "simultaneous" refinement

A partition-refinement algorithm

[Kanellakis & Smolka, 1983]

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

Time complexity

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

The partition-refinement algorithm to compute *TS*$/\sim$

has a worst-case time complexity in $\mathcal{O}\big(|S| \cdot |\mathit{AP}| + |S| \cdot M\big)$

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 $C_1 = C' \cap \textit{Pre}(D)$ and $C_2 = C' \setminus \textit{Pre}(D)$
- 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 $C' \setminus C$
- $\textit{Refine}(\Pi, C, C' \setminus C) = \textit{Refine}\Big(\textit{Refine}(\Pi, C),\ C' \setminus C \Big)$ where $|C| \leqslant |C' \setminus C|$
  - the decomposed blocks are stable with respect to C and $C' \setminus C$

The new refinement operator

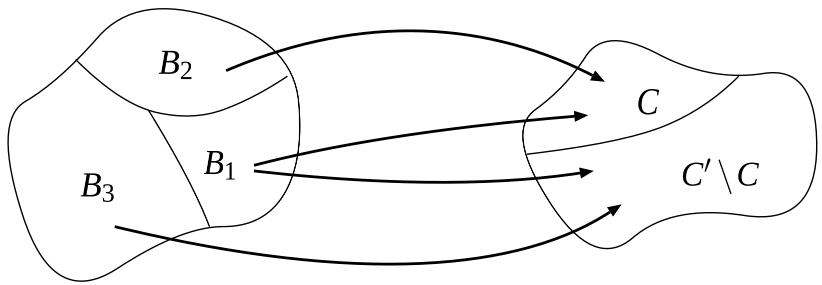
- Let: $\mathit{Refine}(\Pi, C, C' \setminus C) \;=\; \bigcup_{B \in \Pi} \mathit{Refine}(B, C, C' \setminus C)$
  - where $\mathit{Refine}(B, C, C' \setminus C) \;=\; \{\, B_1, B_2, B_3 \,\} \setminus \{\, \varnothing \,\}$ with:

$$B_1 \;=\; B \cap \mathit{Pre}(C) \cap \mathit{Pre}(C' \setminus C) \quad \text{to both } C \text{ and } C \setminus C'$$

$$B_2 \;=\; (B \cap \mathit{Pre}(C)) \setminus \mathit{Pre}(C' \setminus C) \quad \text{only to } C$$

$$B_3 \;=\; (B \cap \mathit{Pre}(C' \setminus C)) \setminus \mathit{Pre}(C) \quad \text{only to } C' \setminus C$$

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



Improved partition-refinement algorithm

[Paige & Tarjan, 1987]

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} *)

repeat
  choose block $C' \in \Pi_{old} \setminus \Pi$ and block $C \in \Pi$ with $C \subseteq C'$ and $|C| \leqslant \frac{|C'|}{2}$;
  $\Pi_{old} := \Pi$;
  $\Pi := \mathit{Refine}(\Pi, C, C' \setminus C)$;
until $\Pi = \Pi_{old}$
return Π

Example

Time complexity

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

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

Proof

Summary of today's lecture

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