Model Checking

Bisimulation Quotienting

[Baier & Katoen, Chapter 7.2+7.3]

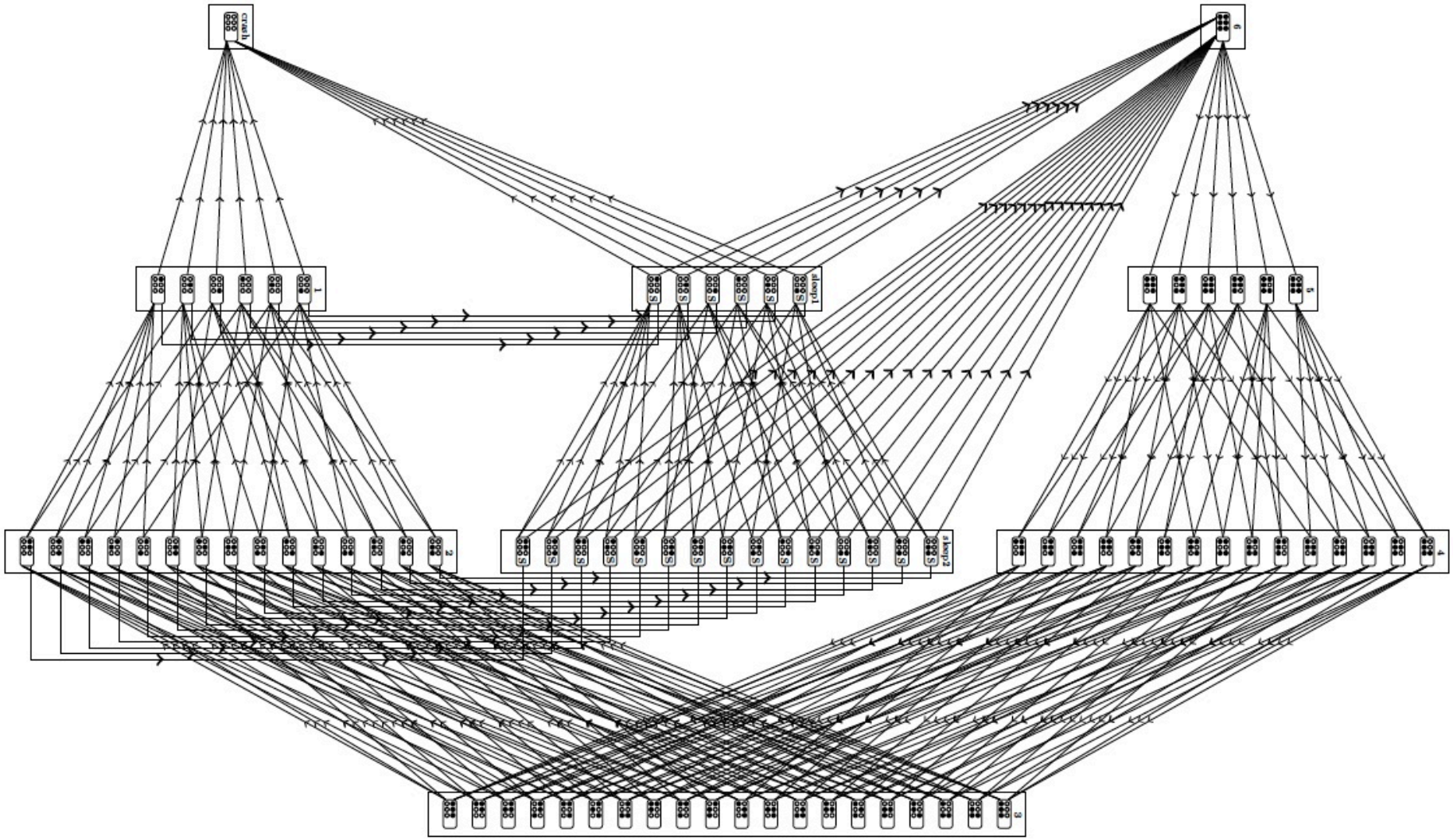
Joost-Pieter Katoen and Tim Quatmann

Software Modeling and Verification Group

RWTH Aachen, SoSe 2022

Overview

State Spaces Can Be Gigantic

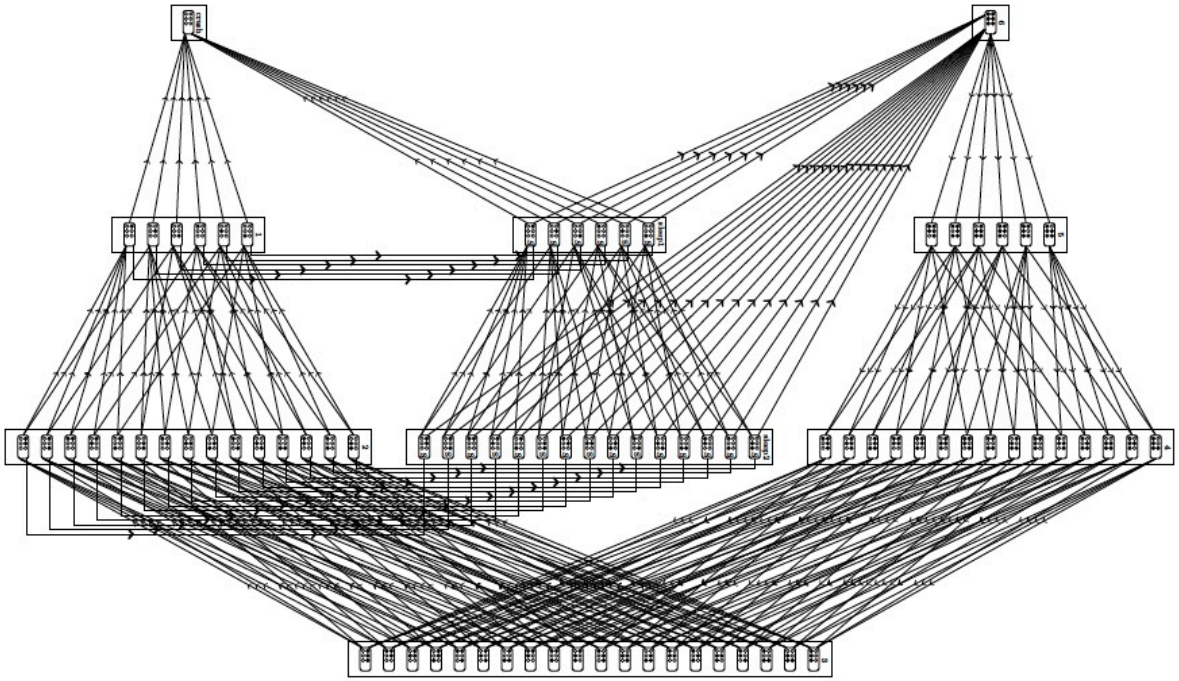


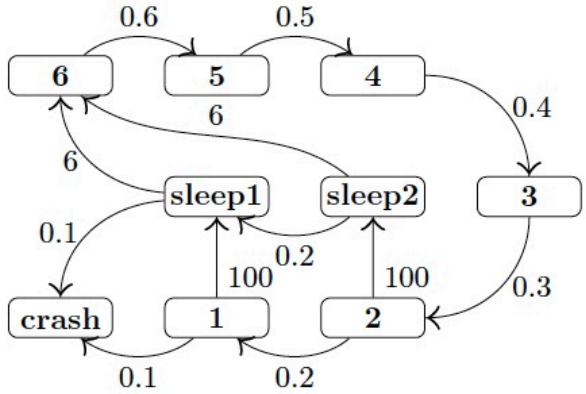
A model of the Hubble telescope

Treating Gigantic Models?

- Use compact data structures
- Make models smaller prior to (or: during) model checking
  - bisimulation quotienting
- Try to make them even smaller
  - partial order reduction
- If possible, try to obtain the smallest possible model
- While preserving the properties of interest
- Do this all algorithmically and possibly fast

Abstraction





	Gigantic	versus	smallest
Is a crash state reachable?	✔		✔
Is a failure repaired on time?	✖		✖

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

$\widehat{TS} \models \varphi \rightarrow TS \models \varphi$

$\widehat{TS} \not\models \varphi \rightarrow TS \not\models \varphi$

e.g. $TS \sim \widehat{TS}$ iff $TS \equiv_{CTL^*} \widehat{TS}$

```
~> storm --qvbs bluetooth 1 --engine dd-to-sparse --bisimulation
Storm 1.6.5 (dev)

Date: Mon Jun 20 09:35:47 2022
Command line arguments: --qvbs bluetooth 1 --engine dd-to-sparse --bisimulation
Current working directory: /Users/tim

-------------------------------------------------------------
QVBS dtmc-Benchmark: Bluetooth Device Discovery (bluetooth) v1
2 instances:
         0      bluetooth.jani  mrec=1  (3411945339 states)
        *1      bluetooth.jani  mrec=2  (55300038488 states)
1 property:
        *time   (exp-reward)
-------------------------------------------------------------
Time for model input parsing: 0.007s.

Time for model construction: 8.168s.

-------------------------------------------------------------
Model type:     DTMC (symbolic)
States:         55286451194 (811817 nodes)
Transitions:    58622673757 (1773402 nodes)
Reward Models:  time
Variables:      rows: 17 meta variables (57 DD variables), columns: 17 meta variables (57 DD variables)
Labels:         2
   * deadlock -> 0 state(s) (1 nodes)
   * init -> 536870912 state(s) (32 nodes)
-------------------------------------------------------------

Time for model preprocessing: 127.323s.

-------------------------------------------------------------
Model type:     DTMC (sparse)
States:         775062
Transitions:    1394568
Reward Models:  time
State Labels:   3 labels
   * (rec = 2) -> 1 item(s)
   * deadlock -> 0 item(s)
   * init -> 268717 item(s)
Choice Labels:  none
-------------------------------------------------------------

Model checking property "time": R[exp]{"time"}min=? [F (rec = 2)] ...
Result (for initial states): 16565
Time for model checking: 0.209s.
```

Overview

Bisimulation

Definition: bisimulation relation

Let $TS_i = (S_i, Act_i, \rightarrow_i, I_i, AP, L_i)$, i=1, 2, be transition systems. The symmetric relation $\mathfrak{R} \subseteq \big((S_1 \cup S_2) \times (S_1 \cup S_2)\big)$ is a bisimulation for (TS_1, TS_2) whenever:

1. for i=1, 2 and all $s_1 \in I_i$ there exists $s_2 \in I_{3-i}$ such that $(s_1, s_2) \in \mathfrak{R}$
2. for all $(s_1, s_2) \in \mathfrak{R}$ it holds:
   - 2.1 $L_1(s_1) = L_2(s_2)$, and
   - 2.2 $s_1' \in Post(s_1)$ implies $(s_1', s_2') \in \mathfrak{R}$ for some $s_2' \in Post(s_2)$.

Visually

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

Visually

$$\begin{array}{lll} s_1 & \rightarrow & s_1' \\ \mathfrak{R} & & \\ s_2 & & \end{array} \quad \text{can be completed to} \quad \begin{array}{lll} s_1 & \rightarrow & s_1' \\ \mathfrak{R} & & \mathfrak{R} \\ s_2 & \rightarrow & s_2' \end{array}$$

and by symmetry

$$\begin{array}{lll} s_1 & & \\ \mathfrak{R} & & \\ s_2 & \rightarrow & s_2' \end{array} \quad \text{can be completed to} \quad \begin{array}{lll} s_1 & \rightarrow & s_1' \\ \mathfrak{R} & & \mathfrak{R} \\ s_2 & \rightarrow & s_2' \end{array}$$

Bisimulation Equivalence

Definition: bisimulation equivalence

TS_1 and TS_2 are bisimulation equivalent (short: bisimilar), denoted $TS_1 \sim TS_2$, if there exists a bisimulation for (TS_1, TS_2). That is:

$$\sim \;=\; \bigcup \{\, \mathfrak{R} \mid \mathfrak{R} \text{ is a bisimulation on } (TS_1, TS_2) \,\}.$$

Bisimilarity ($\sim$) is an equivalence relation.

Bisimulation on States

Definition: bisimulation/bisimilarity on states

Symmetric relation $\mathfrak{R} \subseteq S \times S$ is a bisimulation on TS (with state space S) if for any $(s_1, s_2) \in \mathfrak{R}$:

1. $L(s_1) = L(s_2)$
2. $s_1' \in Post(s_1)$ then $(s_1', s_2') \in \mathfrak{R}$ for some $s_2' \in Post(s_2)$.

The states s_1 and s_2 are bisimilar, denoted $s_1 \sim_{TS} s_2$, if $(s_1, s_2) \in \mathfrak{R}$ for some bisimulation $\mathfrak{R}$ for TS.

$s_1 \sim_{TS} s_2$ if and only if $TS_{s_1} \sim TS_{s_2}$ where TS_{s_i} denotes the transition system TS in which s_i is the only initial state.

Overview

Coarsest Bisimulation

The relation $\sim_{TS}$ is a bisimulation, an equivalence, and the coarsest bisimulation for TS.

- There is no bisimulation $\mathfrak{R}$ with $(s_1, s_2) \in \mathfrak{R}$ and $s_1 \not\sim_{TS} s_2$

Quotient Transition System

Definition: quotient transition system

For $TS = (S, Act, \rightarrow, I, AP, L)$ and bisimulation $\sim_{TS} \subseteq S \times S$ on TS, let the quotient transition system

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

where

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

- $TS/\sim_{TS}$ is the quotient of TS under $\sim_{TS}$

Property

For every transition system TS it holds: $TS \sim TS/{\sim_{TS}}$.

Proof sketch.

Show that $\mathfrak{R} = \{(s, [s]_\sim) \mid s \in S\}$ is a bisimulation for $(TS, TS/{\sim_{TS}})$. □

- Hence, $TS \equiv_{\mathrm{CTL}^*} TS/{\sim_{TS}}$

Example

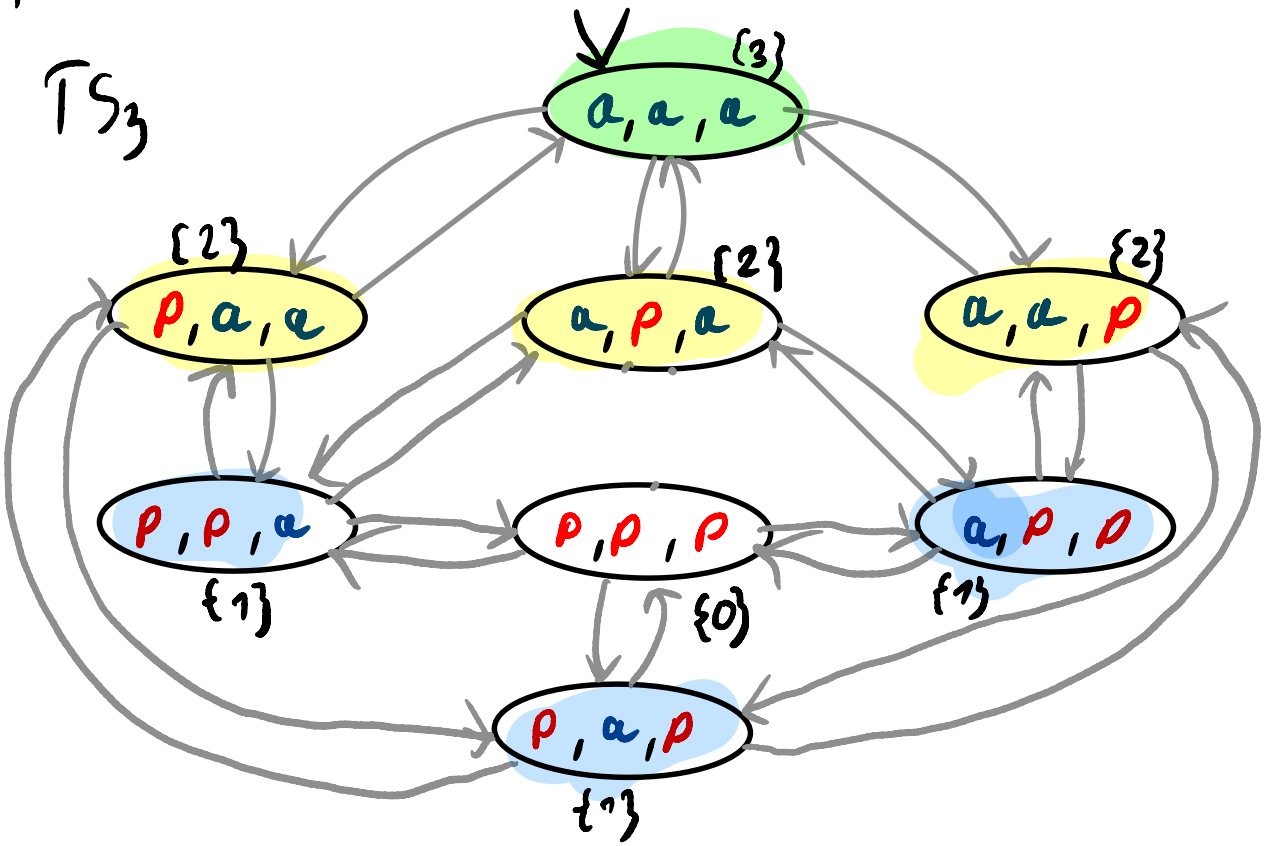
Let P: 

// simplified model of a printer that is either available or printing

$TS_n = \underbrace{P \parallel \cdots \parallel P}_{n \text{ printers}}$ over $AP = \{0, 1, 2, \ldots\}$ // number of available printers

For $n=3$:

TS_3



$TS_{3/\sim}$

(Simplified) Lamport's Bakery Algorithm

Thread 1:

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

Thread 2:

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

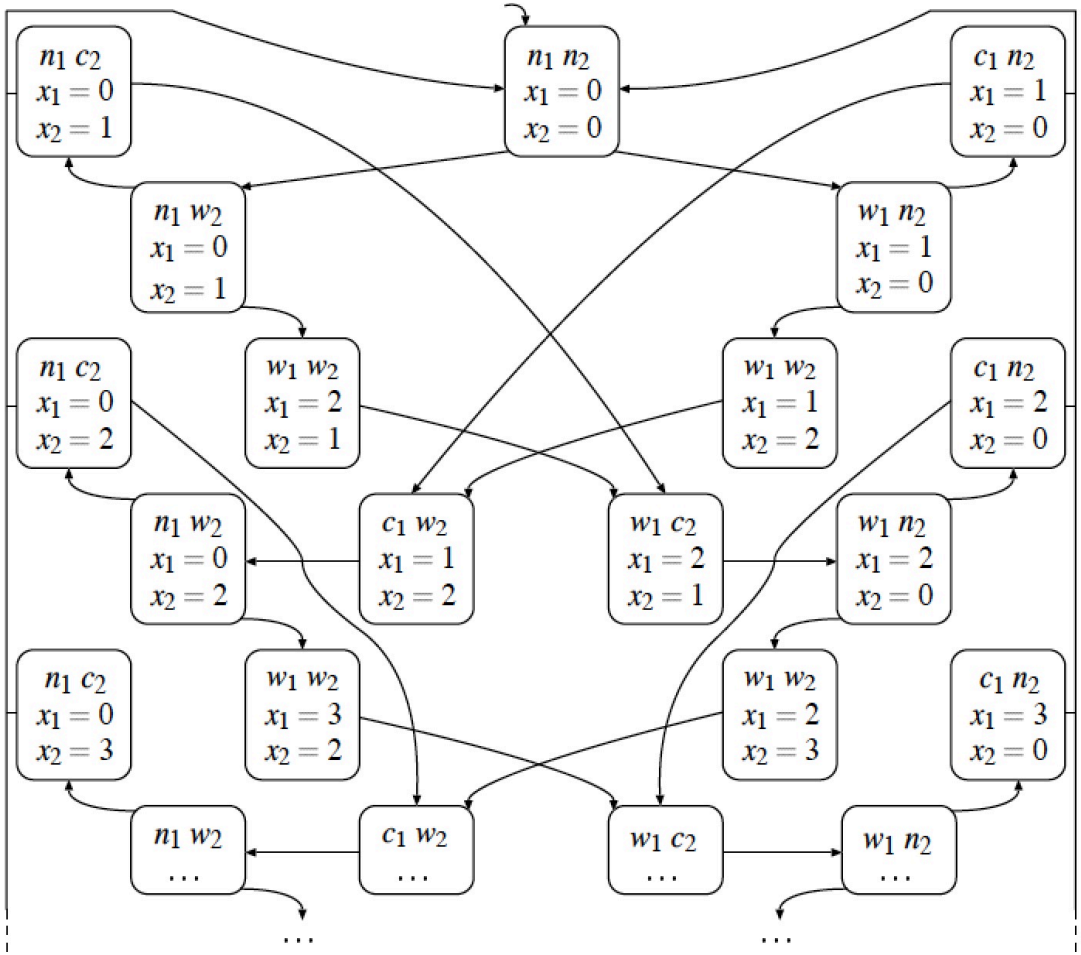
- If initially $x_1 = x_2 = 0$, can the counters grow unboundedly large?

Example Bakery Algorithm Run

thread P_1	thread 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
...	...	..	..	...

Counters may grow unboundedly large.

Bakery Algorithm Transition System



Infinite state space due to possible unbounded increase of counters

Bisimulation Relation

"abstraction function"

Let function f map a reachable state of TS_{Bak} onto a state in TS^{abs}_{Bak}

Let $s = \langle \ell_1, \ell_2, x_1 = b_1, x_2 = b_2 \rangle \in TS_{Bak}$ with $\ell_i \in \{ n_i, w_i, c_i \}$ and $b_i \in \mathbb{N}$

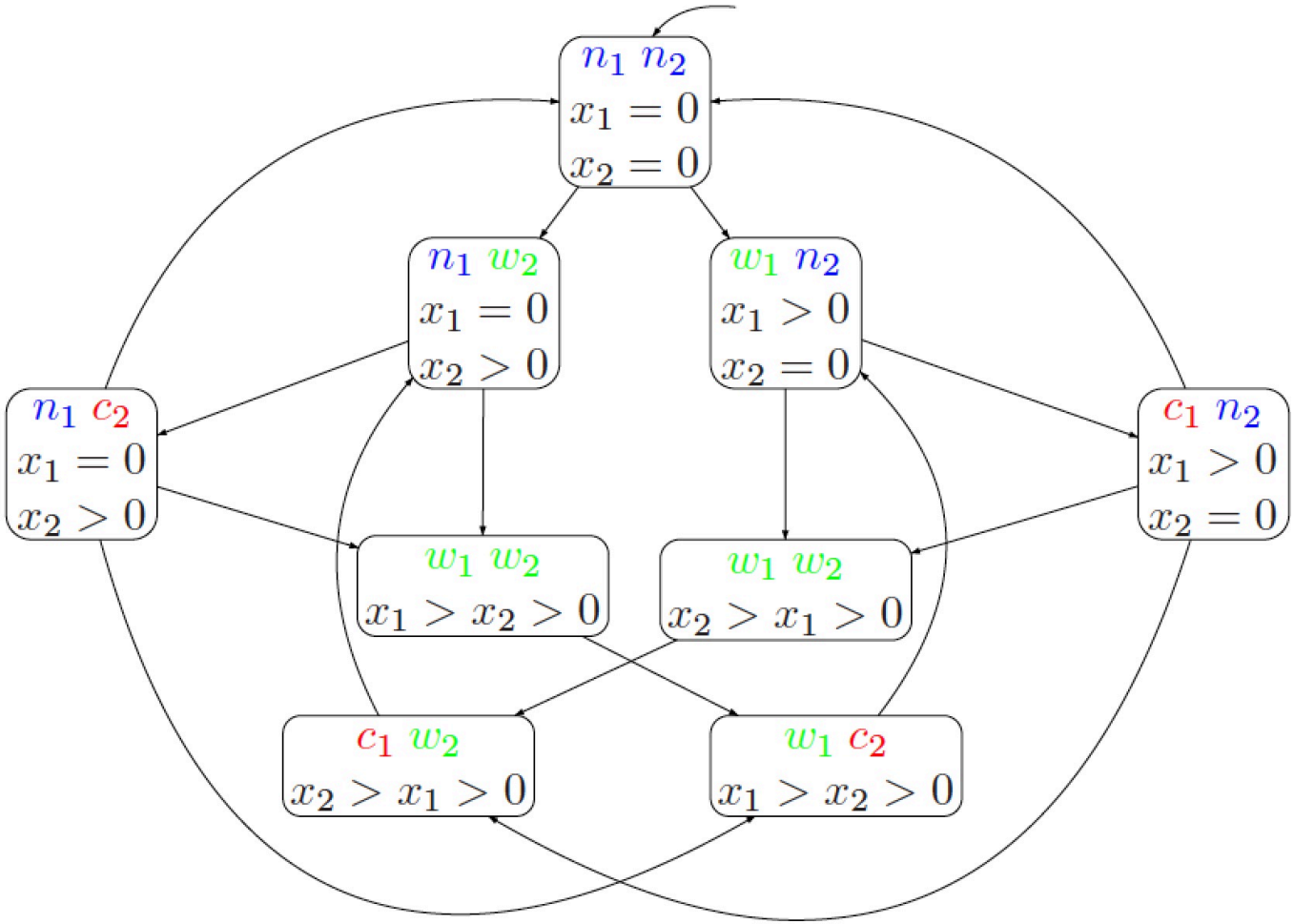
$\in \mathbb{N}$ $\in \mathbb{N}$ AP

Then:

$$f(s) = \begin{cases} \langle \ell_1, \ell_2, x_1 = 0, x_2 = 0 \rangle & \text{if } b_1 = b_2 = 0 \\ \langle \ell_1, \ell_2, x_1 = 0, x_2 > 0 \rangle & \text{if } b_1 = 0 \text{ and } b_2 > 0 \\ \langle \ell_1, \ell_2, x_1 > 0, x_2 = 0 \rangle & \text{if } b_1 > 0 \text{ and } b_2 = 0 \\ \langle \ell_1, \ell_2, x_1 > x_2 > 0 \rangle & \text{if } b_1 > b_2 > 0 \\ \langle \ell_1, \ell_2, x_2 > x_1 > 0 \rangle & \text{if } b_2 > b_1 > 0 \end{cases}$$

It follows: $\mathfrak{R} = \{ (s, f(s)) \mid s \in S \}$ is a bisimulation for $(TS_{Bak}, TS^{abs}_{Bak})$ for any subset of $AP = \{ noncrit_i, wait_i, crit_i \mid i = 1, 2 \}$.

Quotient of Bakery Algorithm



$TS^{abs}_{Bak} = TS_{Bak} / \sim$ for $AP = \{ noncrit_i, wait_i, crit_i \mid i = 1, 2 \}$

Bisimulation Quotienting

Overview

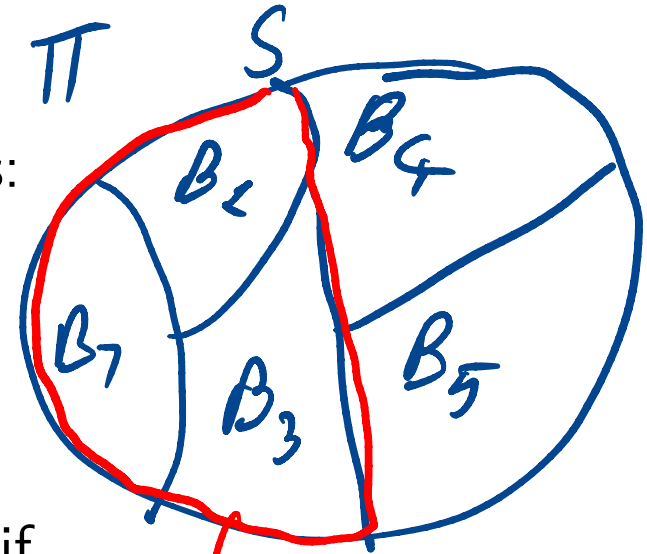
for finite TS

└ computing $\nsim_{TS}$

Bisimulation Quotienting

Partitions

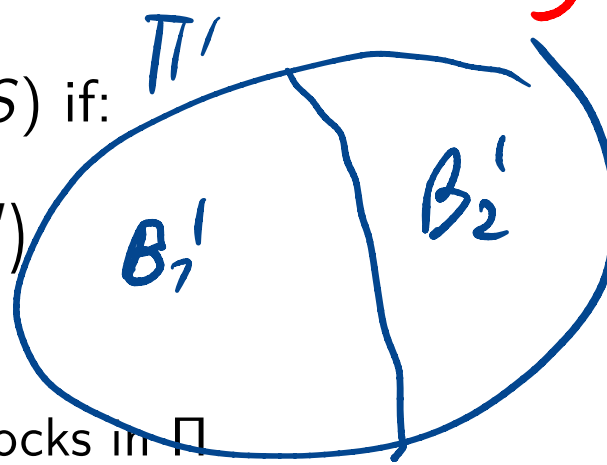
- A partition $\Pi = \{ B_1, \ldots, B_k \}$ of S satisfies:
  - B_i is non-empty; B_i is called a block
  - $B_i \cap B_j = \varnothing$ for all i, j with $i \neq j$
  - $B_1 \cup \ldots \cup B_k = S$



- $C \subseteq S$ is a super-block of partition Π of S if
  $C = B_{i_1} \cup \ldots \cup B_{i_m}$ for $B_{i_j} \in \Pi$ for $0 < j \leq m$

- Partition Π (of S) is finer than partition Π' (of S) if:

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



  - each block of Π' equals the union of a set of blocks in Π

- Π is strictly finer than Π' if it is finer than Π' and $\Pi \neq \Pi'$

Bisimulation Quotienting

Partitions and Equivalences

, set of eq. classes

- $\mathfrak{R}$ is an equivalence on $S \quad \Rightarrow \quad S/\mathfrak{R}$ is a partition of S

- Partition $\Pi = \{ B_1, \ldots, B_k \}$ of S induces the equivalence relation

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

  where it holds: $S/\mathfrak{R}_\Pi \;=\; \Pi$.

There is a one-to-one relationship between partitions and equivalences.

Bisimulation Quotienting

Partition Refinement

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 is ensured:

$$S \times S \supseteq \mathfrak{R}_{\Pi_0} \supsetneq \mathfrak{R}_{\Pi_1} \supsetneq \mathfrak{R}_{\Pi_2} \supsetneq \ldots \supsetneq \mathfrak{R}_{\Pi_i} = {\sim_{TS}}$$

  - time complexity: maximally $|S|$ iterations needed

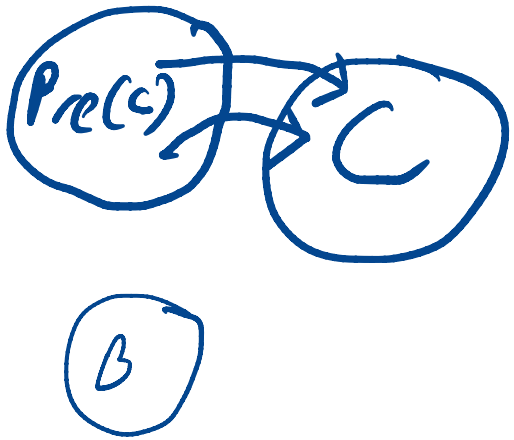
Bisimulation Quotienting

Theorem

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

1. Π is finer than the initial partition Π_{AP}, and
2. for all $B, C \in \Pi$ it holds[1]:

$$B \cap Pre(C) = \varnothing \text{ or } B \subseteq Pre(C).$$



[1] In fact, this also holds for all $B \in \Pi$ and all super-blocks C of Π.

Proof: we show that

(a) R_Π is a bisimulation iff Π satisfies 1.+2.

(b) $S/_\sim$ is the coarsest bisimulation satisfying 1.+2.

ad (a): "$\Leftarrow$" assume Π satisfies 1.+2. Let $(s_1, s_2) \in R_\Pi$ and $B = [s_1]_\Pi = [s_2]_\Pi$, the unique block containing s_1 and s_2. Due to 1., Π is finer than Π_{AP}, so $B \subseteq B' \in \Pi_{AP}$ for some B'. Thus $L(s_1) = L(s_2)$. Let $s_1 \to s_1'$ and $C = [s_1']_\Pi$. Thus, $s_1 \in B \cap Pre(C)$. By 2., $B \subseteq Pre(C)$. So, $s_2 \in Pre(C)$. Thus $s_2 \to s_2'$ with $s_2' \in C = [s_1']_\Pi$. Thus $(s_1', s_2') \in R_\Pi$.

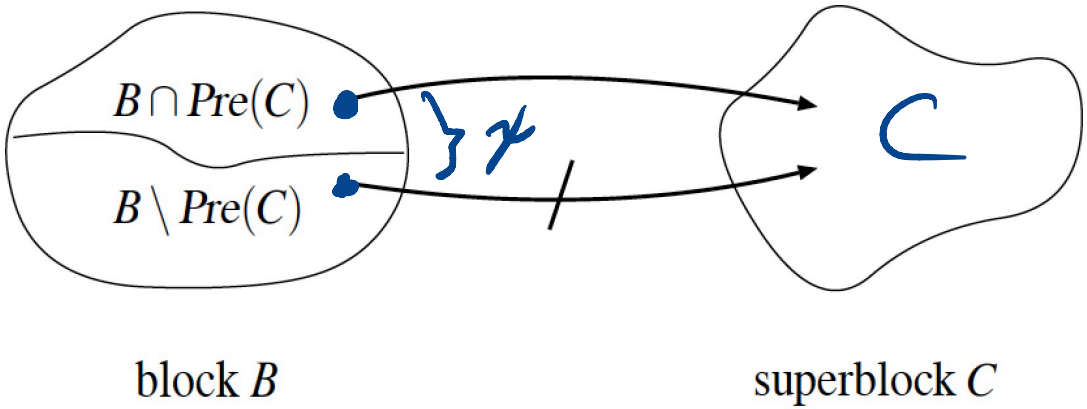
"$\Rightarrow$" assume R_Π is a bisimulation. Evidently, Π satisfies 1. Let B, C be blocks of R_Π. Assume $B \cap Pre(C) \neq \emptyset$. We show: $B \subseteq Pre(C)$. If $B \cap Pre(C) \neq \emptyset$, $s_1 \to C$ for some $s_1 \in B$. Let $s_1 \to s_1' \in C$ and $s_2 \in B$. Since $s_1, s_2 \in B$, $(s_1, s_2) \in R_\Pi$. As $s_1 \to s_1'$, $s_2 \to s_2'$ for some s_2' with $(s_1', s_2') \in R_\Pi$. Thus $s_2' \in Pre(C)$.

(b) this follows immediately from the fact that $\sim_{TS}$ is the coarsest bisimulation on TS.

Bisimulation Quotienting

Refinement Operator

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

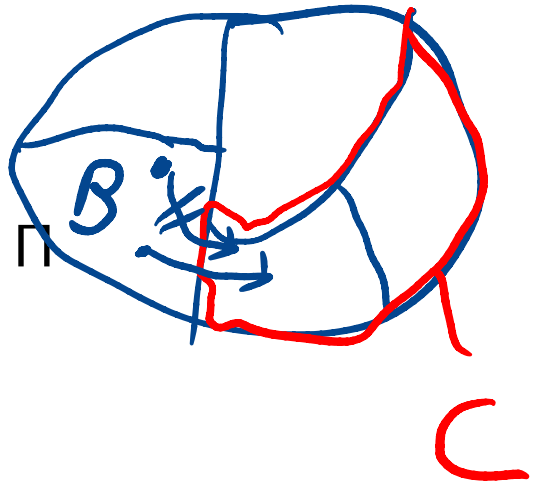


- Basic properties:
  - for Π finer than Π_{AP} and coarser than $S/\sim$:

    $\mathit{Refine}(\Pi, C)$ 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

Bisimulation Quotienting

Splitters



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

$$B \cap Pre(C) \neq \varnothing \quad \text{and} \quad B \setminus Pre(C) \neq \varnothing$$

- Block B is stable wrt. C if

$$B \cap Pre(C) = \varnothing \quad \text{or} \quad B \setminus Pre(C) = \varnothing$$

- Π is stable w.r.t. C if every $B \in \Pi$ is stable wrt. C

Bisimulation Quotienting

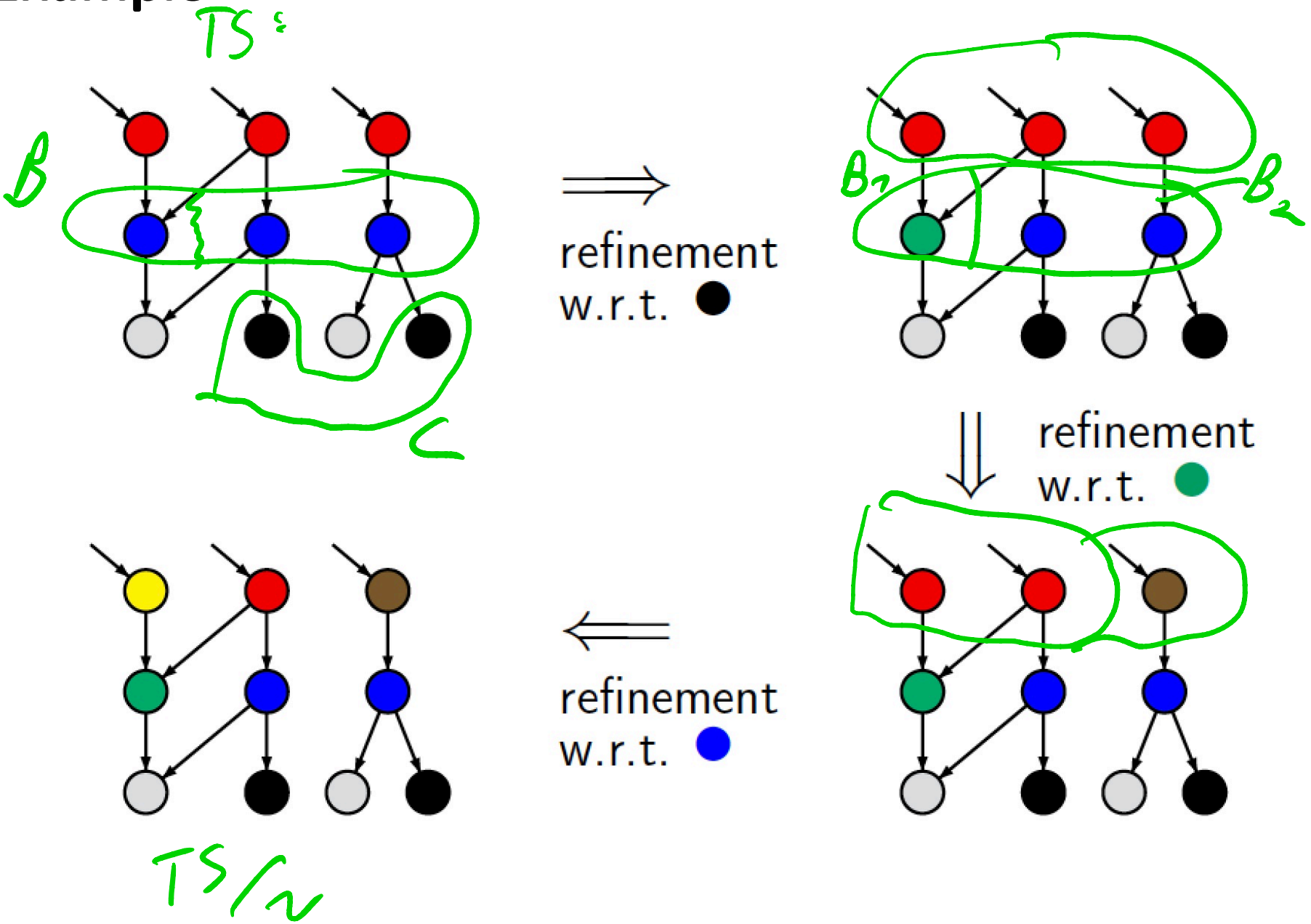
Algorithm Skeleton

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

Bisimulation Quotienting

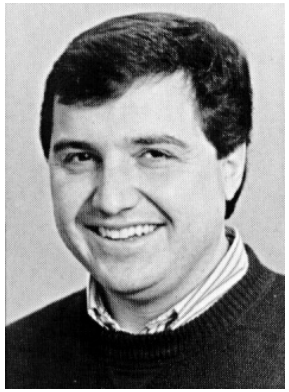
Example



Bisimulation Quotienting

Splitter Selection

Scott Smolka (1954 –)



Paris Kanellakis (1953 – †1995)

Robert A. Paige (†1999)

Robert E. Tarjan (1948 –)

Bisimulation Quotienting

Which Splitter to Take?

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

1. Simple strategy: $O(|S| \cdot |\rightarrow|)$

   use any block of Π_i as splitter candidate

2. Advanced strategy: $O(\log(|S|) \cdot |\rightarrow|)$

   use only "smaller" blocks of Π_i as splitter candidates

   and apply "a ternary" refinement

Bisimulation Quotienting

Simple Selection Strategy

by Smolka & Kanellakis

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

$\Pi \leftarrow \Pi_{AP}$;
repeat
  $\Pi_{old} \leftarrow \Pi$;
  foreach $C \in \Pi_{old}$ **do**
    $\Pi \leftarrow \mathit{Refine}(\Pi, C)$
until $\Pi = \Pi_{old}$;
return Π

Bisimulation Quotienting

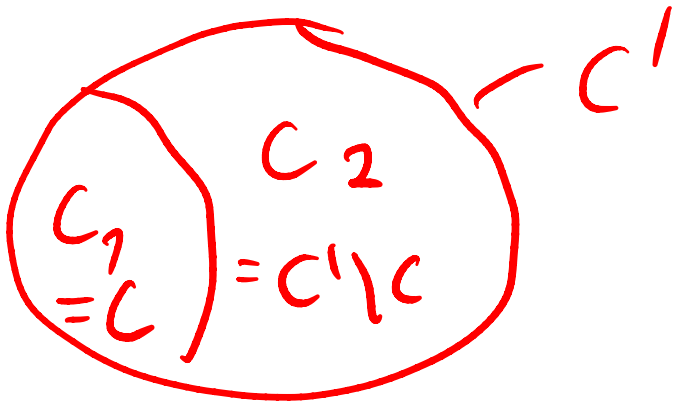
Advanced Selection Strategy

by Paige & Tarjan

- 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 $C_1 = C' \cap Pre(D)$ and $C_2 = C' \setminus Pre(D)$
- Step i+1: use the *smallest* $C \in \{ C_1, C_2 \}$ as splitter
  - let C be such that $|C| \leq |C'|/2$, thus $|C| \leq |C' \setminus C|$



Bisimulation Quotienting

Advanced Selection Strategy

by Paige & Tarjan

- 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 $C_1 = C' \cap Pre(D)$ and $C_2 = C' \setminus Pre(D)$
- Step i+1: use the *smallest* $C \in \{ C_1, C_2 \}$ as splitter
  - let C be such that $|C| \leq |C'|/2$, thus $|C| \leq |C' \setminus C|$
- Combine the refinement steps with respect to C and $C' \setminus C$
  - using the ternary refinement operator

$$Refine(\Pi, C, C' \setminus C) := Refine(\, Refine(\Pi, C),\ C' \setminus C \,)$$

where $|C| \leq |C' \setminus C|$

  - the decomposed blocks are stable with respect to C and $C' \setminus C$

Bisimulation Quotienting

The Ternary Refinement Operator

We have $\mathit{Refine}(\Pi, C, C' \setminus C) = \bigcup_{B \in \Pi} \mathit{Refine}(B, C, C' \setminus C)$

where $\mathit{Refine}(B, C, C' \setminus C) = \{ B_1, B_2, B_3, B_4 \} \setminus \{ \varnothing \}$ with:

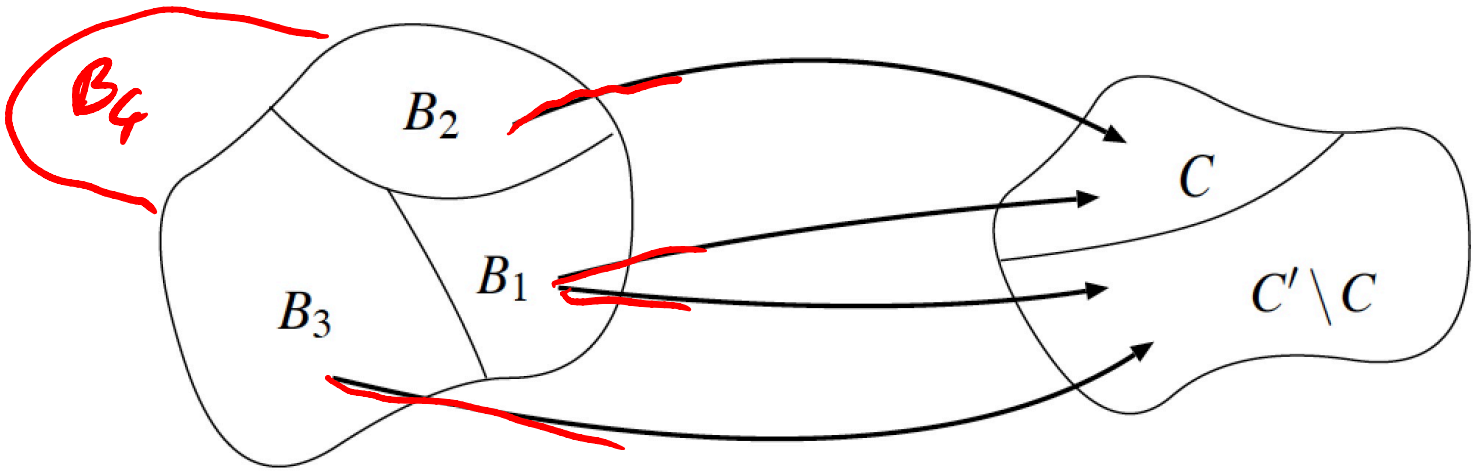
$B_1 = B \cap \mathit{Pre}(C) \cap \mathit{Pre}(C' \setminus C)$ to both C and $C' \setminus C$

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

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

$B_4 = B \setminus \mathit{Pre}(C')$ to neither C nor $C' \setminus C$

$\Rightarrow$ blocks B_1, B_2, B_3, B_4 are stable with respect to C and $C' \setminus C$



Bisimulation Quotienting

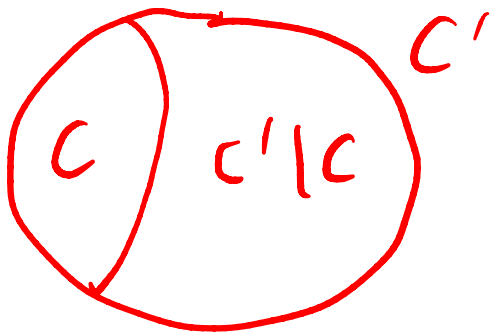
Quotienting Algorithm

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

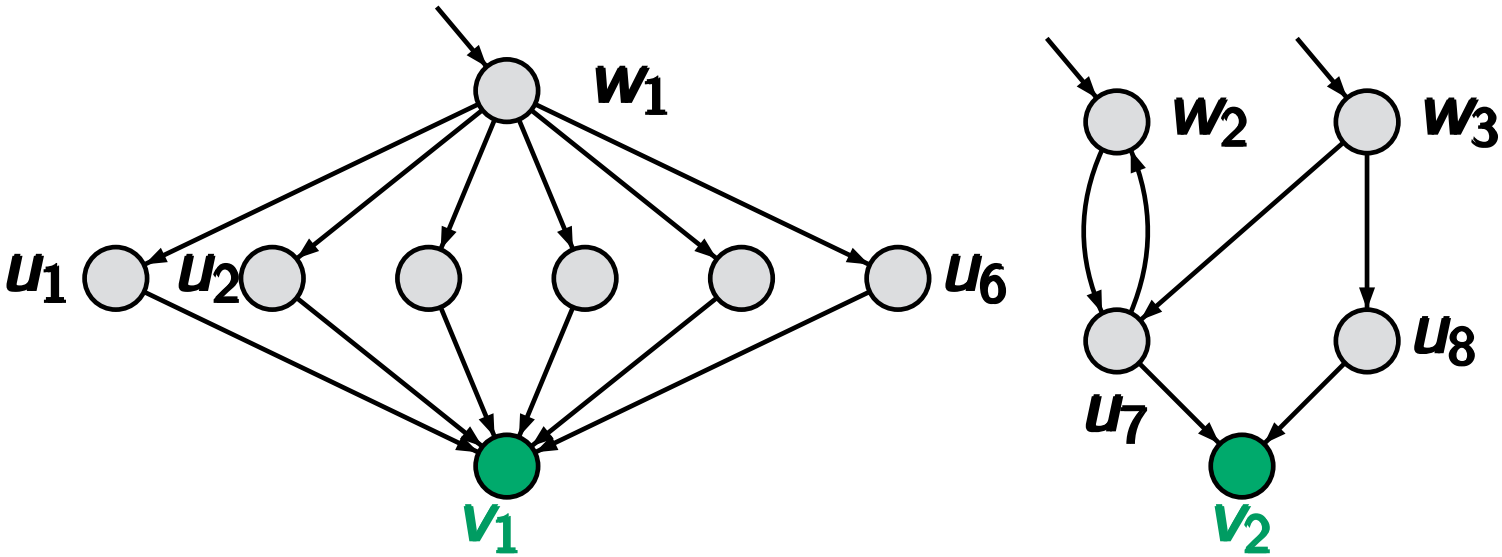
$\Pi_{old} := \{ S \}$;
$\Pi := \mathit{Refine}(\Pi_{AP}, S)$;

(* 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} *)

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

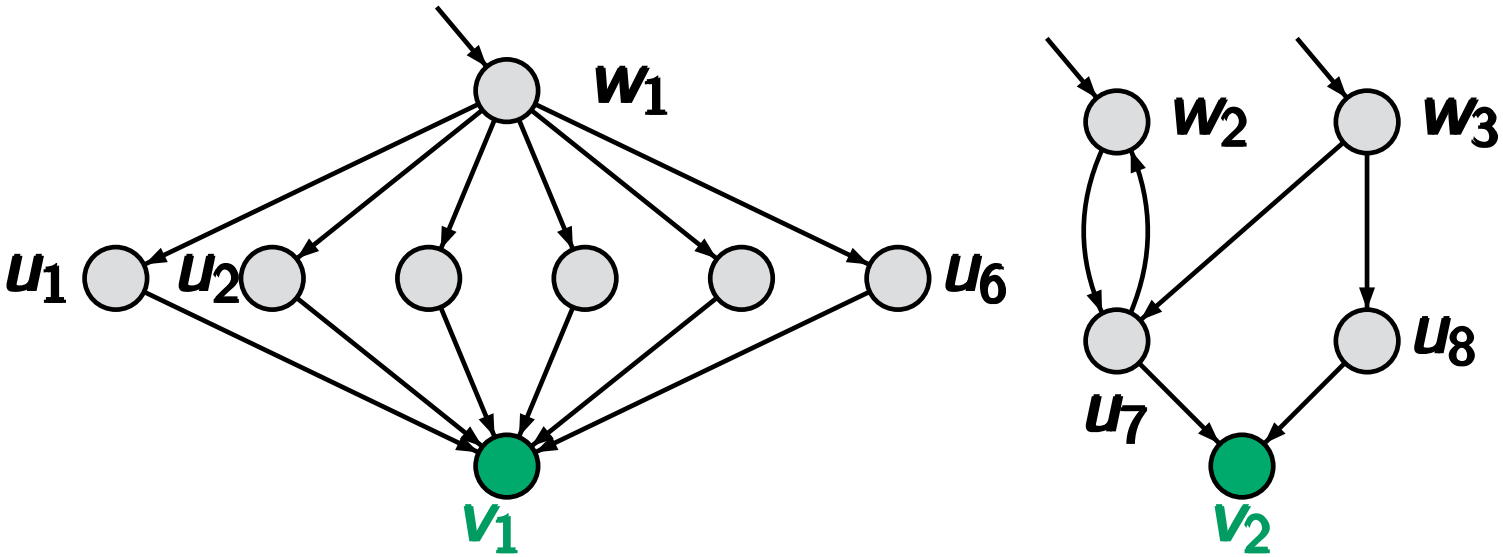


Example: Paige-Tarjan algorithm



$AP = \{green, gray\}, \quad \mathcal{B}_{\text{old}} = \{S\}$

Example: Paige-Tarjan algorithm

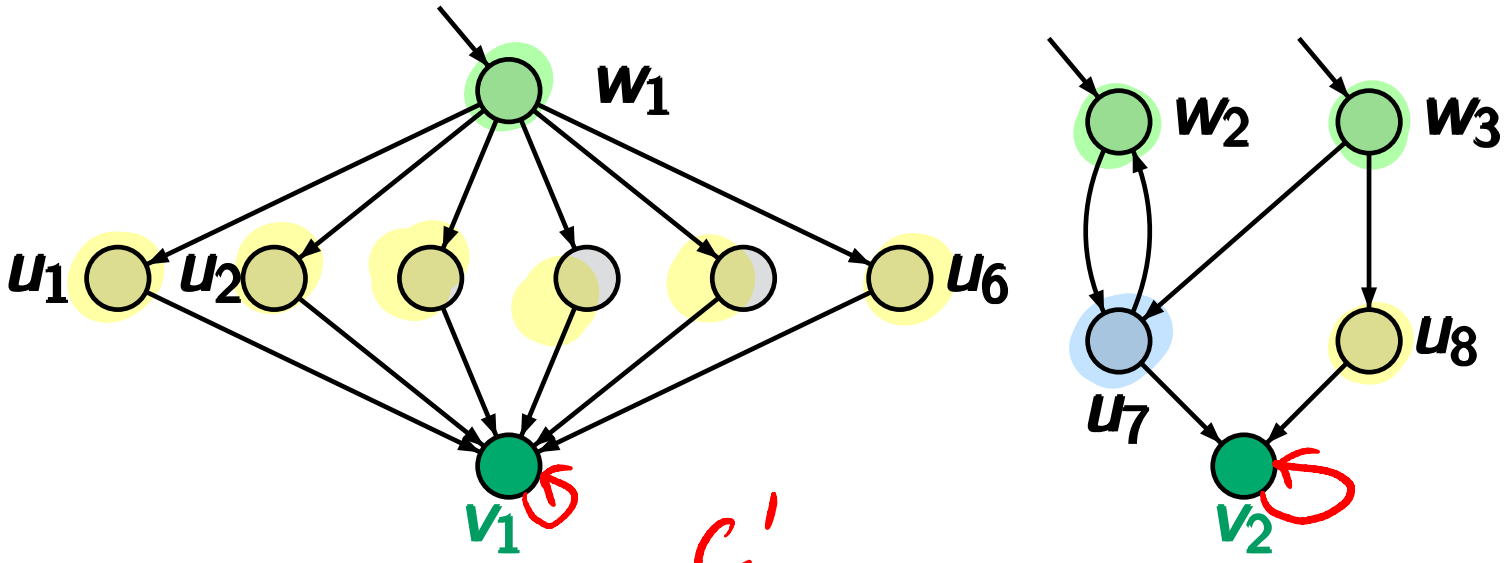


$$AP = \{green, gray\}, \quad \mathcal{B}_{\text{old}} = \{S\}$$

initial partition:

$$\mathcal{B}_0 = Refine(\mathcal{B}_{AP}, S) = \mathcal{B}_{AP}$$
$$= \{\{v_1, v_2\}, \{u_1, \ldots, u_8, w_1, w_2, w_3\}\}$$

Example: Paige-Tarjan algorithm



initially: $\mathcal{B}_{\text{old}} = \{S\}$

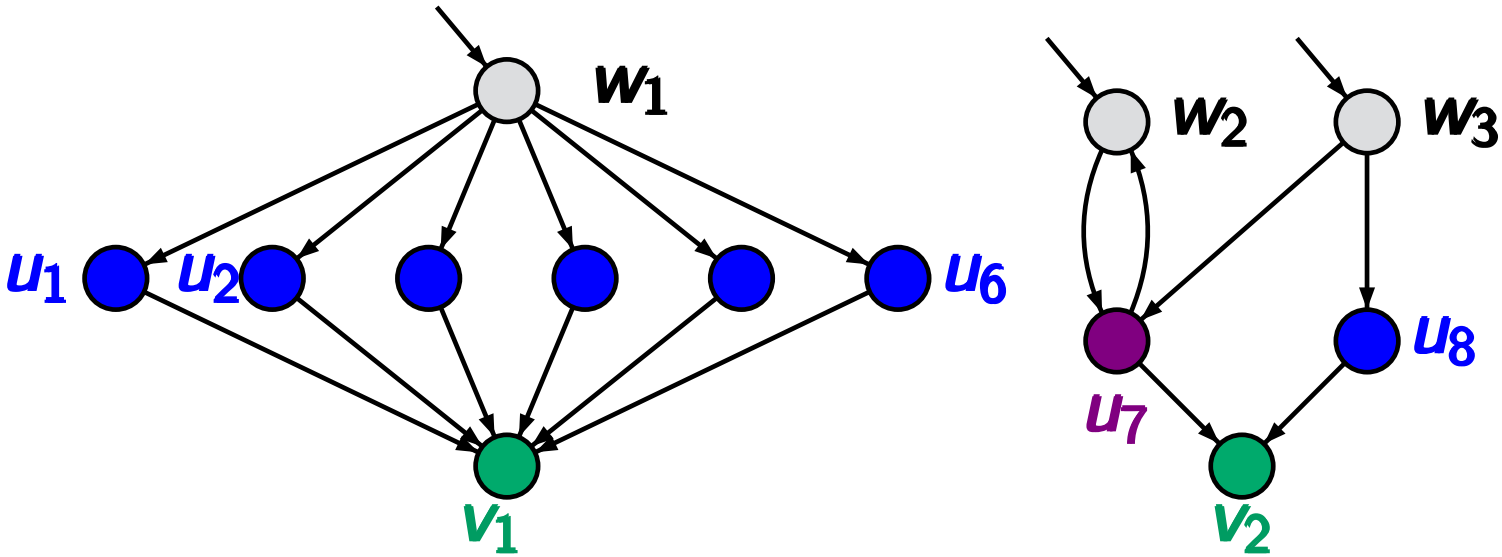
$\mathcal{B}_0 = \{\{v_1, v_2\}, \{u_1, \ldots, u_8, w_1, w_2, w_3\}\}$

C $\quad$ $C' \setminus C$

first refinement step:

$\mathit{Refine}(\mathcal{B}_0, \{v_1, v_2\}, S \setminus \{v_1, v_2\})$

Example: Paige-Tarjan algorithm



initially: $\mathcal{B}_{\text{old}} = \{S\}$

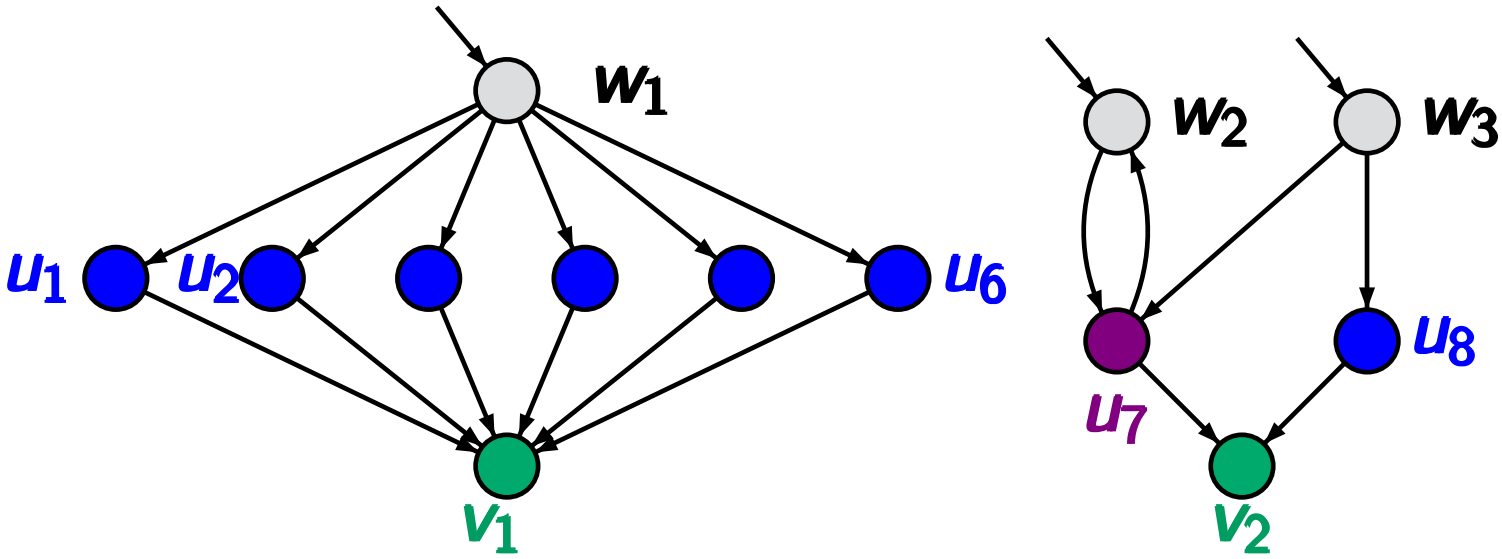
$$\mathcal{B}_0 = \big\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\big\}$$

first refinement step:

$$\mathit{Refine}(\mathcal{B}_0, \{v_1, v_2\}, S \setminus \{v_1, v_2\}) =$$

$$\mathcal{B}_1 = \big\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1, w_2, w_3\}\big\}$$

Example: Paige-Tarjan algorithm



initially: $\mathcal{B}_{\text{old}} = \{S\}$

$\mathcal{B}_0 = \big\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\big\}$

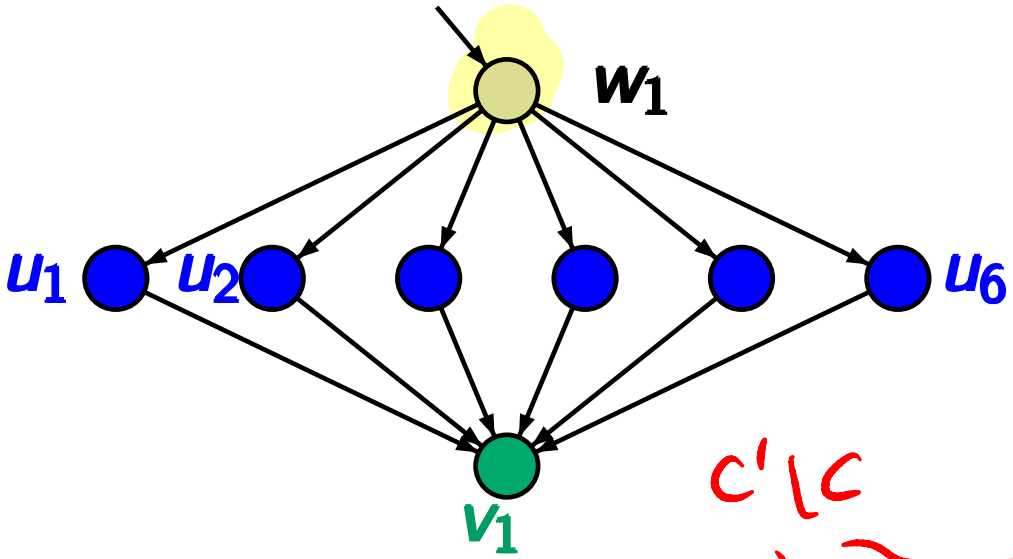
first refinement step:

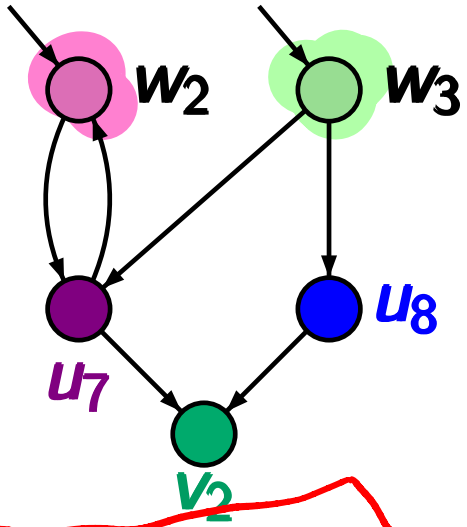
$\mathit{Refine}(\mathcal{B}_0, \{v_1, v_2\}, S \setminus \{v_1, v_2\}) =$

$\mathcal{B}_1 = \big\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1, w_2, w_3\}\big\}$

$\mathcal{B}_{\text{old}} = \big\{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\big\}$

Example: Paige-Tarjan algorithm





first refinement step:

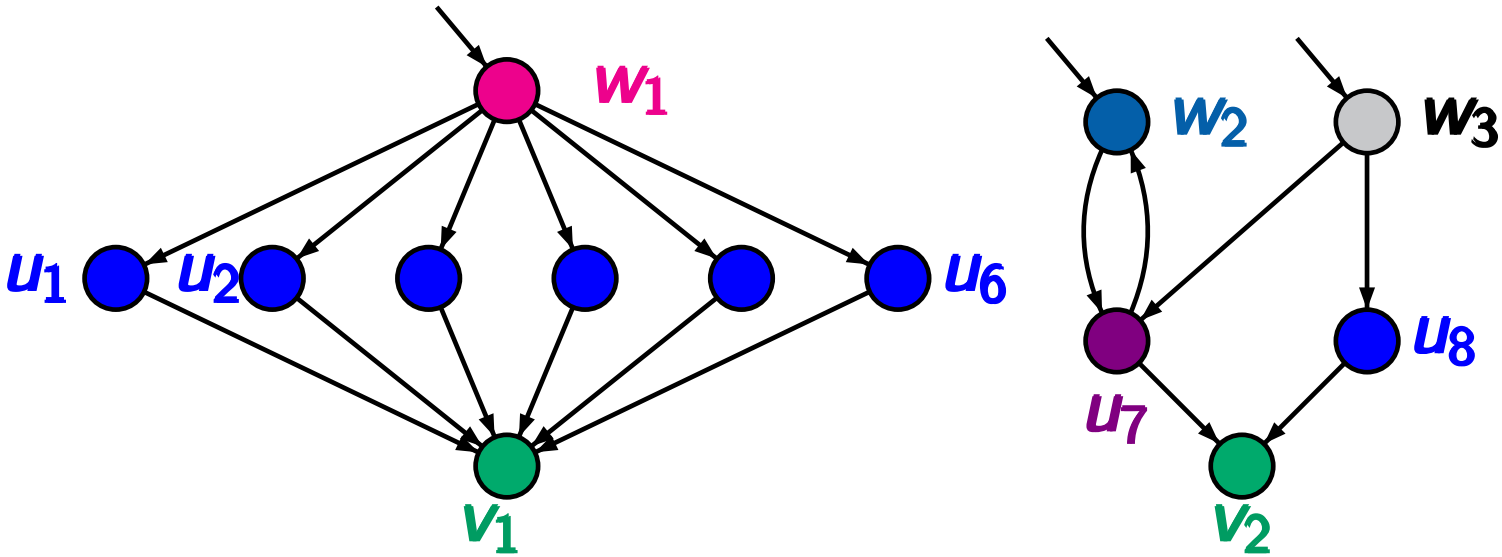
$$\mathcal{B}_1 = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1, w_2, w_3\}\}$$

$$\mathcal{B}_{\text{old}} = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\}$$

second refinement step:

Refine$(\mathcal{B}_1, ?, ?)$

Example: Paige-Tarjan algorithm



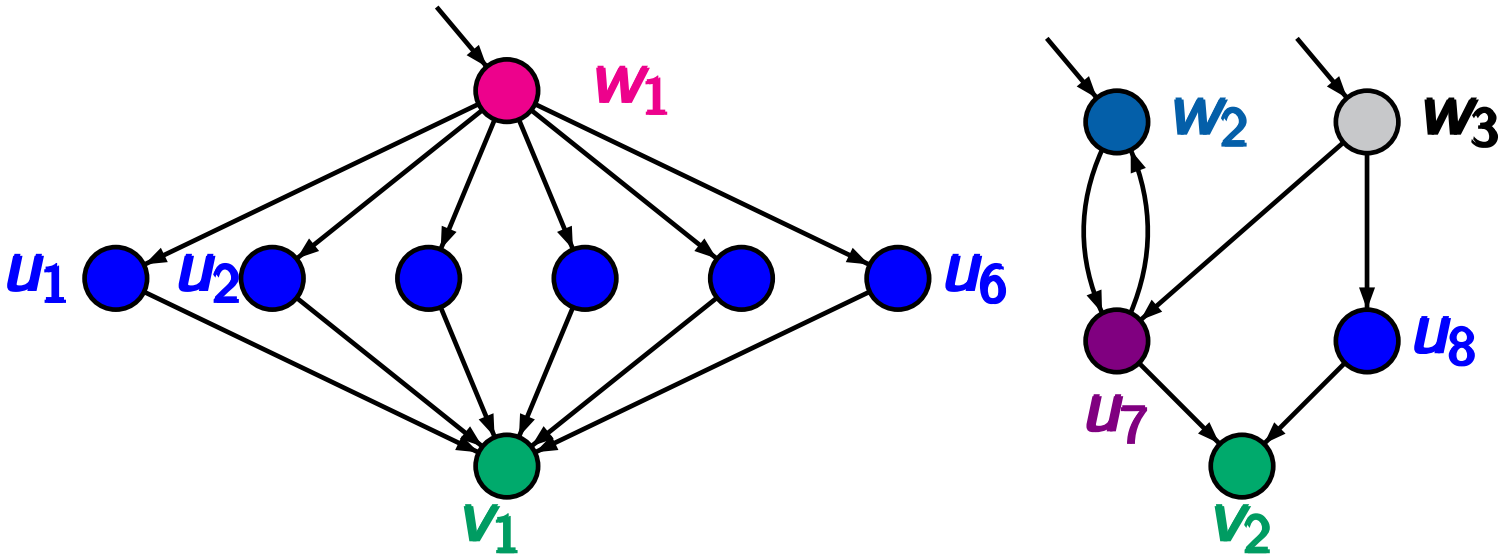
first refinement step:

$$\mathcal{B}_1 = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1, w_2, w_3\}\}$$
$$\mathcal{B}_{\text{old}} = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\}$$

second refinement step:

$$\mathit{Refine}(\mathcal{B}_1, \{u_7\}, \{u_1, \ldots, u_6, u_8, w_1, w_2, w_3\})$$

Example: Paige-Tarjan algorithm



first refinement step:

$$\mathcal{B}_1 = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1, w_2, w_3\}\}$$
$$\mathcal{B}_{\text{old}} = \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8, u_7, w_1, w_2, w_3\}\}$$

second refinement step:

$$\mathit{Refine}(\mathcal{B}_1, \{u_7\}, \{u_1, \ldots, u_6, u_8, w_1, w_2, w_3\})$$
$$= \{\{v_1, v_2\}, \{u_1, \ldots, u_6, u_8\}, \{u_7\}, \{w_1\}, \{w_2\}, \{w_3\}\}$$

$\mathcal{B}_{\text{old}} = \{\{v_1, v_2\}, \{u_7\}, \{u_1, \ldots u_6, u_8, w_1, w_2, w_3\}\}$

Complexity

The bisimulation quotient of finite transition system TS can be computed in $O(M \cdot \log N)$ where N and M are the number of states and transitions in TS respectively.

Checking bisimilarity is PTIME-complete.

Proof.

Reduction from the direct circuit value problem. Outside the scope of this lecture. □

Summary

- The quotient transition system $TS/{\sim_{TS}}$ of TS can be much smaller than TS while preserving CTL* (in particular LTL and CTL) properties
- For finite TS, $\sim_{TS}$ can be computed via partition refinement
- Two approaches for selecting splitters:
  - Simple strategy by Smolka & Kanellakis
    use any block as splitter candidate
  - Advanced strategy by Paige & Tarjan
    use small blocks as splitters together with ternary refinement
- The latter yields an $O(\log |S| \cdot |{\rightarrow}|)$ algorithm

Bisimulation Quotienting

Next Lecture

Thursday June 23, 12:30