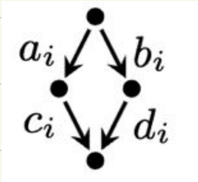
Statefull
Partial-order reduction

Igor Walukiewicz

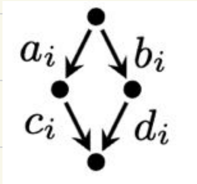
joint work with Frederic Herbreteau, Gerald Point, and Gautham Viswanathan

Partial-order reduction: example

P_1

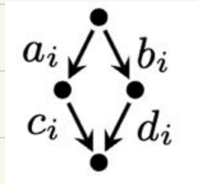


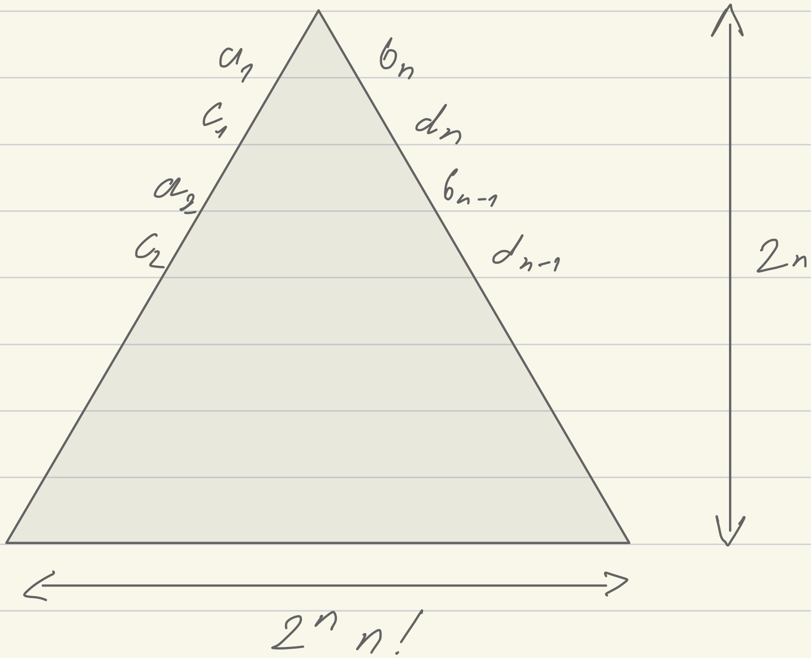
P_2



. . .

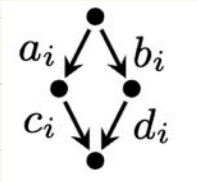
P_n



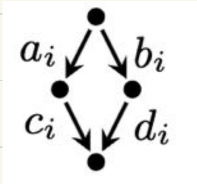


Partial-order reduction: example

P_1

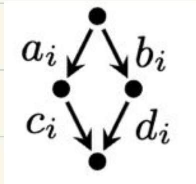


P_2

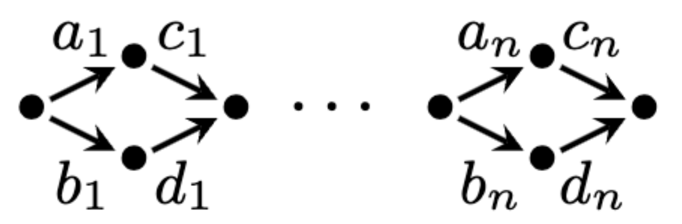


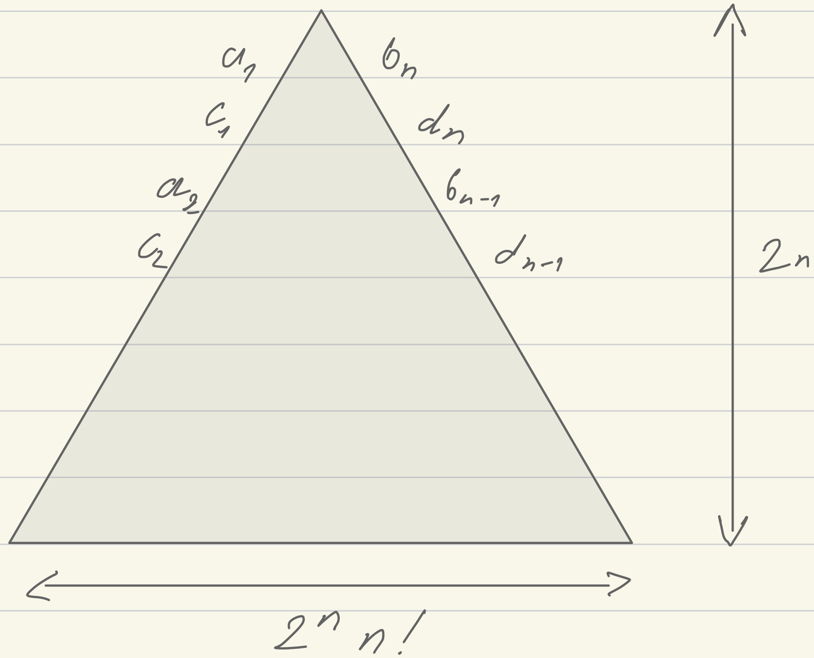
...

P_n



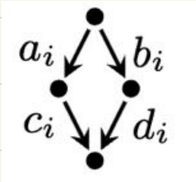
Small transition system



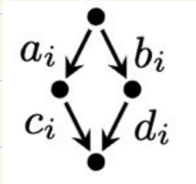


Partial-order reduction: example

P_1

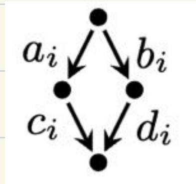


P_2

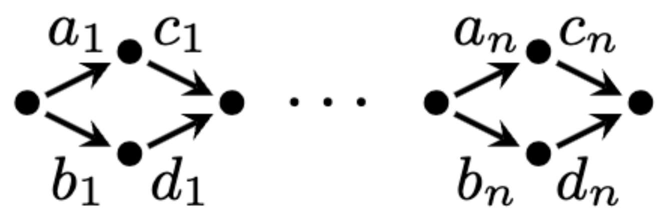


...

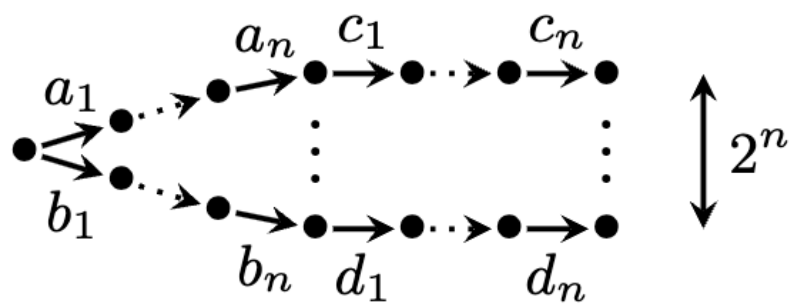
P_n



Small transition system



Big transition system

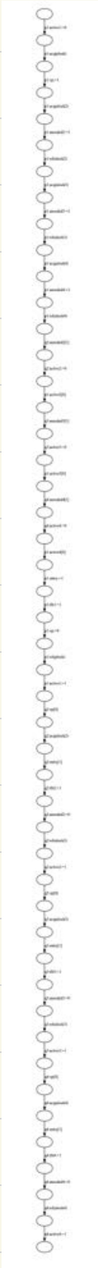


$2^n \, n!$

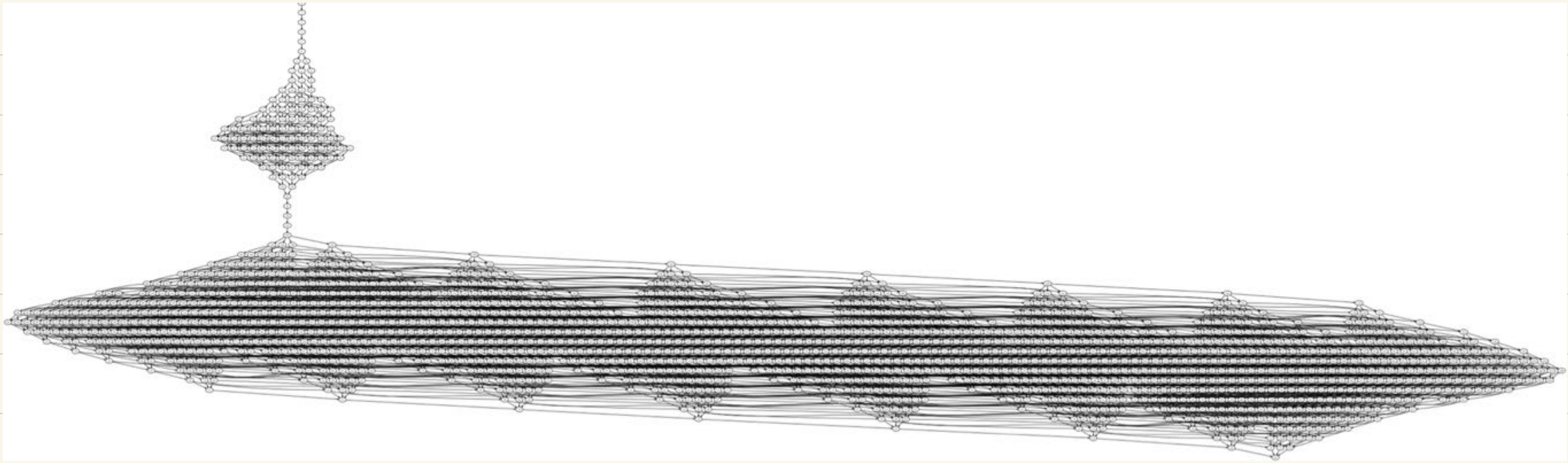
TLB shootdown

[Thread modularity on many levels, Hoenicke, Majumdar, Podelski] POPL '17

1 writer 2 readers



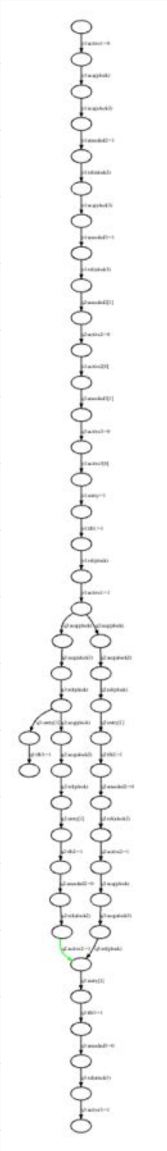
POR vs Reach



Potential applications

- looking at transition systems,
- proving correctness,
- verification of timed or probabilistic systems,
- understanding parallelism in the system.

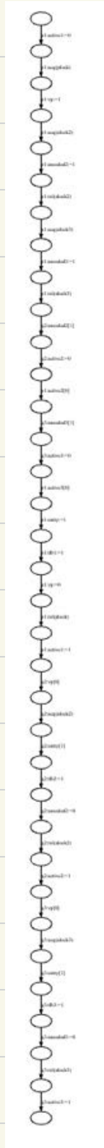
POR exposes parallelism



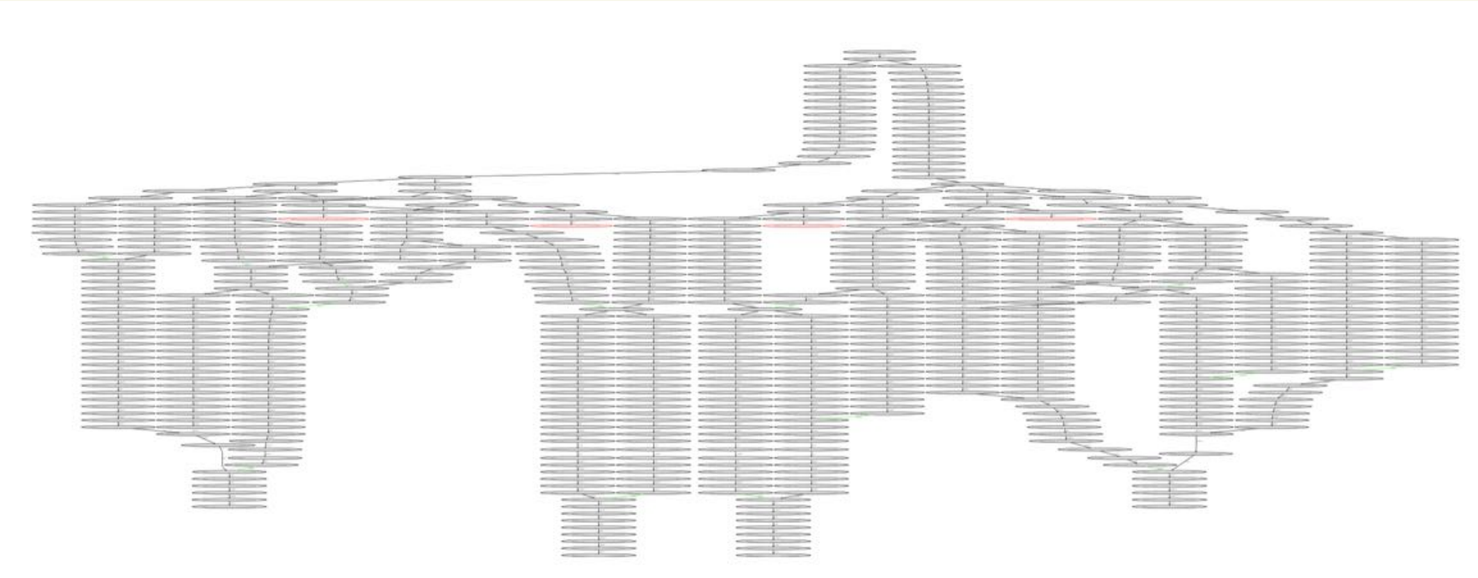
A simple correction
using a lock

sequentialization
on a lock

A more efficient correction

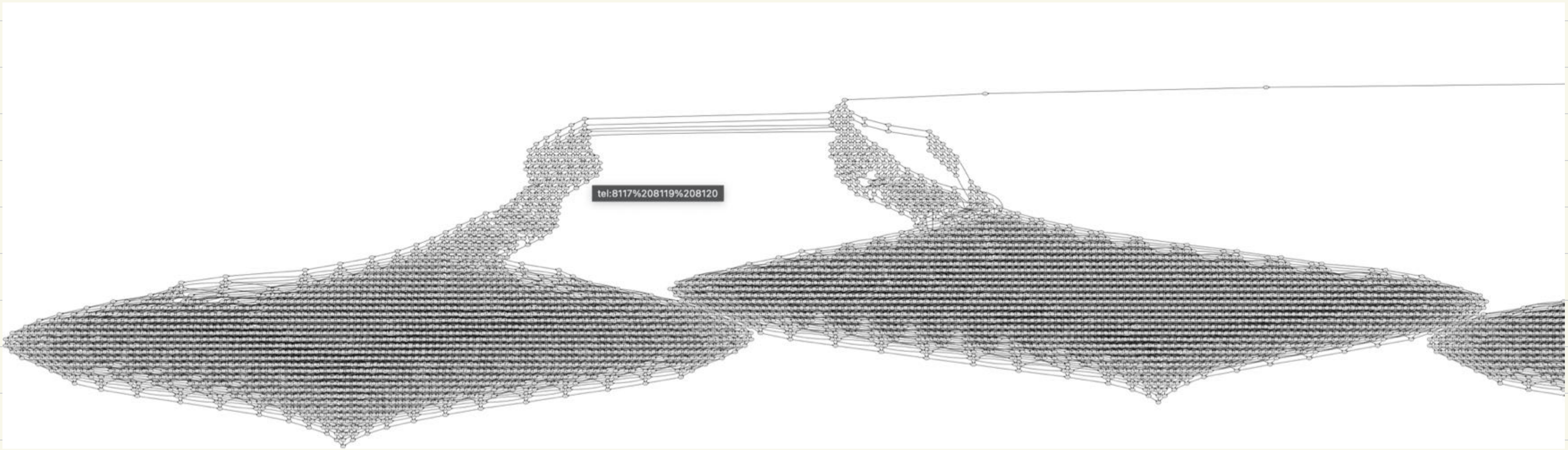


TLB shootdown



2 writers 1 reader

← POR vs Reach



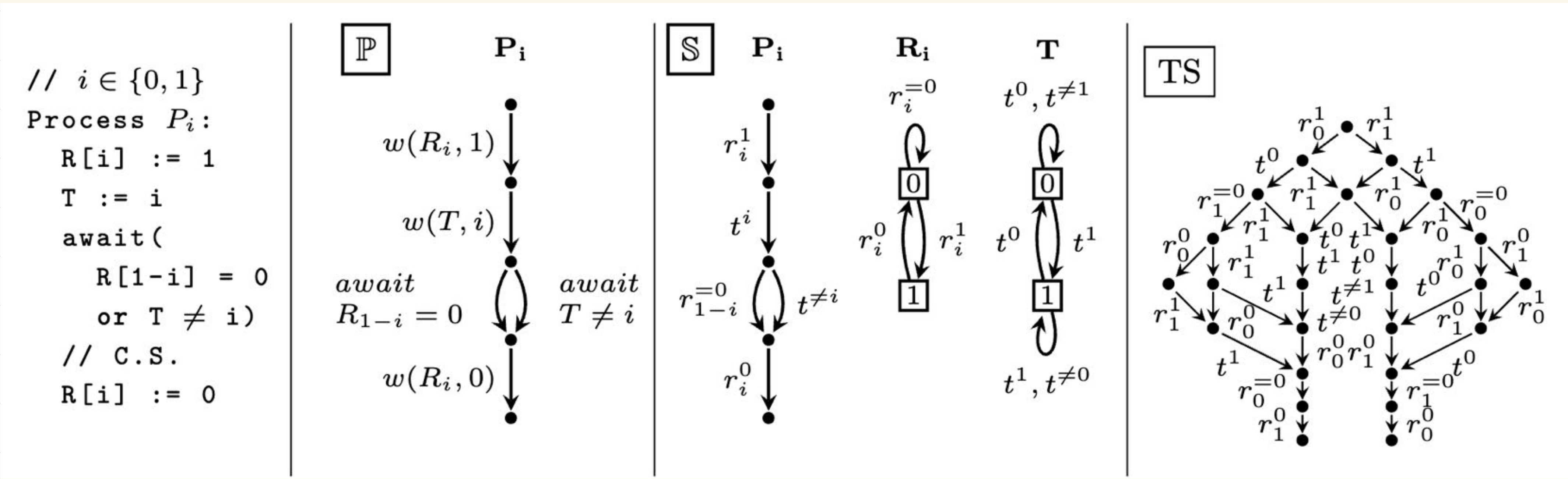
Concurrent programs

Concurrent programs

write(x, i) await(x, I) acq(ℓ) rel(ℓ)

process: a DAG whose edges are labeled with actions

every process is acyclic



Partial-order reduction

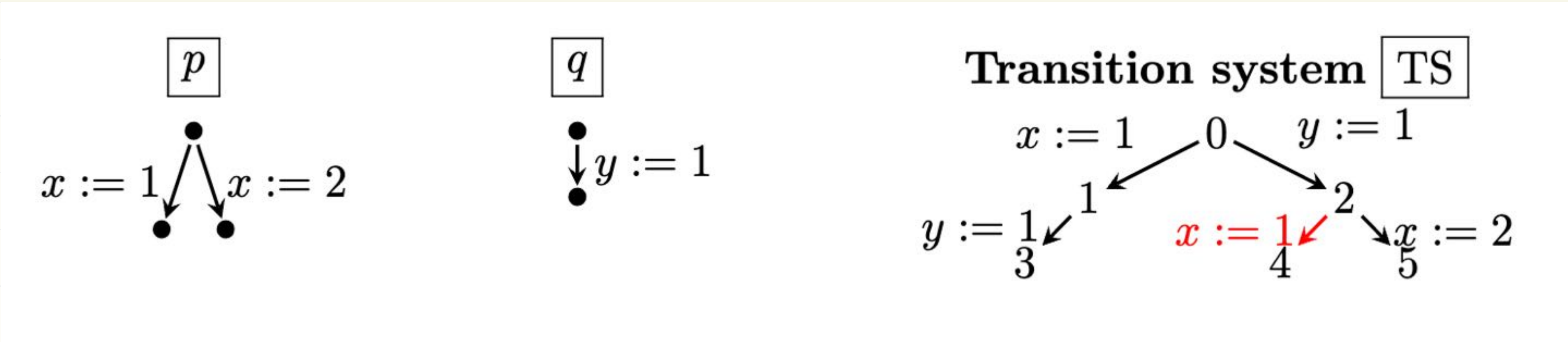
$\approx$ - equivalence relation on sequences of actions

- $a_1(x,10)\; w_2(y,13)\; a_3(x,20) \approx a_1(x,10)\; r_3(x,20)\; a_2(y,13)$
- $ab \approx ba$ if $dom(a) \cap dom(b) = \emptyset$

TS_r is a <u>reduced transition system</u> for TS if

- every full run of TS_r is a full run of TS
- $\forall$ full run u of TS $\exists$ full run v of TS_r $u \approx v$

<u>Goal</u>: construct a reduced transition system



Partial-order reduction

$\approx$ - equivalence relation on sequences of actions

- $a_1(x,10)\, w_2(y,13)\, a_3(x,20) \approx a_1(x,10)\, a_3(x,20)\, w_2(y,13)$
- $ab \approx ba$ if $dom(a) \cap dom(b) = \emptyset$

TS_r is a <u>reduced transition system</u> for TS if

- every full run of TS_r is a full run of TS
- $\forall$ full run u of TS $\exists$ full run v of TS_r $u \approx v$

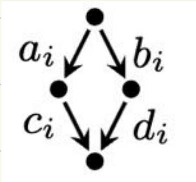
<u>Goal</u>: construct a reduced transition system

Trace optimality: TS_r has the least number of full paths.

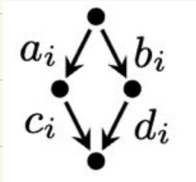
State optimality: TS_r has the least number of states.

Partial-order reduction: example

P_1

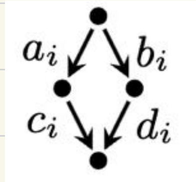


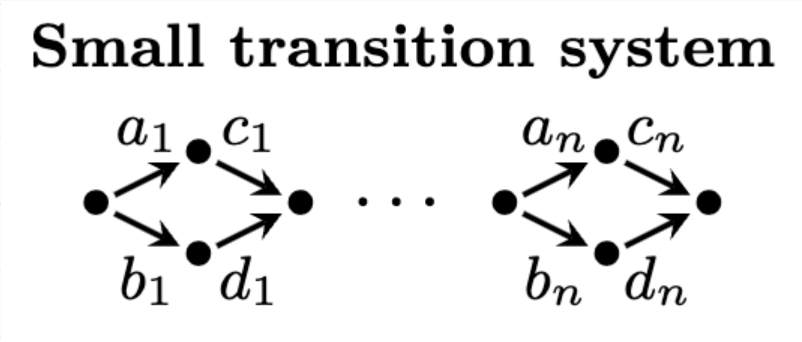
P_2

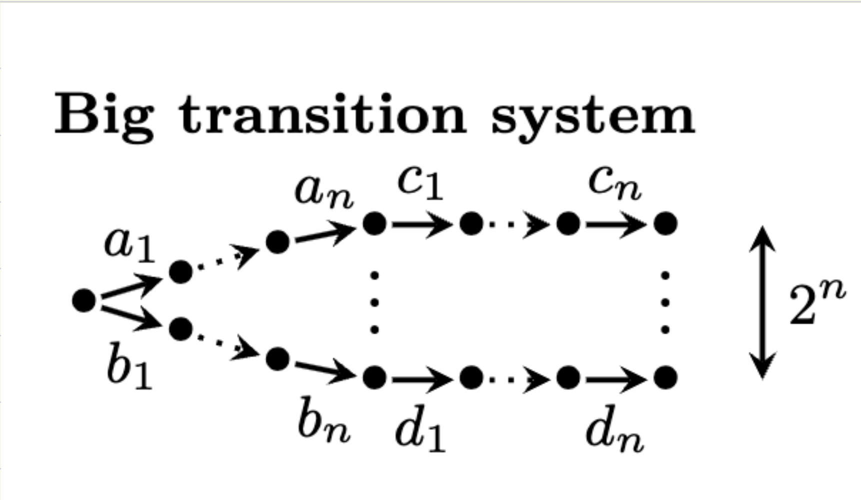


...

P_n





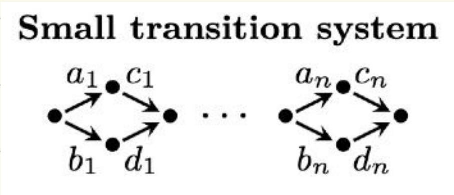


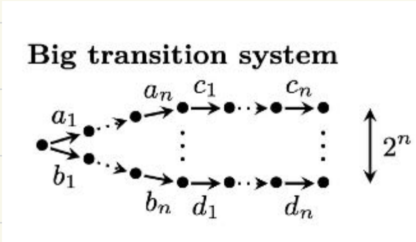
$2^n n!$

Stateful vs Stateless POR

Stateful : store visited states, this allows to stop on revisit

Stateless : store only the current path, unique solution if states are too big





Since 2005 Flanagan & Godefroid, stateless is the mainstream
2014 Abdulla

In statefull the best we have is stubborn/persistent/ample set method

- Yang, Chen, Gopalakrishnan, Kirby "Efficient Stateful Dynamic POR"
  Model Checking Software '2008

- Cirisci, Enea, Farzan, Mutluergil "A Pragmatic Approach to Statful POR"
  VMCAI '2023

Plan

1. POR is NP-hard
2. An idealized POR algorithm with IFS oracle
3. A heuristic for IFS oracle + implementation

POR is NP-hard

$minTS(\mathcal{P})$: minimal size of a reduced TS for $\mathcal{P}$

An <u>excellent POR algorithm</u> : constructs TS for $\mathcal{P}$
of size $\leq q(minTS(\mathcal{P}))$
in time $r(|\mathcal{P}| + minTS(\mathcal{P}))$

<u>Thm</u>: If $P \neq NP$ then there is no excellent POR algorithm
even for programs using only write and await operations.

POR is NP-hard

$\min TS(P)$: minimal size of a reduced TS for P

An <u>excellent POR algorithm</u>: constructs TS for P
of size $\leq q(\min TS(P))$
in time $r(|P| + \min TS(P))$

<u>Thm</u>: If $P \neq NP$ then there is no excellent POR algorithm
even for programs using only write and await operations.

Proof: Use 3-SAT. For φ construct S_φ s.t.

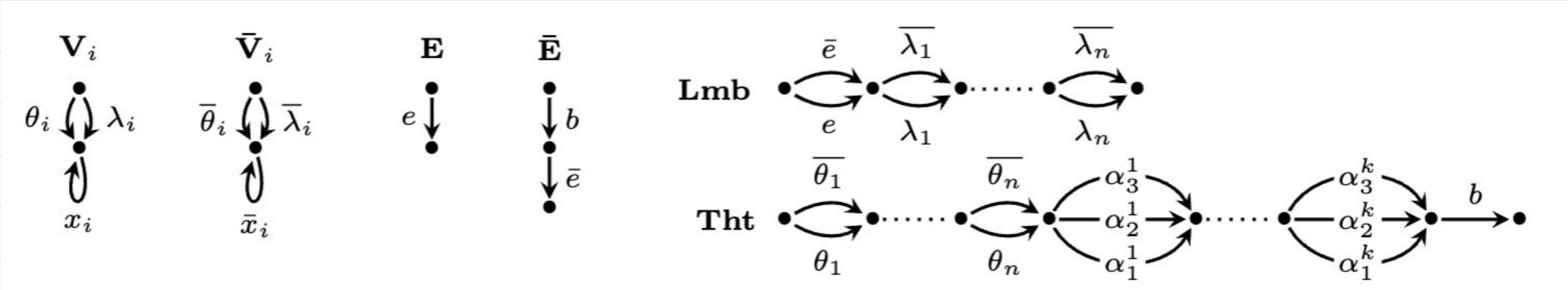
- φ not SAT $\Rightarrow$ $\min TS(S_\varphi) < 6|\varphi|$
- φ SAT $\Rightarrow$ $\min TS(S_\varphi) >$ # of sat valuations of φ

Consider $\psi \equiv \varphi \wedge (z_1 \vee z_2) \wedge \ldots \wedge (z_{2m-1} \vee z_{2m})$

Run hypothetical Alg on S_ψ for $r(6|\psi|)$ time.
If φ not SAT then Alg stops producing a TS for S_ψ
If φ SAT then Alg cannot stop as the smallest TS has $2^m > r(6|\varphi|)$ states.

$\square$

How to construct S_φ

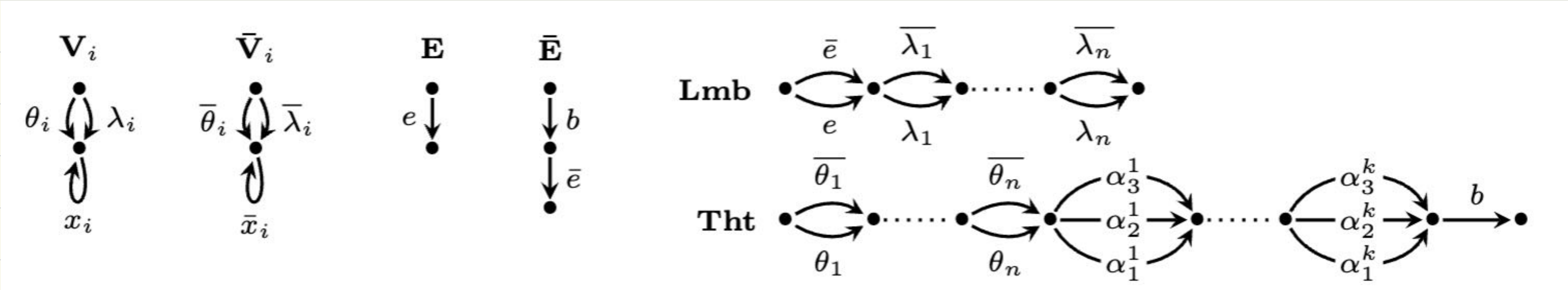


<u>L 1</u>: If φ not SAT then all runs of S_φ start with e.

<u>Proof</u>: Otherwise $b\ \bar{e}$ should appear before.

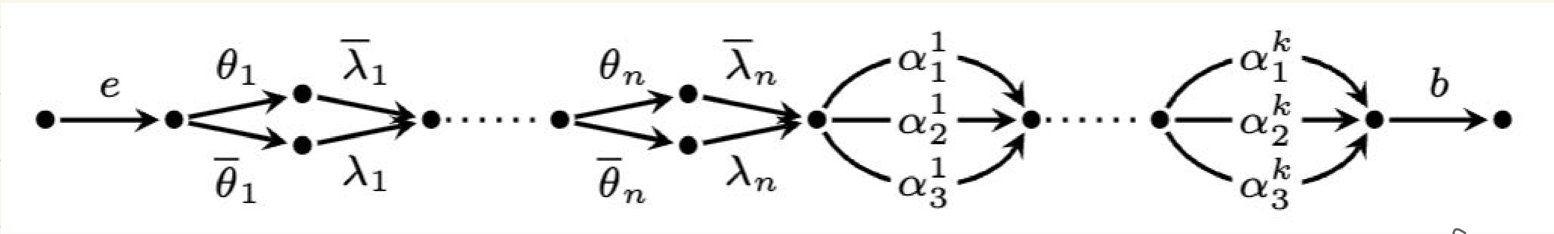
For this we need a sat valuation of φ $\square$

How to construct S_φ



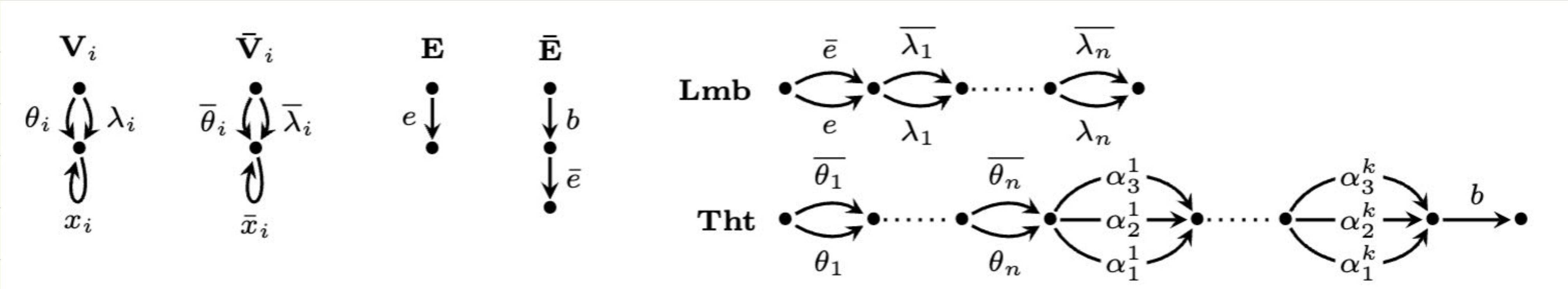
L1: If φ not SAT then all runs of S_φ start with e.
Proof: Otherwise $b\ \bar{e}$ should appear before.
For this we need a sat valuation of φ □

L2 If φ not SAT then the following is a good TS for S_φ



Every run starting from e is equivalent to a run of this form

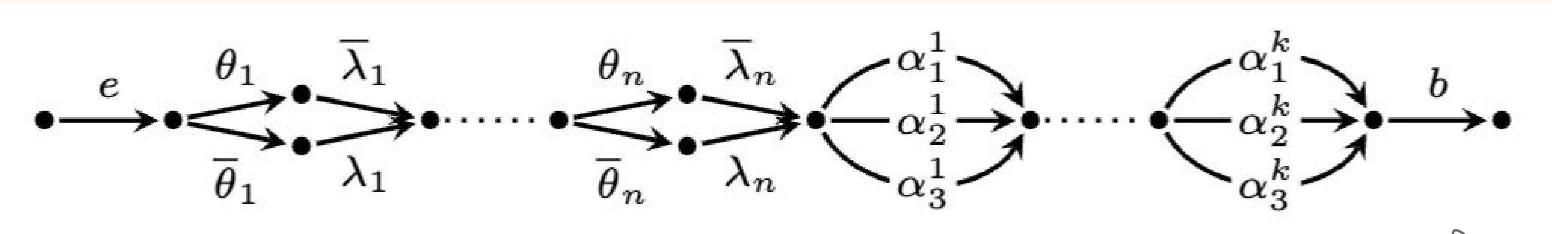
How to construct S_φ



L1: If φ not SAT then all runs of S_φ start with e.

Proof: Otherwise $b\,\bar{e}$ should appear before.
For this we need a sat valuation of φ $\square$

L2 If φ not SAT then the following is a good TS for S_φ



Every run starting from e is equivalent to a run of this form

L3 If φ SAT then there is a run starting from $\bar{e}$ for every valuation sat φ.

$$\theta_1\bar{\theta}_2\ldots\theta_n\alpha^1_{i_1}\ldots\alpha^k_{i_k}b\bar{e}\bar{\lambda}_1\lambda_2\ldots\bar{\lambda}_n$$

Plan

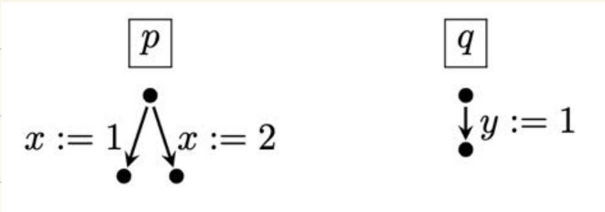
1. POR is NP-hard
2. An idealized POR algorithm with IFS oracle
3. A heuristic for IFS oracle + implementation

First sets

Goal: construct a reduced transition system

$$\mathit{first}(u) = \{ b : \exists v \;\; bv \sim u \}$$

$$\mathit{First}(s) = \{ \mathit{first}(u) : u \text{ a maximal run from } s \}$$



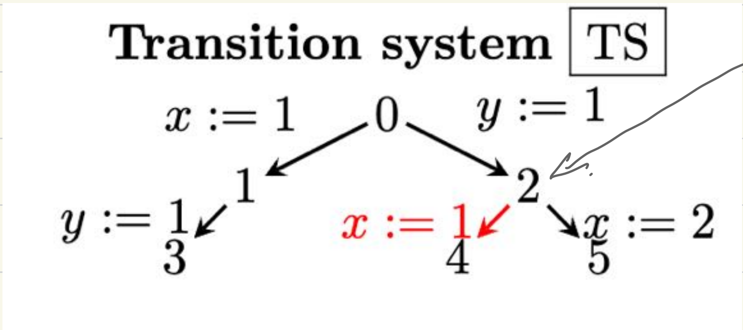
$\mathit{first}(w(x,2)\, w(y,1)) = \{w(x,2), w(y,1)\}$

$\mathit{First}(s^0) = \{\{w(x,2), w(y,1)\}, \{w(x,1), w(y,1)\}\}$

Def E is a <u>source set in</u> s if $E \cap F \neq \emptyset$ for every $F \in \mathit{First}(s)$.

Prop It is enough to find E_s for every s and explore only E_s from s.

Rem Source sets are not enough even for trace optimality.



$\{w(x,2)\}$ is not a source set here

Idealized algorithm, IFS oracle

IFS(s, B) includes first set : $\exists\ s \xrightarrow{u}$ full run $first(u) \subseteq B$

procedure *TreeExplore*(n):

  $Sl := sleep(n)$ // invariant: $Sl = sleep(n) \cup$ {labels of transitions outgoing from n}

  while $enabled(n) - Sl \neq \emptyset$

    choose smallest $e \in (enabled(n) - Sl)$ w.r.t. linear ordering on actions

    let s' **such that** $s(n) \xrightarrow{e} s'$ in $\mathrm{TS}(\mathbb{S})$

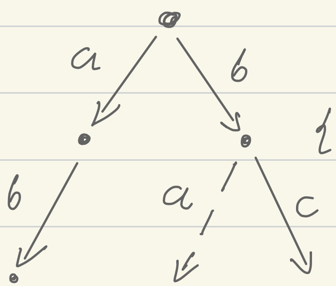
    if $IFS(s', enabled(s') - (Sl - De))$

      create node n' with $s(n') = s'$ **and** $sleep(n') = Sl - De$

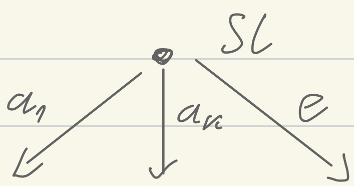
      add edge $n \xrightarrow{e} n'$

      TreeExplore(n')

    $Sl := Sl \cup \{e\}$



$\{a\}$ as $a\,I\,b$



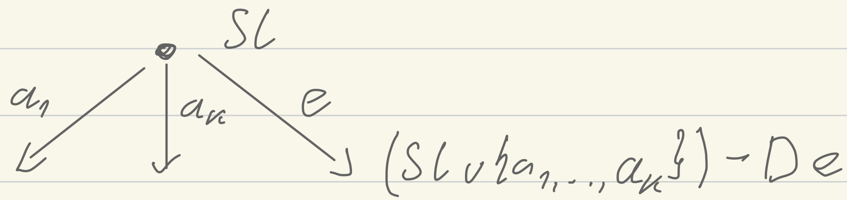
$(Sl \cup \{a_1, \ldots, a_k\}) - De$

Idealized algorithm, IFS oracle

$IFS(s, B)$ includes first set : $\exists\ s \xrightarrow{u}$ full run $first(u) \subseteq B$

procedure *TreeExplore*(n):
  $Sl := sleep(n)$ // invariant: $Sl = sleep(n) \cup \{\text{labels of transitions outgoing from } n\}$
  while $enabled(n) - Sl \neq \emptyset$
    choose smallest $e \in (enabled(n) - Sl)$ w.r.t. linear ordering on actions
    let s' **such that** $s(n) \xrightarrow{e} s'$ in $\mathrm{TS}(\mathbb{S})$
    if $IFS(s', enabled(s') - (Sl - De))$
      create node n' with $s(n') = s'$ **and** $sleep(n') = Sl - De$
      add edge $n \xrightarrow{e} n'$
      TreeExplore(n')
    $Sl := Sl \cup \{e\}$

Fact: This algorithm constructs
a trace optimal tree



Fact: $IFS(s, B)$ test is
NP-hard.

Plan

1. POR is NP-hard
2. An idealized POR algorithm with IFS oracle
3. A heuristic for IFS oracle + implementation

A heuristic for IFS(s, B)

Concurrent programs

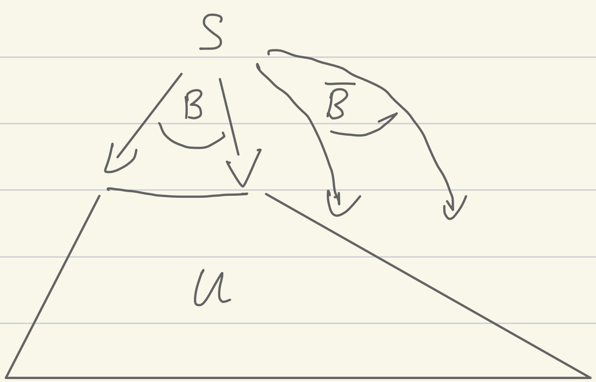
$write(x, i) \quad read(x, I) \quad acq(\ell) \quad rel(\ell)$

every process is acyclic

Equivalence $\quad u \sim v$

$$first(u) = \{ b : \exists\, bv \sim u \}$$

IFS(s, B) $\quad first(u) \subseteq B$

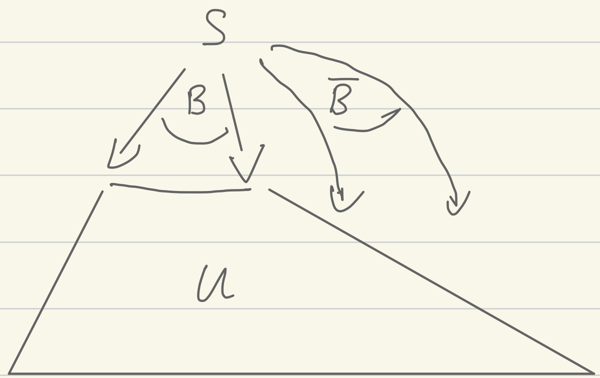


$\forall a \in \overline{B} \quad \exists\, b_a \in u \qquad a \ D \ b_a$

$$a_p(x,5) \ D \ w_q(x,6)$$
$$w_p(x,1) \ D \ a_q(x,0) \Leftarrow \text{initial read}$$

A heuristic for IFS

$IFS(s, B)$



$first(u) \subseteq B$

$\forall a \in \bar{B} \quad \exists\, b_a \in u \qquad a \ D \ b_a$

B		
p	q	r
$w(x,1)\downarrow$	$a(x,1)\downarrow$	$w(y,4)\downarrow$
$a(x,2)\downarrow$	$w(x,2)\downarrow$	
$w(y,3)\downarrow$		

1) $w(x,1) \in APW_{s,B}$
2) $w(x,2) \in APW_{s,B}$
3) $w(y,3) \in APW_{s,B}$

$w(y,3) \ D \ w(y,4)$
so $IFS(s,B)$ holds

Computing AIR, APW

B: p	B: q	r
$w(x,1)\downarrow$	$a(x,1)\downarrow$	$w(y,4)\downarrow$
$a(x,2)\downarrow$	$w(x,2)\downarrow$	
$w(y,3)\downarrow$		

1) $w(x,1) \in APW_{s,B}$
2) $w(x,2) \in APW_{s,B}$
3) $w(y,3) \in APW_{s,B}$

$w(y,3)\ D\ w(y,4)$
so $IFS(s,B)$ holds

Rem: This is only an approximation

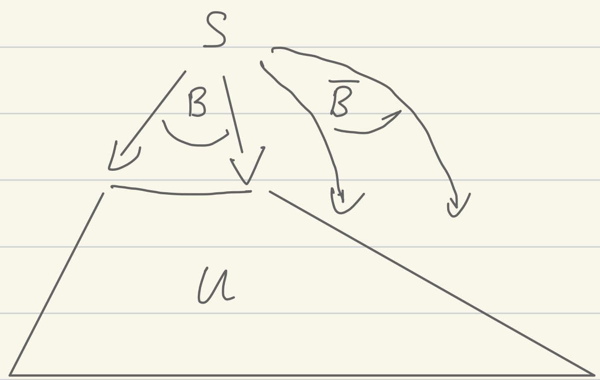
B: p'	B: q	r
$w(x,1)\downarrow$	$q\ a(x,1)\downarrow$	$w(y,4)\downarrow$
$a(x,2)\downarrow$	$w(x,2)\downarrow$	
$w(x,1)\downarrow$		
$a(x,2)\downarrow$		
$w(y,3)\downarrow$		

We still get $w(y,3) \in APW_{s,B}$

but this time it is wrong.

A heuristic for $IFS(s, B)$

$IFS(s, B)$



$first(u) \subseteq B$

$\forall a \in \overline{B} \quad \exists b_a \in u \qquad\qquad a \, D \, b_a$

$a_p(x, 5) \; D \; w_q(x, 6)$

$w_p(x, 1) \; D \; r_q(x, 0)$ ← initial read

We want to compute:

$$(AIR, APW) = \{ IR(u), PW(u)) : u \text{ a full run from } s, \; first(u) \subseteq B \}$$

all initial reads — all produced writes

Signatures

$$Sig(u) = (IR, NR, PW)$$

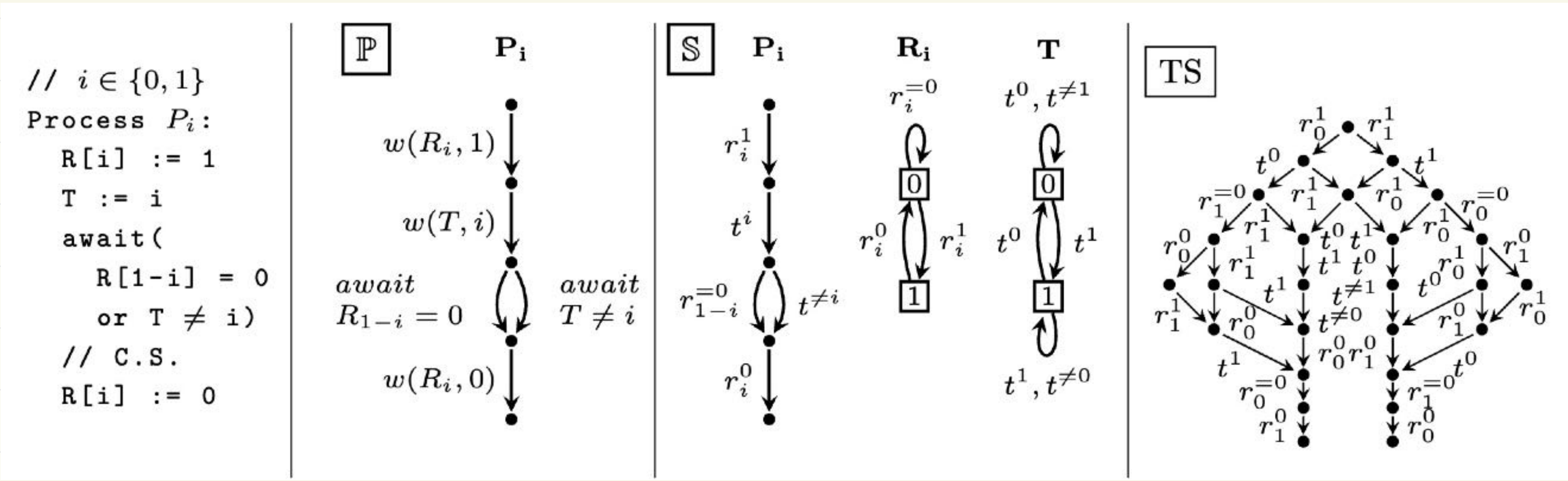
initial reads ↑ needed reads ↑ produced writes

$$Sig(a_p(x,0)\, w_p(x,1)) = (\{a(x,0)\}, \emptyset, \{w(x,1)\})$$

$$Sig(w_p(x,1)\, a_p(x,2)) = (\emptyset, \{a(x,2)\}, \{w(x,1)\})$$

We will precompute $Sigs_p(s_p, b)$ for every process p and $s_p \xrightarrow{b}$ in p

$$Sigs_p(s_p, b) = \{ Sig(u) : s_p \xrightarrow{b} \xrightarrow{u} \text{ a run} \}$$



Our approach

1) Do pre-processing: fully explore each process to compute signatures

2) At each state do poly(|P|) work to compute required PIFS(s, B)

3) Do not traverse constructed TS more than DFS does

TS grows exponentially wrt to |P| while we can hope to keep poly(|P|) small

$$|TS|^{3/2} \gg \text{poly}(|P|) \cdot |TS|$$

Rem 1: PIFS(s, B) is sometimes strictly stronger than persistent sets.

Rem 2: We use some (weak) contextual independence relation.

Experiments

Models	opifs		pifs			p-sets		spin		ic3	bdd
	N	time	N	Tr	time	N	time	N	time	time	time
bk, 3	1.4 k	0.0	3.6 k	557.0	0.0	10.7 k	0.0	11.7 k	0.0	TO	TO
bk, 4	86.4 k	15.8	197.1 k	1.6e+05	26.5	1.7 M	28.4	1.7 M	1.3	TO	TO
bk-lt, 3	331	0.0	912	85.0	0.0	2.7 k	0.0	3.0 k	0.0	272.0	637.4
bk-lt, 4	6.2 k	0.3	16.4 k	2.6e+03	0.4	129.0 k	1.4	118.6 k	0.1	TO	TO
bk-lt, 5	159.1 k	50.1	406.5 k	1.5e+05	91.8	9.2 M	171.6	7.0 M	7.2	TO	TO
bk-lt, 6	TO		TO			TO		573.6 M	1350.0	TO	TO
bk-b, 3	632	0.0	1.7 k	281.0	0.0	4.8 k	0.0	4.6 k	0.0	8.9	288.4
bk-b, 4	17.9 k	1.1	49.2 k	2.4e+04	1.6	357.3 k	3.8	284.2 k	0.4	TO	TO
bk-b, 5	650.2 k	305.4	1.8 M	5.3e+06	596.6	42.9 M	945.9	27.3 M	86.9	TO	TO
bk-ba, 3	277	0.0	803	95.0	0.0	2.8 k	0.0	3.0 k	0.0	9.3	710.3
bk-ba, 4	4.2 k	0.2	11.4 k	3.2e+03	0.3	152.7 k	1.6	141.2 k	0.1	TO	TO
bk-ba, 5	74.2 k	28.7	197.8 k	2.1e+05	58.2	14.4 M	271.0	11.2 M	23.9	TO	TO
dp-o, 20	8.1 k	0.9	29.1 k	1e+06	1.1	TO		TO		4.4	TO
dp-o, 29	33.0 k	6.9	119.6 k	5.4e+08	7.8	TO		TO		9.8	TO
dp-o, 30	37.9 k	7.9	135.8 k	1.1e+09	8.9	TO		TO		TO	TO
dp-o, 50	276.2 k	149.2	986.2 k	1.1e+15	154.2	TO		TO		TO	TO
dp-b, 11	256.6 k	20.9	724.1 k	4.8e+06	25.2	TO		TO		10.6	TO
dp-b, 13	1.3 M	146.4	3.6 M	7.7e+07	170.8	TO		TO		6.6	TO
dp-b, 15	6.2 M	913.6	17.4 M	1.2e+09	1151.3	TO		TO		8.3	TO
dp-ao, 16	33.9 k	3.9	86.5 k	1.7e+07	4.0	TO		TO		TO	TO
dp-ao, 30	473.5 k	149.5	1.1 M	3.6e+13	139.4	TO		TO		TO	TO
lz, 12	16.4 k	0.5	36.8 k	1.5e+04	0.7	11.8 M	298.6	16.1 M	41.1	23.0	TO
lz, 13	32.7 k	1.1	73.7 k	3.3e+04	1.6	43.9 M	1926.3	59.9 M	195.0	200.8	TO
lz, 14	65.5 k	2.5	147.4 k	7e+04	3.5	TO		223.6 M	884.0	149.9	TO
lz, 16	262.1 k	11.1	589.7 k	3.1e+05	16.2	TO		TO		TO	TO
lz, 18	1.0 M	50.4	2.4 M	1.4e+06	72.7	TO		TO		320.6	TO
lz, 20	4.2 M	230.8	9.4 M	6e+06	368.5	TO		TO		TO	TO
lz, 22	16.8 M	1330.8	37.7 M	2.6e+07	1948.0	TO		TO		TO	TO
pet, 3	41	0.0	124	32.0	0.0	210	0.0	276	0.0	13.5	7.6
pet, 4	630	0.0	1.7 k	2.8e+03	0.0	3.4 k	0.0	4.5 k	0.0	TO	TO
pet, 5	15.5 k	1.2	39.3 k	3.9e+06	1.3	106.1 k	1.4	129.6 k	0.2	TO	TO
pet, 6	633.6 k	150.2	1.5 M	2e+11	182.6	5.4 M	107.2	5.8 M	22.2	TO	TO
pet, 7	TO		TO			TO		371.8 M	1090.0	TO	TO
rcu, 7	26.4 k	2.1	93.1 k	7.8e+04	2.9	76.1 M	2115.2	55.4 M	147.0	2.9	2307.1
rcu, 8	100.8 k	9.7	355.7 k	3.9e+05	14.1	TO		602.5 M	2090.0	3.7	TO
rcu, 9	388.1 k	48.7	1.4 M	2e+06	74.0	TO		TO		13.5	TO
szy, 4	3.1 k	0.1	9.0 k	1.1e+05	0.1	34.5 k	0.2	29.1 k	0.0	1605.8	323.3
szy, 5	41.4 k	2.6	112.1 k	1.3e+08	3.3	868.5 k	9.0	603.1 k	1.1	TO	TO
szy, 6	620.9 k	69.0	1.6 M	1.6e+12	93.8	25.1 M	416.1	14.1 M	42.8	TO	TO
szy, 7	TO		TO			TO		365.9 M	762.0	TO	TO
tlb, 2, 1	125	0.0	552	24.0	0.0	12.9 k	0.1	14.1 k	0.0	TO	TO
tlb, 2, 2	607	0.0	2.6 k	160.0	0.0	307.4 k	4.3	327.9 k	0.9	TO	TO
tlb, 3, 1	13.6 k	1.1	58.0 k	2.8e+04	1.5	2.7 M	45.9	1.7 M	5.7	TO	TO
tlb, 3, 2	173.6 k	19.8	745.5 k	1.8e+06	25.8	TO		52.0 M	177.0	TO	TO
tlb, 4, 1	2.3 M	330.4	9.6 M	3e+08	425.8	TO		183.2 M	618.0	TO	TO
token, 3	990	0.0	2.0 k	3.9e+04	0.0	20.6 k	0.1	21.1 k	0.0	16.6	250.5
token, 4	31.9 k	1.3	65.6 k	1.7e+15	1.5	1.5 M	18.0	1.5 M	1.9	TO	TO
token, 5	589.6 k	34.0	1.2 M	1.9e+18	39.0	64.5 M	1562.7	62.9 M	150.0	TO	TO
token, 6	8.1 M	700.3	17.1 M	1.6e+19	806.5	TO		TO		TO	TO

19
57
66
189
bakery

dinning philosophers

last zero

8,5 peterson

read-copy-update

40 szymański

79 TLB cache coherence

108 demand driven token-ring

Conclusions

- POR is computationally difficult, so we need heuristics.
- We do not have a convienient success measure (state optimality)
- Our heuristic is quite satisfactory for variables and locks.

TODO

1) Handling cycles
2) Symmetry reduction.
3) Weak memory models
4) PIFS for other constructs (channels, ...).