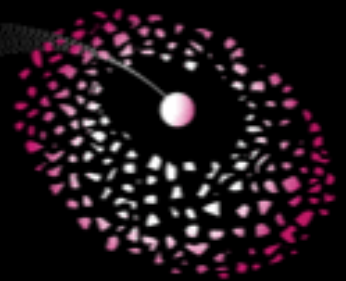
UNIVERSITY OF TWENTE.
formal methods & tools.

Improved On-The-Fly Livelock Detection with $\mathrm{DFS}_{\mathrm{FIFO}}$

Alfons Laarman

Joint with David Faragó
(Karlsruhe Institute of Technology)

May 14th, 2013

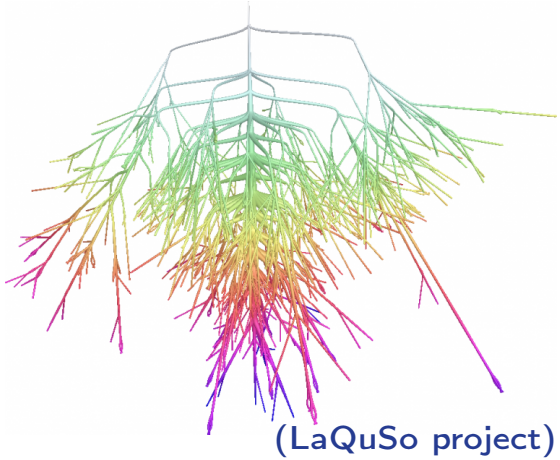


NFM, Moffett Field, CA, USA

Dealing with State Space Explosion

Problem

State Explosion in LTL Model Checking



(LaQuSo project)

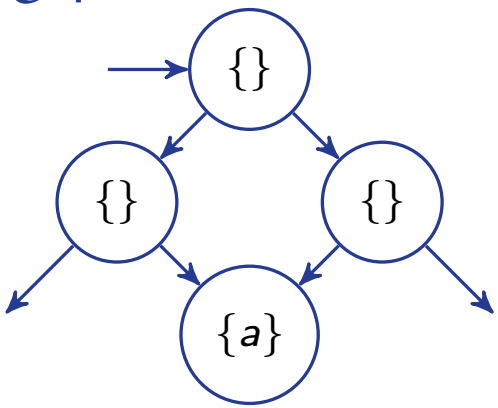
Viable solutions:

- Parallelization
- ↕
- Partial-Order/Confluence Reduction
- ↕
- Specialization on subclasses of LTL (Livelocks, Weak LTL)

LTL Model Checking

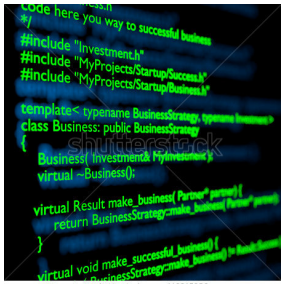
$\mathcal{M}$:

$\mathcal{TS}$:

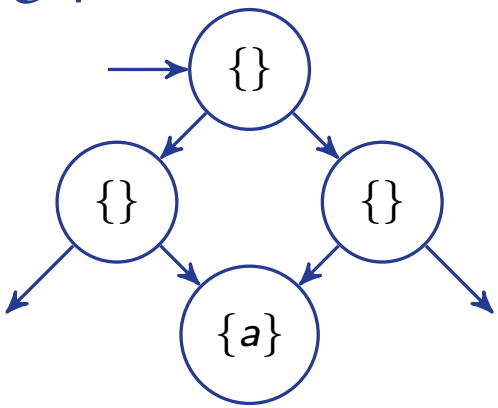


LTL Model Checking

$\mathcal{M}$:



$\mathcal{TS}$:



$\mathcal{M} \models \varphi$

LTL formula, e.g.:

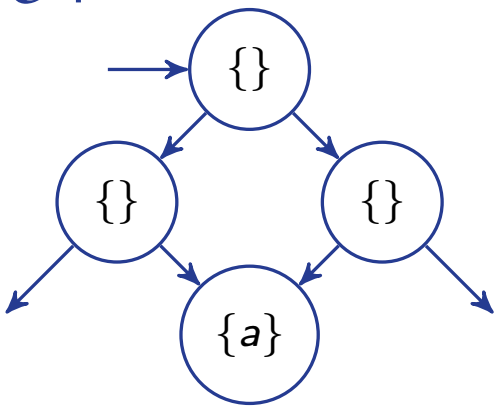
$\varphi = \Box(a \Rightarrow \Diamond b)$

LTL Model Checking

$\mathcal{M}$:

$\downarrow$

$\mathcal{TS}$:



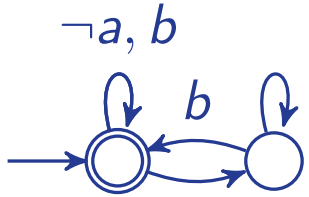
$$\mathcal{M} \models \varphi$$
$$\Updownarrow$$
$$\mathcal{L}(\mathcal{TS}) \subseteq \mathcal{L}(\mathcal{A}_\varphi)$$

LTL formula, e.g.:

$\varphi = \Box(a \Rightarrow \Diamond b)$

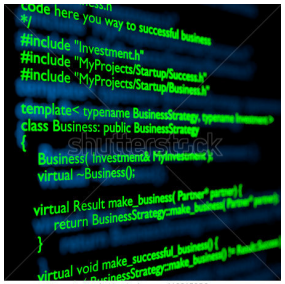
$\downarrow$

$\mathcal{A}_\varphi$ (Büchi) :



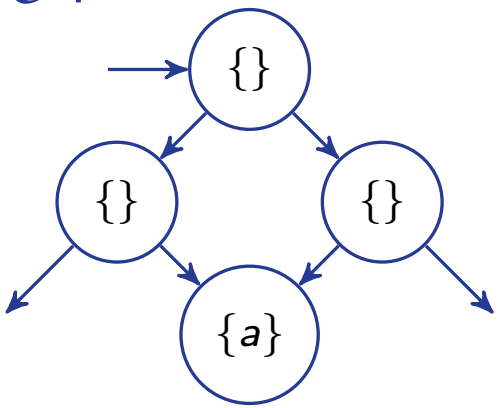
LTL Model Checking

$\mathcal{M}$:



$\downarrow$

$\mathcal{TS}$:



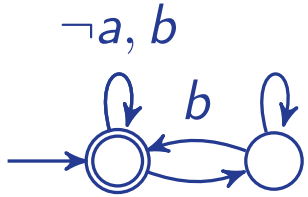
$$\mathcal{M} \models \varphi$$
$$\Updownarrow$$
$$\mathcal{L}(\mathcal{TS}) \subseteq \mathcal{L}(\mathcal{A}_\varphi)$$
$$\Updownarrow$$
$$\mathcal{L}(\mathcal{TS}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi}) = \emptyset$$

LTL formula, e.g.:

$\varphi = \Box(a \Rightarrow \Diamond b)$

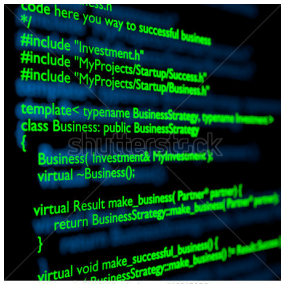
$\downarrow$

$\mathcal{A}_\varphi$ (Büchi) :



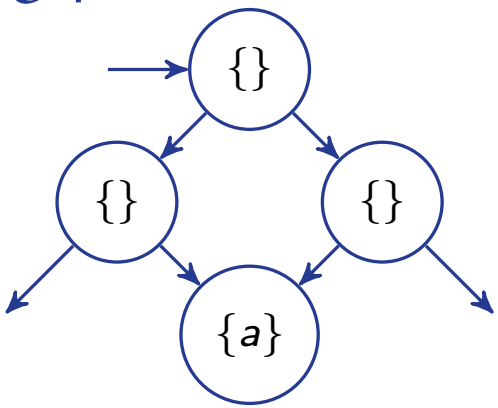
LTL Model Checking

$\mathcal{M}$:



$\downarrow$

$\mathcal{TS}$:



$$\mathcal{M} \models \varphi$$
$$\Updownarrow$$
$$\mathcal{L}(\mathcal{TS}) \subseteq \mathcal{L}(\mathcal{A}_\varphi)$$
$$\Updownarrow$$
$$\mathcal{L}(\mathcal{TS}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi}) = \emptyset$$
$$\Updownarrow$$
$$\mathcal{TS} \otimes \mathcal{A}_{\neg\varphi}$$

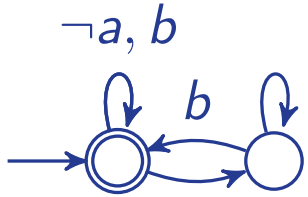
contains no accepting cycle

LTL formula, e.g.:

$\varphi = \Box(a \Rightarrow \Diamond b)$

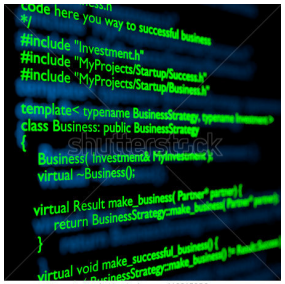
$\downarrow$

$\mathcal{A}_\varphi$ (Büchi) :



LTL Model Checking

$\mathcal{M}$:



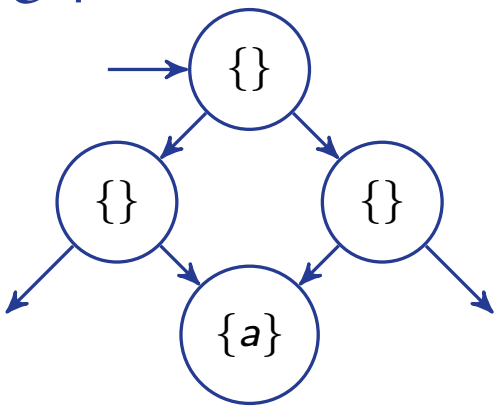
$$\mathcal{M} \models \varphi$$
$$\Updownarrow$$
$$\mathcal{L}(\mathcal{TS}) \subseteq \mathcal{L}(\mathcal{A}_\varphi)$$
$$\Updownarrow$$
$$\mathcal{L}(\mathcal{TS}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi}) = \emptyset$$
$$\Updownarrow$$
$$\mathcal{TS} \otimes \mathcal{A}_{\neg\varphi}$$

contains no accepting cycle

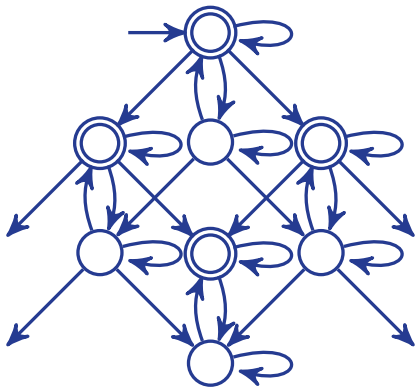
LTL formula, e.g.:

$\varphi = \Box(a \Rightarrow \Diamond b)$

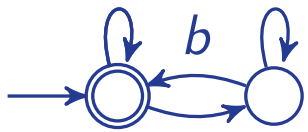
$\mathcal{TS}$:



$\mathcal{C}$:

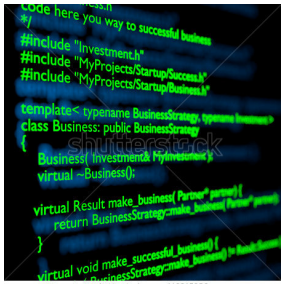


$\mathcal{A}_\varphi$ (Büchi) :



LTL Model Checking

$\mathcal{M}$:



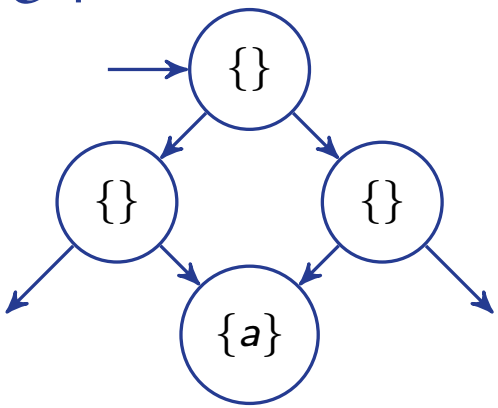
$$\mathcal{M} \models \varphi$$
$$\Updownarrow$$
$$\mathcal{L}(\mathcal{TS}) \subseteq \mathcal{L}(\mathcal{A}_\varphi)$$
$$\Updownarrow$$
$$\mathcal{L}(\mathcal{TS}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi}) = \emptyset$$
$$\Updownarrow$$
$$\mathcal{TS} \otimes \mathcal{A}_{\neg\varphi}$$

contains no accepting cycle

LTL formula, e.g.:

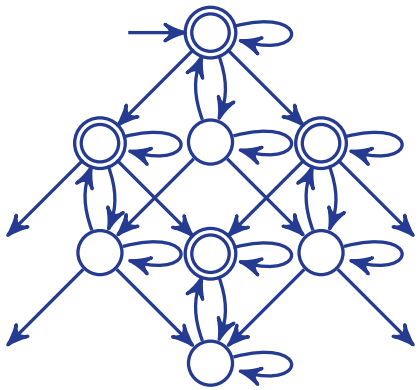
$\varphi = \Box(a \Rightarrow \Diamond b)$

$\mathcal{TS}$:



2^C

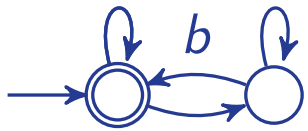
$\mathcal{C}$:



$2^C \times 2^{AP}$

$\mathcal{A}_\varphi$ (Büchi) :

$\neg a, b$



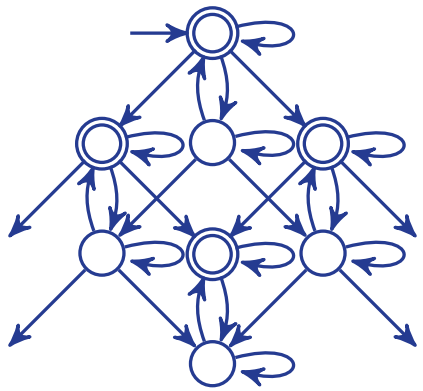
2^{AP}

Nested Depth-First Search

Algorithm

- Two DFS searches
- Linear ($2N$) in size of the cross product
- DFS order (hard to parallelize)

$\mathcal{C}$:



CNDFS for Parallel LTL Model Checking

Simple idea: *P independent, randomized* NDFS instances (swarm)
Store states in lockless hash/tree table [FMCAD 2010/SPIN 2011]

+ More on the fly (bug hunting)
−− No speedup for full verification

CNDFS for Parallel LTL Model Checking

Simple idea: *P independent, randomized* NDFS instances (swarm)
Store states in lockless hash/tree table [FMCAD 2010/SPIN 2011]

+ More on the fly (bug hunting)

−− No speedup for full verification

Better idea: Store information in graph to prune work:
Multi-core NDFS $\rightarrow$ CNDFS [ATVA 2011/2012]

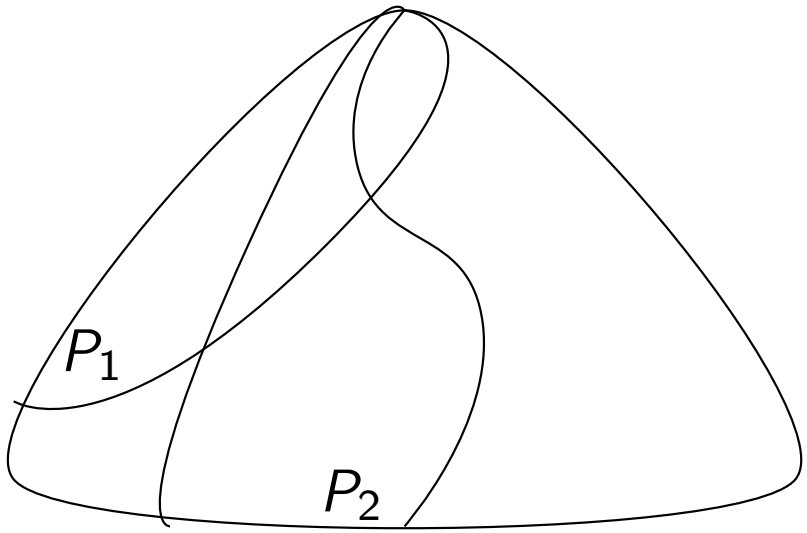
CNDFS for Parallel LTL Model Checking

Simple idea: *P independent, randomized* NDFS instances (swarm)
Store states in lockless hash/tree table [FMCAD 2010/SPIN 2011]

\+ More on the fly (bug hunting)

−− No speedup for full verification

Better idea: Store information in graph to prune work:
Multi-core NDFS → CNDFS [ATVA 2011/2012]



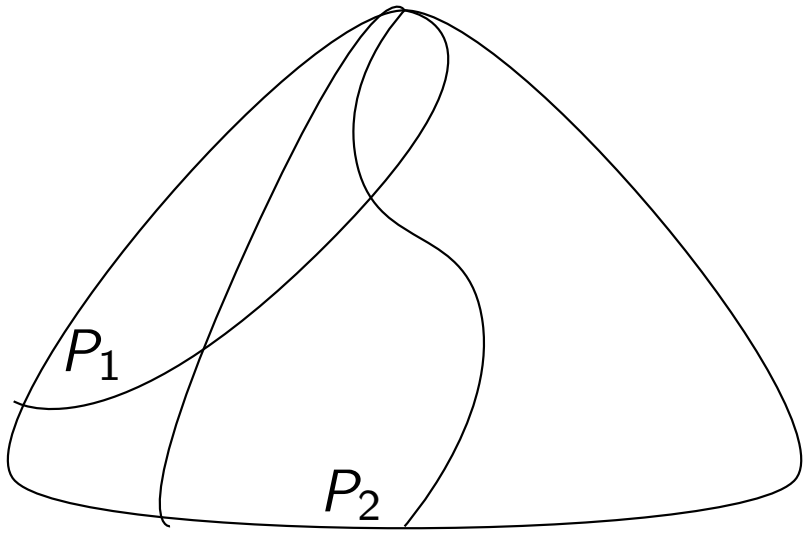
CNDFS for Parallel LTL Model Checking

Simple idea: *P independent, randomized* NDFS instances (swarm)
Store states in lockless hash/tree table [FMCAD 2010/SPIN 2011]

\+ More on the fly (bug hunting)

−− No speedup for full verification

Better idea: Store information in graph to prune work:
Multi-core NDFS → CNDFS [ATVA 2011/2012]



Worst case complexity: $N \times P$, but in practice close to N

CNDFS Experiments

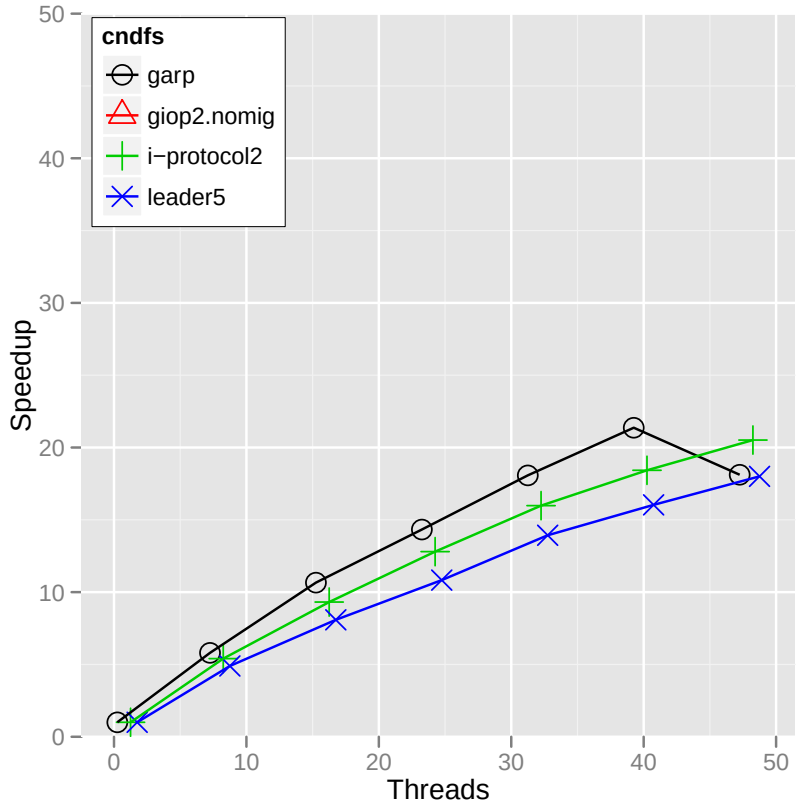
LTL property: $\Box\Diamond progress$

	SPIN	LTSmin	
	NDFS	CNDFS	
		1 core	48 cores
$?extleader_t$	1390	926	51
garp	2050	1061	59
i-prot	103	76	4

CNDFS Experiments

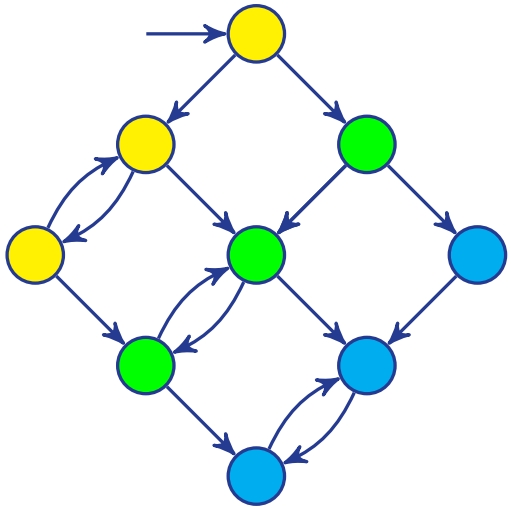
LTL property: $\Box\Diamond progress$

	SPIN	LTSmin	
	NDFS	CNDFS	
		1 core	48 cores
$?extleader_t$	1390	926	51
garp	2050	1061	59
i-prot	103	76	4



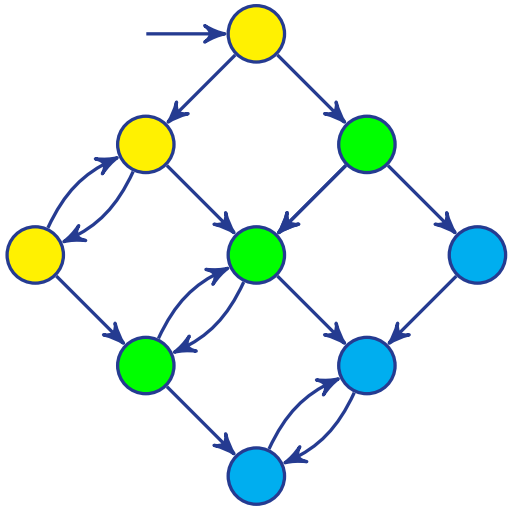
Partial-Order Reduction

$\mathcal{TS}$:



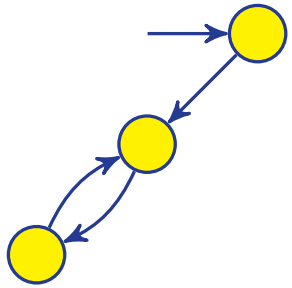
Partial-Order Reduction

$\mathcal{TS}$:



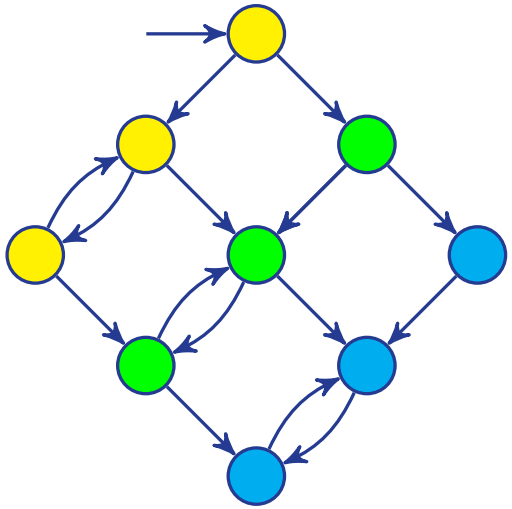
deadlock $\longrightarrow$

$\mathcal{TS}$:



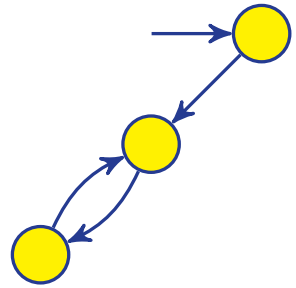
Partial-Order Reduction

$\mathcal{TS}$:



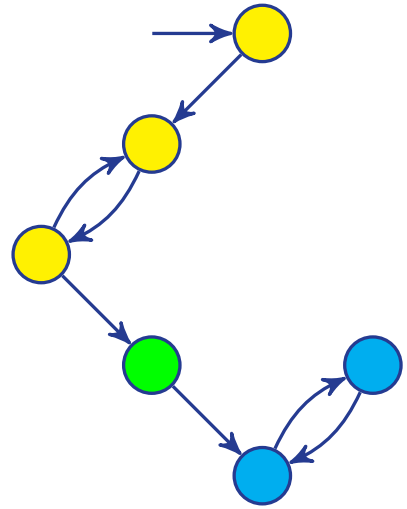
deadlock $\longrightarrow$

$\mathcal{TS}$:

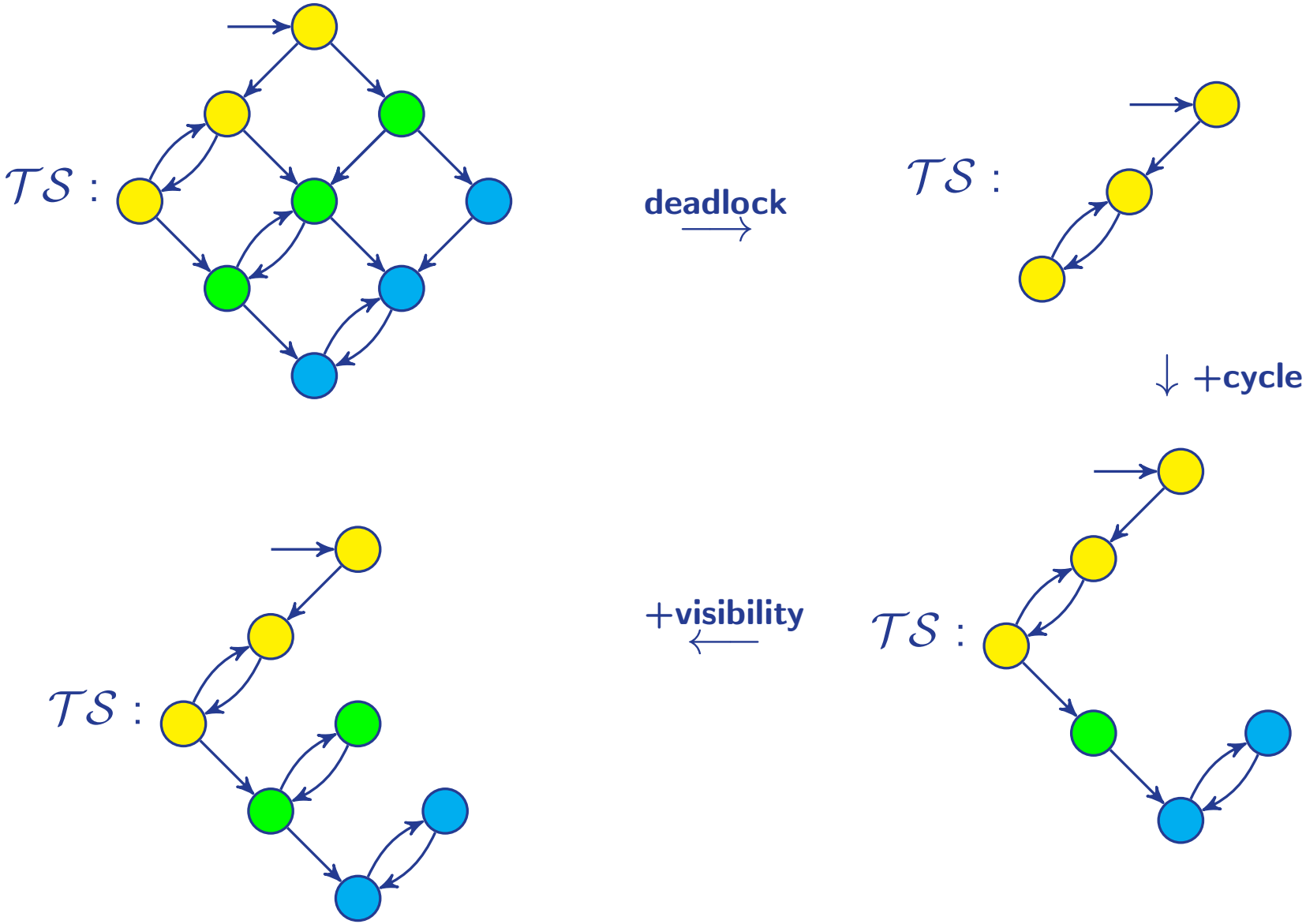


$\downarrow$ **+cycle**

$\mathcal{TS}$:



Partial-Order Reduction



Partial-Order Reduction for LTL

Partial-Order Reduction for (C)NDFS

- stack proviso for NDFS

Partial-Order Reduction for LTL

Partial-Order Reduction for (C)NDFS

- stack proviso for NDFS
- CNDFS has incomplete stacks $\Rightarrow$ No POR

Partial-Order Reduction for LTL

Partial-Order Reduction for (C)NDFS

- stack proviso for NDFS
- CNDFS has incomplete stacks $\Rightarrow$ No POR

Other solutions to solve the problem of parallelism & POR:
OWCTY + Topological Sort Proviso [Barnat et al. 2010]

Specializing on Livelock Detection

Livelock LTL property: $\Box\Diamond\mathcal{P}$

Livelocks are important properties, they are used for:

- $\frac{1}{3}$ of all the BEEM models
- $\frac{1}{2}$ of the models in the Promela Database (`http://www.albertolluch.com/research/promelamodels`)

Specializing on Livelock Detection

Livelock LTL property: $\Box\Diamond\mathcal{P}$

Livelocks are important properties, they are used for:

- $\frac{1}{3}$ of all the BEEM models
- $\frac{1}{2}$ of the models in the Promela Database (`http://www.albertolluch.com/research/promelamodels`)

- Progress detection DFS [Holzmann et al. 1996]
- DFS_{FIFO} with Partial-Order Reduction [Farago et al. 2009]

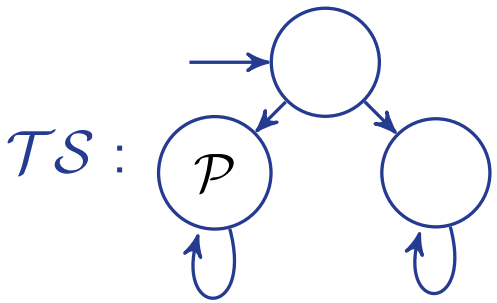
Specializing on Livelock Detection

Livelock LTL property: $\Box\Diamond\mathcal{P}$

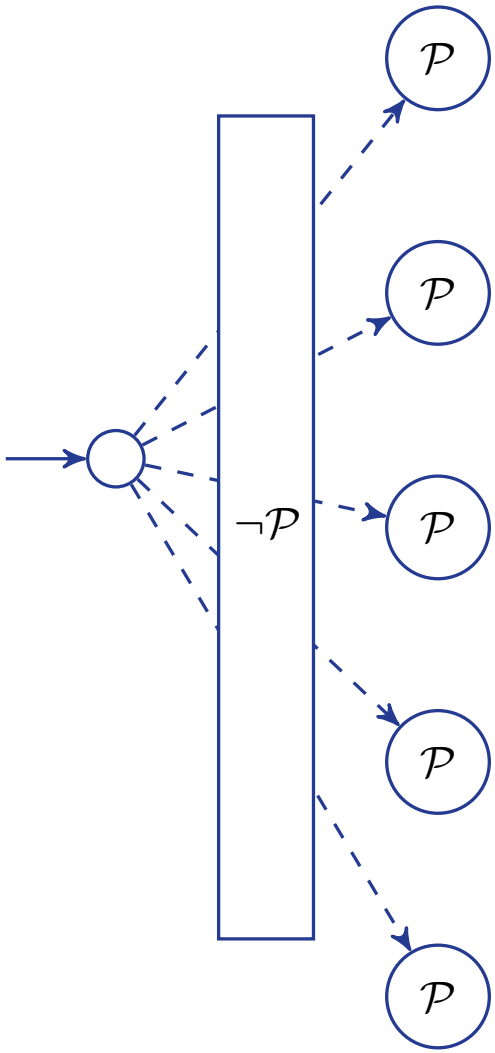
Livelocks are important properties, they are used for:

- $\frac{1}{3}$ of all the BEEM models
- $\frac{1}{2}$ of the models in the Promela Database (`http://www.albertolluch.com/research/promelamodels`)

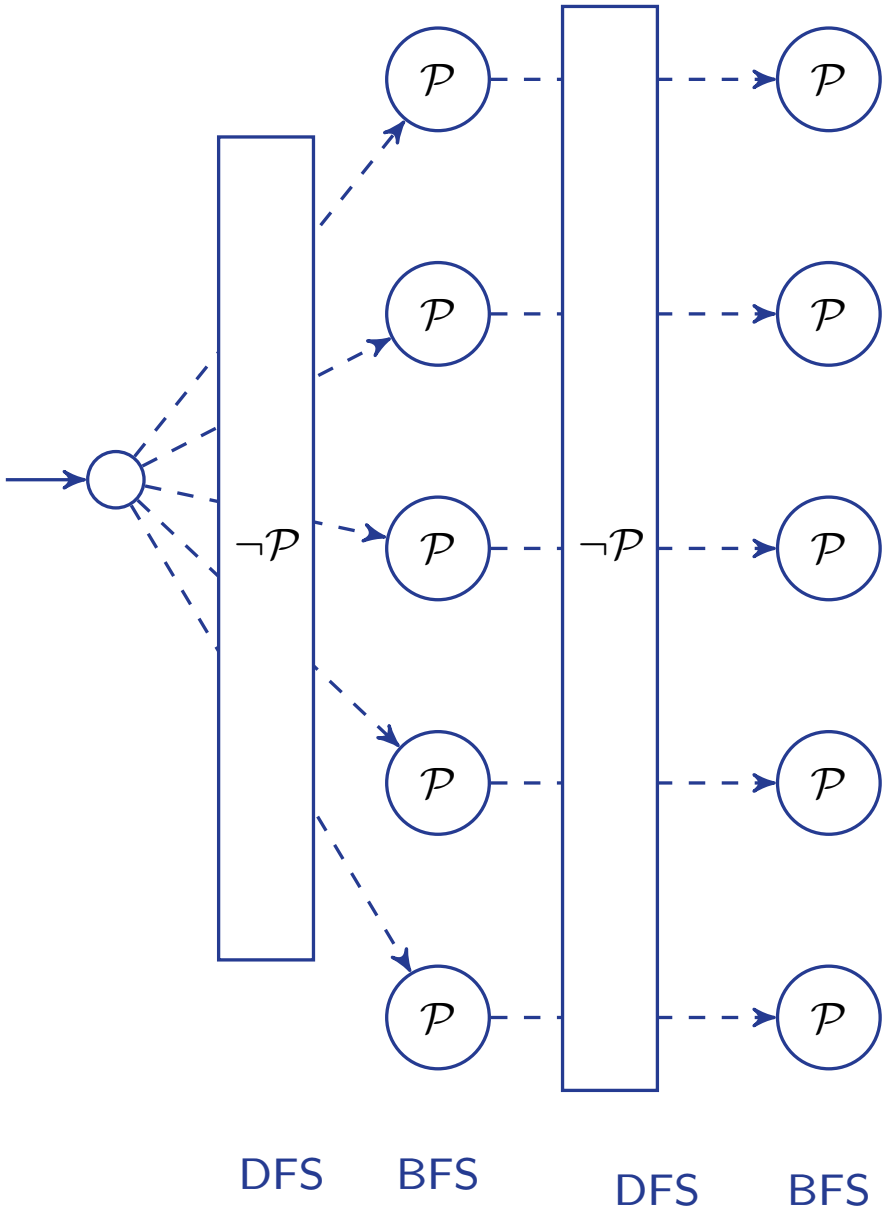
- Progress detection DFS [Holzmann et al. 1996]
- DFS_{FIFO} with Partial-Order Reduction [Farago et al. 2009]



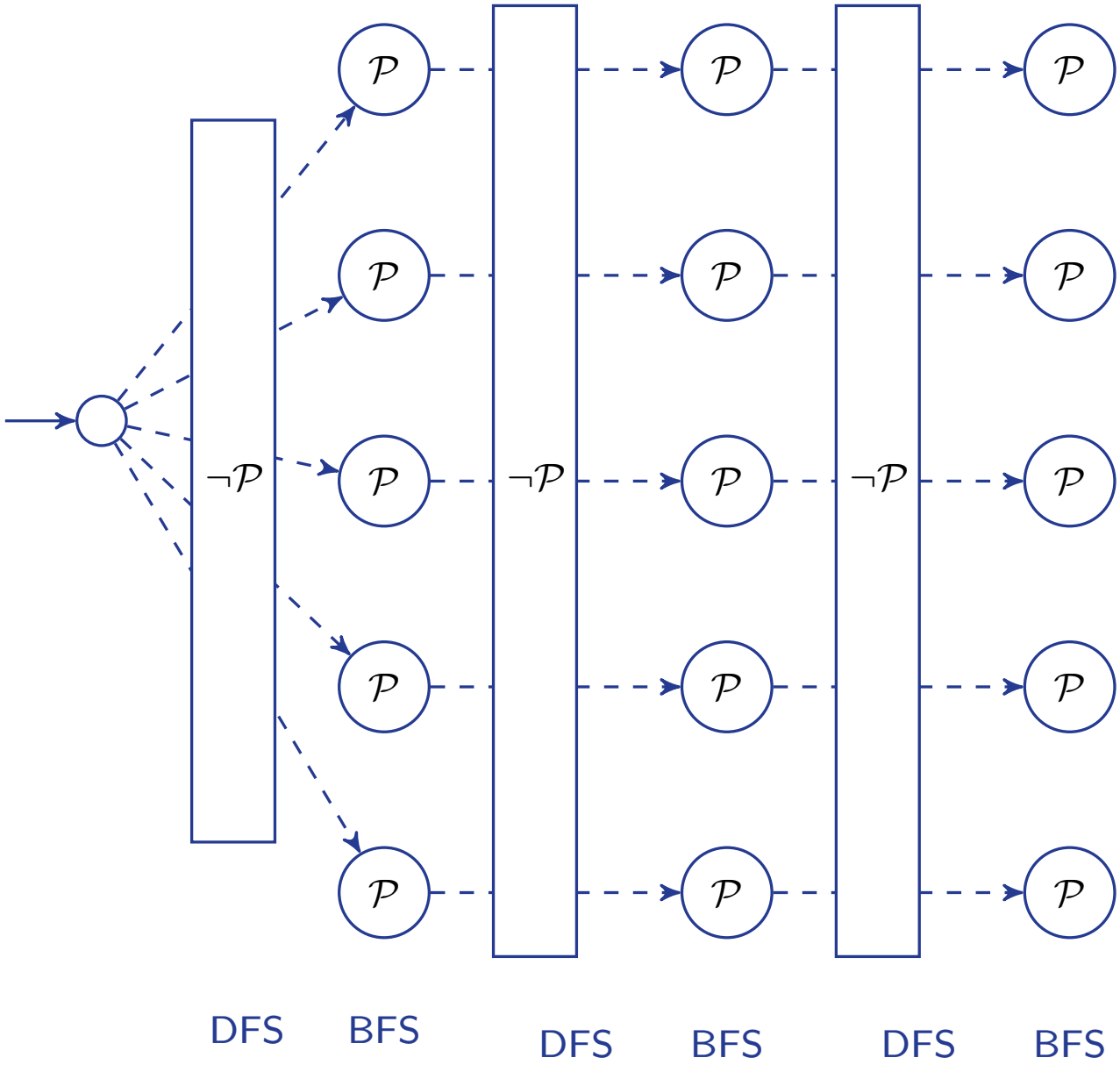
$\mathrm{DFS}_{\mathrm{FIFO}}$ for Livelock Detection



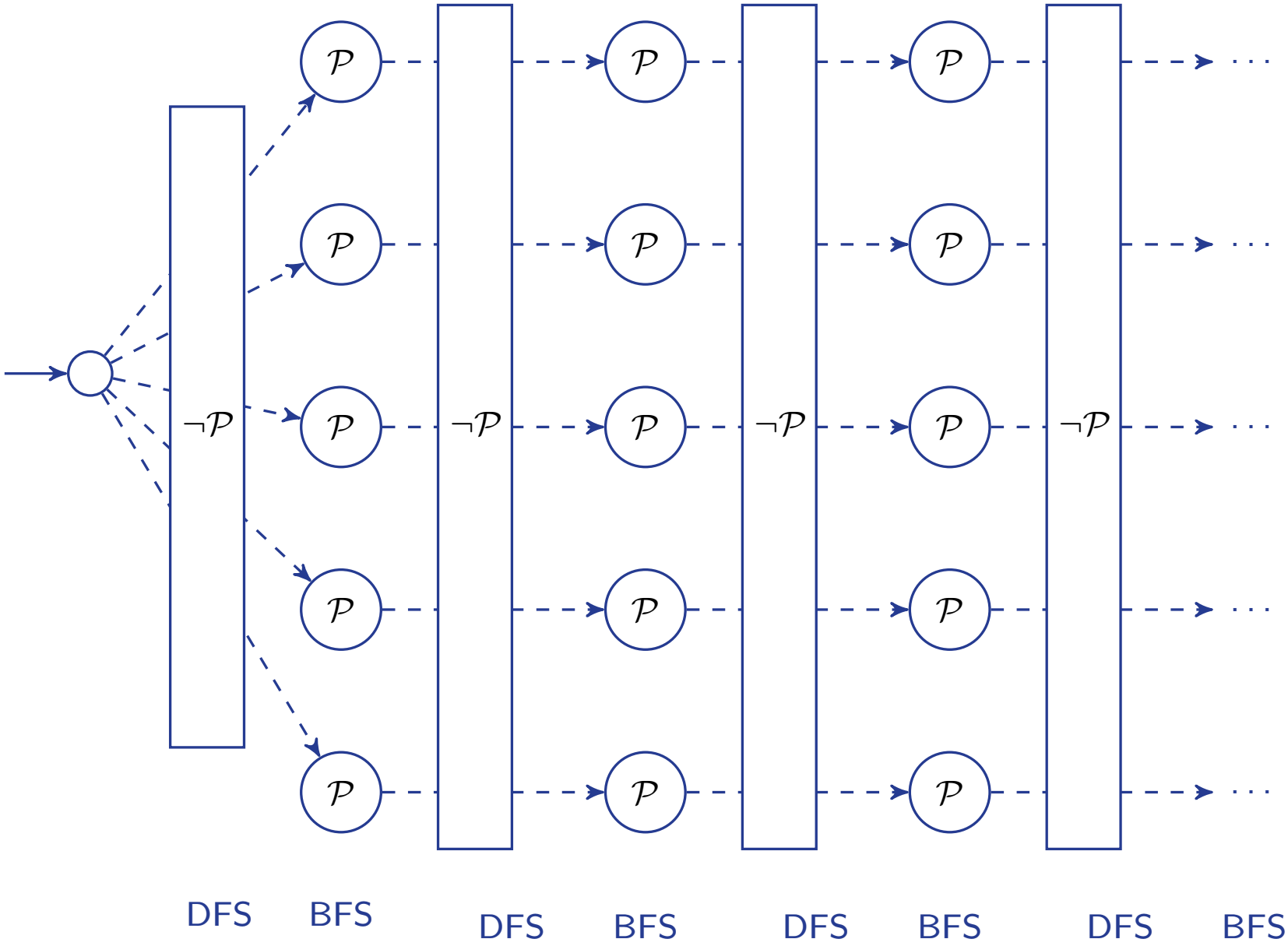
$\mathrm{DFS}_{\mathrm{FIFO}}$ for Livelock Detection



DFS_{FIFO} for Livelock Detection



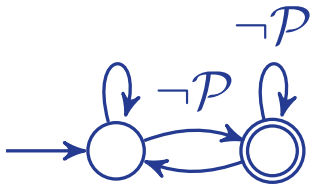
DFS_{FIFO} for Livelock Detection



DFS$_{\mathrm{FIFO}}$ Experiments

Expected outcome:

- DFS$_{\mathrm{FIFO}}$ makes single pass over state space (unlike NDFS)
- $|\mathcal{TS}| \leq \frac{1}{2}|\mathcal{C}|$

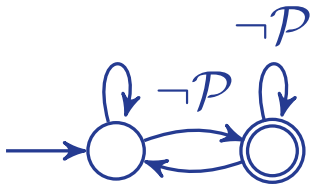


Up to 4 times as fast as NDFS

DFS_{FIFO} Experiments

Expected outcome:

- DFS_{FIFO} makes single pass over state space (unlike NDFS)
- $|\mathcal{TS}| \leq \frac{1}{2}|\mathcal{C}|$



Up to 4 times as fast as NDFS

	LTSmin (sec)	
	DFS_{FIFO}	NDFS
garp	591.2	969.2
i-prot	41.4	70.6
?*extleader*$_t$	233.2	753.6

DFS_{FIFO} and Partial-Order Reduction

Cycle proviso

- progress cycle $\Rightarrow$ visibility proviso
- non-progress cycle $\Rightarrow$ report counter example

No additional stack proviso required!

DFS_{FIFO} and Partial-Order Reduction

Cycle proviso

- progress cycle $\Rightarrow$ visibility proviso
- non-progress cycle $\Rightarrow$ report counter example

No additional stack proviso required!

Model	States	LTSmin (POR) DFS_{FIFO}	LTSmin (POR) NDFS	SPIN (POR) NDFS
garp	72,318,749	2.2%	32.6%	18.3%
i-protocol	20,052,267	31.8%	32.0%	41.4%
$?extleader_t$	89,771,572	0.5%	0.5%	1.2%

DFS_{FIFO} and Partial-Order Reduction

Cycle proviso

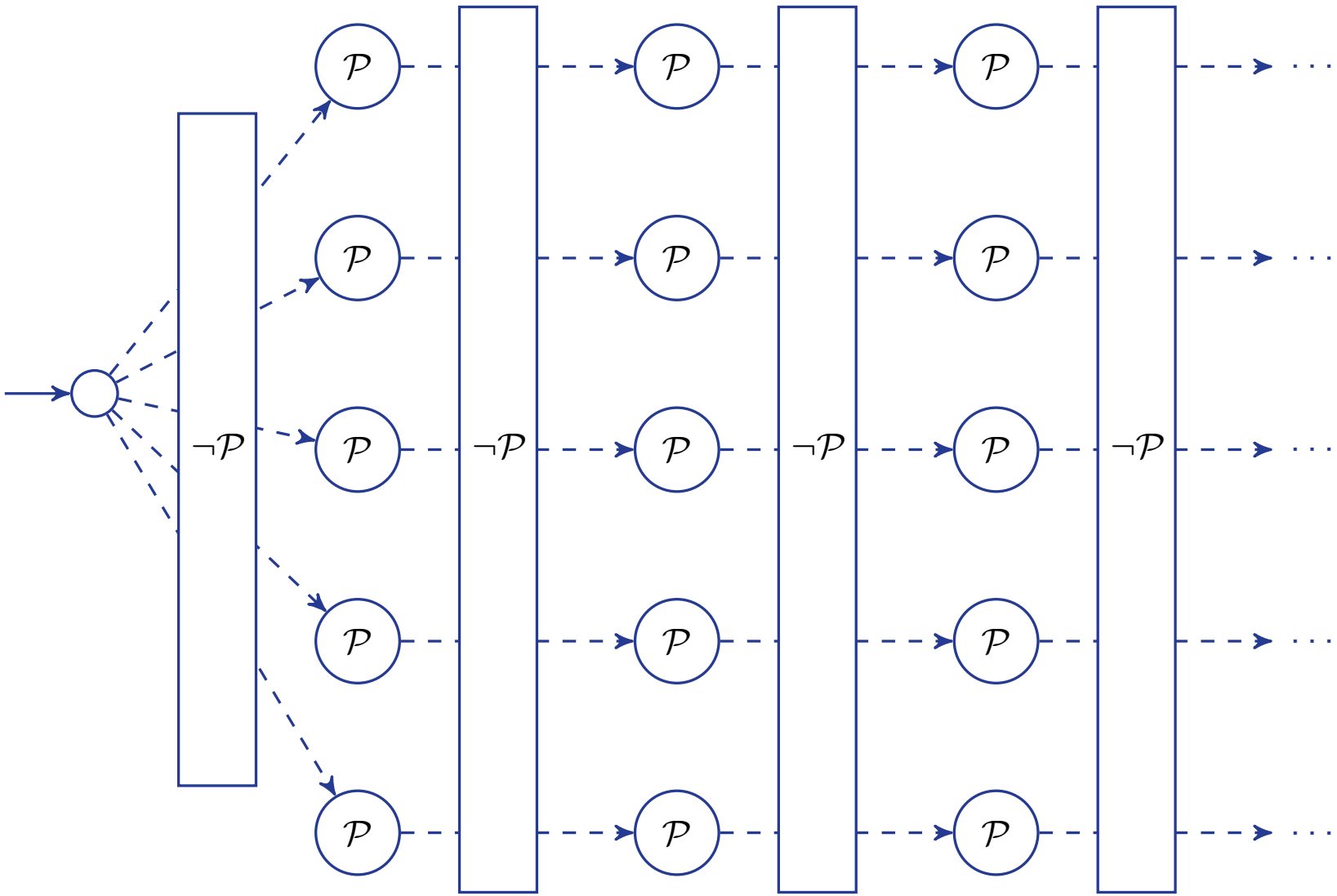
- progress cycle $\Rightarrow$ visibility proviso
- non-progress cycle $\Rightarrow$ report counter example

No additional stack proviso required!

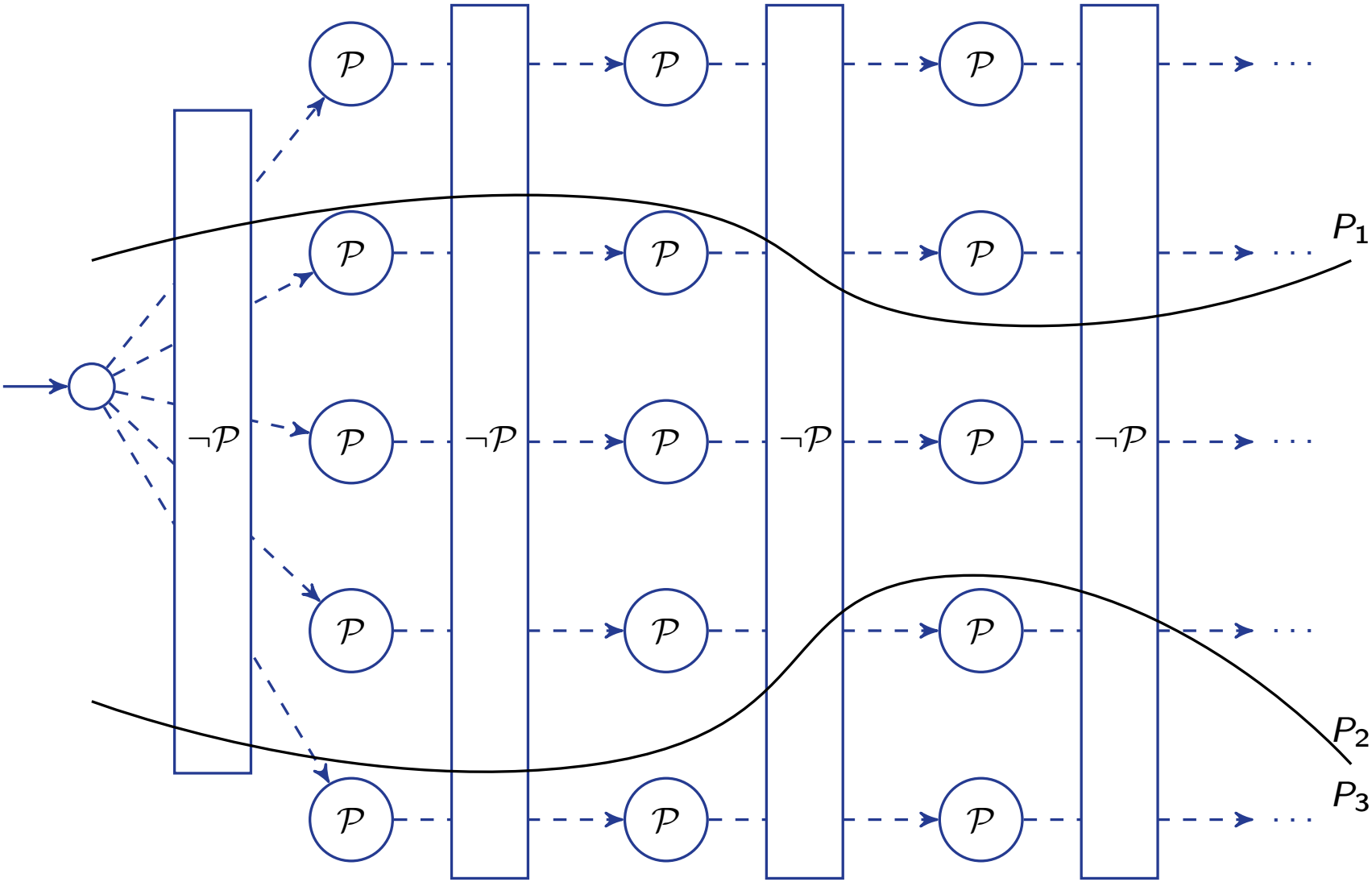
Model	States	LTSmin (POR) DFS_{FIFO}	LTSmin (POR) NDFS	SPIN (POR) NDFS
garp	72,318,749	2.2%	32.6%	18.3%
i-protocol	20,052,267	31.8%	32.0%	41.4%
$?extleader_t$	89,771,572	0.5%	0.5%	1.2%

LTSmin uses the stubborn set method [SPIN 2013]

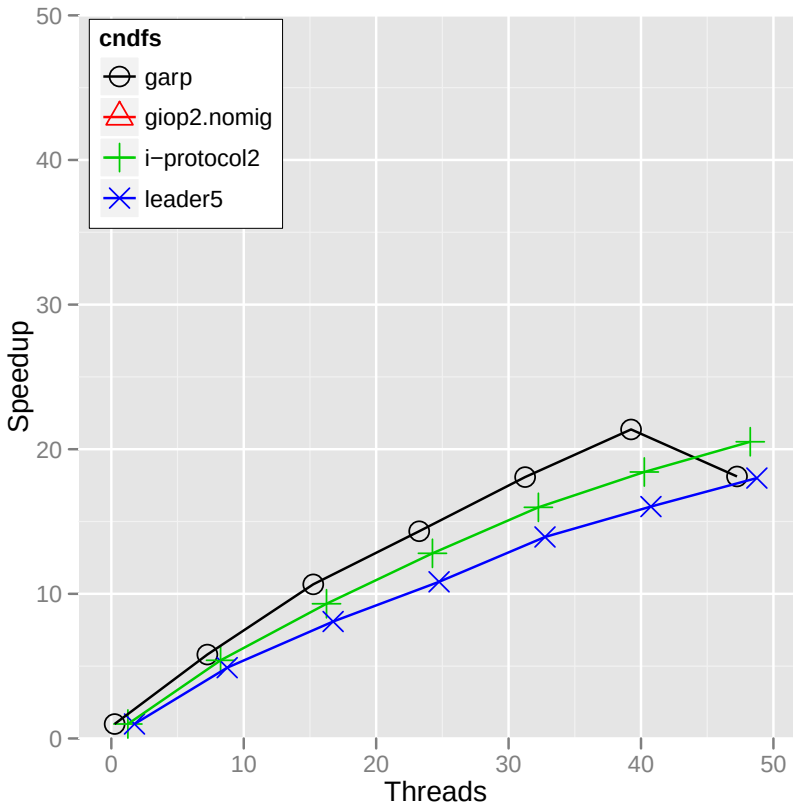
Parallel DFS$_{\mathrm{FIFO}}$

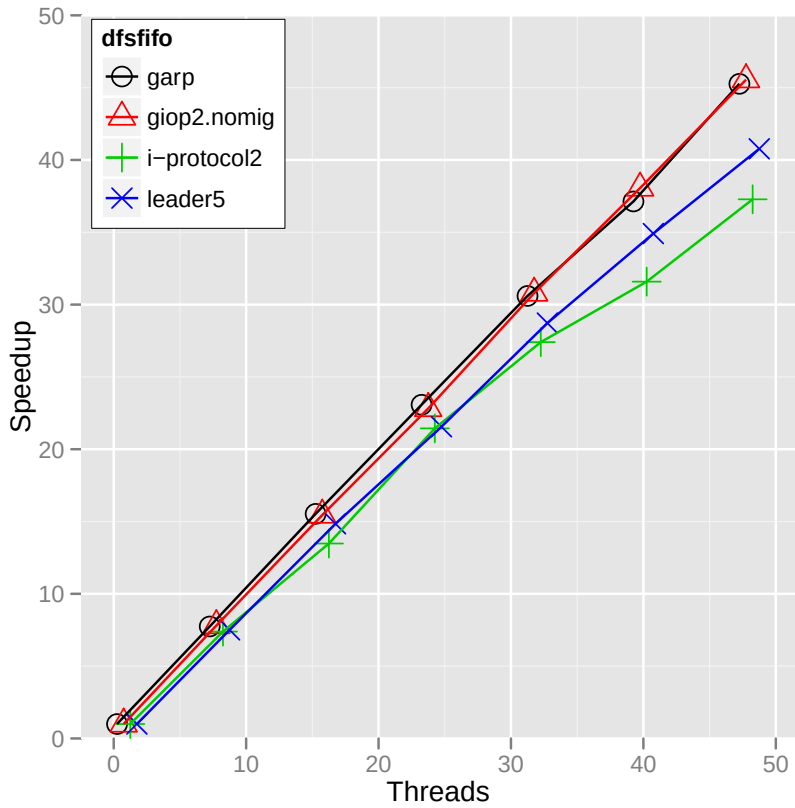


Parallel DFS_{FIFO}

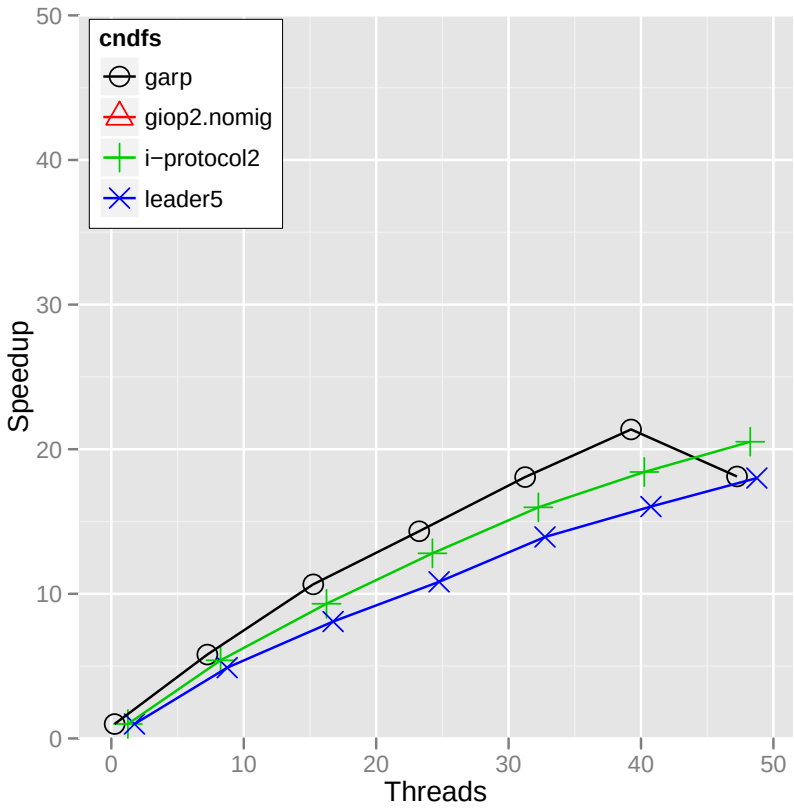


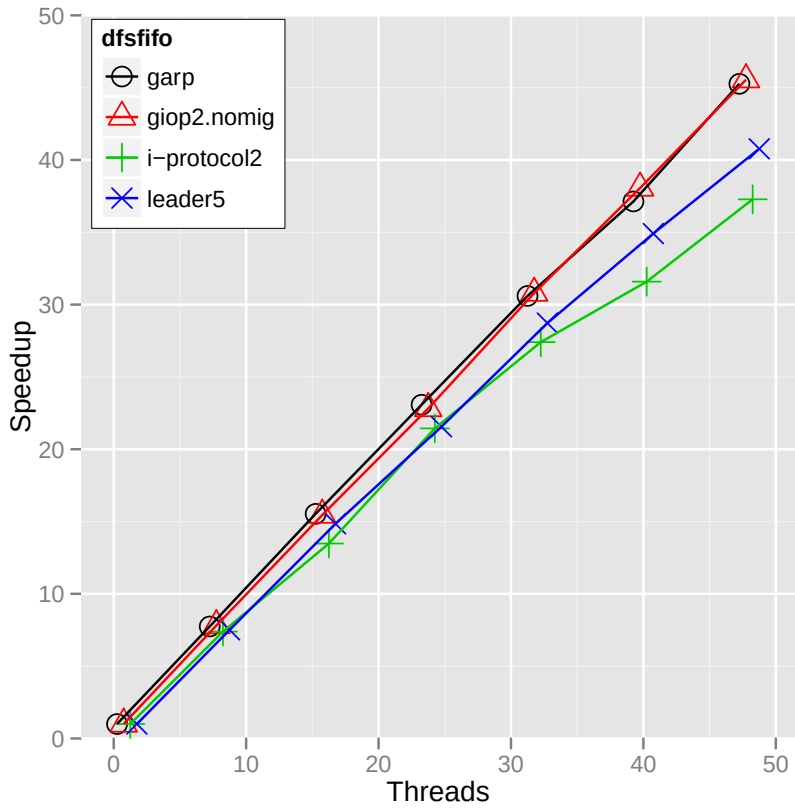
Parallel DFS_{FIFO} Experiments





Parallel DFS_{FIFO} Experiments





No cycle proviso $\Rightarrow$ Parallelism + POR!

Parallel LTL with Partial Order Reduction

DFS_{FIFO} vs OWCTY & Topological Sort Proviso
[Barnat et al 2010]

Parallel LTL with Partial Order Reduction

DFS_{FIFO} vs OWCTY & Topological Sort Proviso [Barnat et al 2010]

N	Alg.	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$	T_1	T_{48}	U	$\lvert\mathcal{R}^{\text{por}}\rvert$	$\lvert\mathcal{T}^{\text{por}}\rvert$	T_1^{por}	T_{48}^{por}	U^{por}
9	cndfs	3.6E7	2.3E8	502.6	12.0	41.8	27.9%	0.1%	211.8	n/a	n/a
9	$\text{pdfs}_{\text{fifo}}$	3.6E7	2.3E8	583.6	14.3	40.8	1.5%	0.0%	12.9	3.6	3.5
9	owcty	3.6E7	2.3E8	498.7	51.9	9.6	12.6%	0.0%	578.4	35.7	16.2
10	cndfs	2.4E8	1.7E9	~~30~~'	90.7	~~30~~'	19.3%	5.4%	1102.7	n/a	n/a
10	$\text{pdfs}_{\text{fifo}}$	2.4E8	1.7E9	~~30~~'	109.3	~~30~~'	0.7%	0.1%	35.0	2.5	14.0
10	owcty	2.4E8	1.7E9	~~30~~'	663.1	~~30~~'	8.7%	2.2%	~~30~~'	156.3	~~30~~'
11	$\text{pdfs}_{\text{fifo}}$	~~30~~'	~~30~~'	~~30~~'	~~30~~'	~~30~~'	5.1E6	7.1E6	109.8	5.3	20.7
11	owcty	~~30~~'	~~30~~'	~~30~~'	~~30~~'	~~30~~'	9.3E7	1.7E8	~~30~~'	1036.5	~~30~~'
12	$\text{pdfs}_{\text{fifo}}$	~~30~~'	~~30~~'	~~30~~'	~~30~~'	~~30~~'	1.6E7	2.2E7	369.1	11.2	33.0
13	$\text{pdfs}_{\text{fifo}}$	~~30~~'	~~30~~'	~~30~~'	~~30~~'	~~30~~'	6.6E7	9.2E7	1640.5	38.1	43.0
14	$\text{pdfs}_{\text{fifo}}$	~~30~~'	~~30~~'	~~30~~'	~~30~~'	~~30~~'	2.0E8	2.9E8	~~30~~'	120.3	~~30~~'
15	$\text{pdfs}_{\text{fifo}}$	~~30~~'	~~30~~'	~~30~~'	~~30~~'	~~30~~'	8.4E8	1.2E9	~~30~~'	527.5	~~30~~'

Conclusions

By specializing for livelocks with DFS_{FIFO}, we

- improved partial-order reduction
- improved the efficiency of parallelization by 100%
- allow efficient combination of POR and parallelism

Conclusions

By specializing for livelocks with DFS_{FIFO}, we

- improved partial-order reduction
- improved the efficiency of parallelization by 100%
- allow efficient combination of POR and parallelism

Future work

- use testers to support more LTL properties [Valmari CAV '93]

Conclusions

By specializing for livelocks with DFS_{FIFO}, we

- improved partial-order reduction
- improved the efficiency of parallelization by 100%
- allow efficient combination of POR and parallelism

Future work

- use testers to support more LTL properties [Valmari CAV '93]

Other mentioned works

1. Boosting Multi-Core Reachability Performance with Shared Hash Tables – FMCAD'10
2. Parallel Recursive State Compression for Free – SPIN'11
3. Multi-Core Nested Depth-First Search – ATVA'11
4. Improved Multi-Core Nested Depth-First Search – ATVA'12
5. SpinS: Extending LTSmin with Promela through SpinJa – PDMC'12