An Improved Construction of Petri Net Unfoldings

César Rodríguez   Stefan Schwoon

LSV, ENS Cachan & CNRS, INRIA Saclay, France

FSFMA 2013, Singapore, July 15, 2013

Reachability Checking in Petri Nets

- Reachability in safe PNs
- Faces state-explosion due to concurrency
- Partial-order semantics
  - Initially developed in the area of semantics
  - Unfolding algorithm by McMillan [McM92]
  - Finite, complete unfolding prefixes

Reachability Checking in Petri Nets

- Reachability in safe PNs
- Faces state-explosion due to concurrency
- Partial-order semantics
  - Initially developed in the area of semantics
  - Unfolding algorithm by McMillan [McM92]
  - Finite, complete unfolding prefixes

Our contribution: improvement in the unfolding algorithm

- Targets the computationally most expensive step
- General: can be integrated into several unfolding approaches
- Preliminary implementation

Petri Nets

Definition

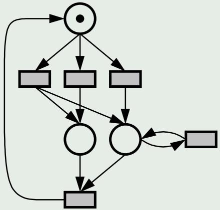
A Petri net is a tuple $N = \langle P, T, F, m_0 \rangle$

- P: finite set of places
- T: finite set of transitions
- $F \subseteq P \times T \cup T \times P$: flow relation
- $m_0 \subseteq P$: initial marking

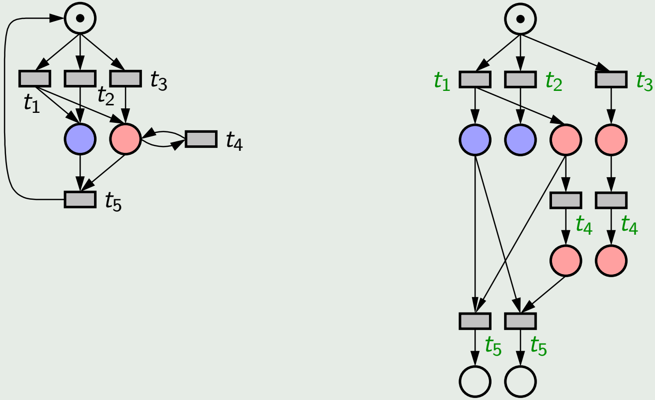
Notation

${}^\bullet x$ for preset, $x^\bullet$ for postset

We only consider 1-safe nets



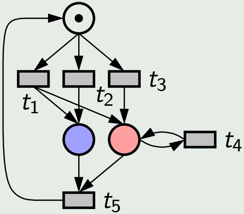
Petri Net Unfoldings (i) — Example



Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Events and conditions
- Labelling is a homomorphism
- Infinite in general

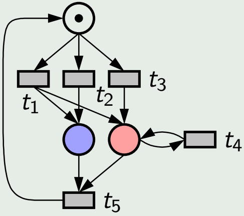
Petri Net Unfoldings (i) — Example

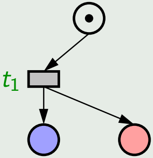


Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Events and conditions
- Labelling is a homomorphism
- Infinite in general

Petri Net Unfoldings (i) — Example

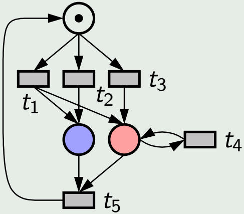


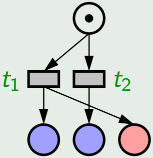


Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Events and conditions
- Labelling is a homomorphism
- Infinite in general

Petri Net Unfoldings (i) — Example

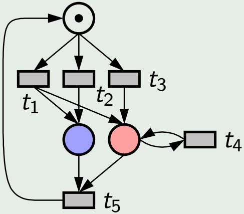


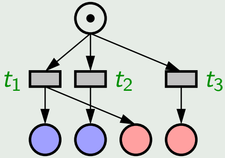


Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Events and conditions
- Labelling is a homomorphism
- Infinite in general

Petri Net Unfoldings (i) — Example

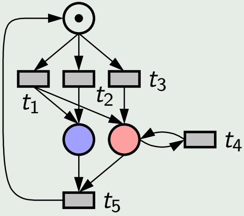


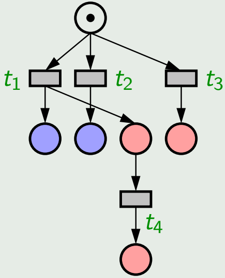


Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Events and conditions
- Labelling is a homomorphism
- Infinite in general

Petri Net Unfoldings (i) — Example

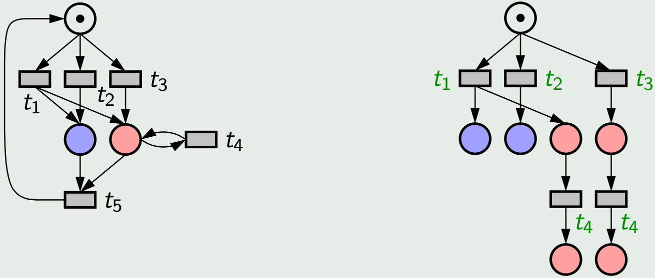




Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Events and conditions
- Labelling is a homomorphism
- Infinite in general

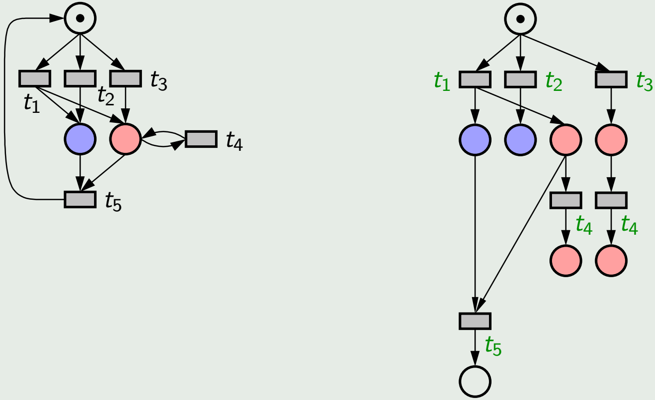
Petri Net Unfoldings (i) — Example



Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Events and conditions
- Labelling is a homomorphism
- Infinite in general

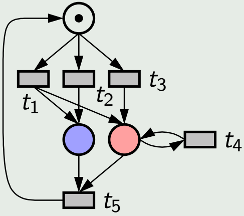
Petri Net Unfoldings (i) — Example

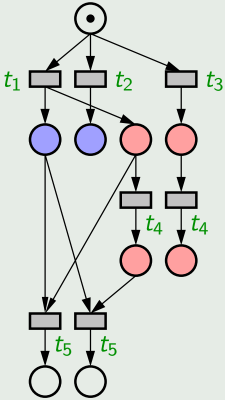


Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Events and conditions
- Labelling is a homomorphism
- Infinite in general

Petri Net Unfoldings (i) — Example





Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Events and conditions
- Labelling is a homomorphism
- Infinite in general

Petri Net Unfoldings (ii) — Complete Prefixes

- $\mathcal{U}_N$ is the result of unfolding 'as much as possible'
- If you stop: finite unfolding prefix $\mathcal{P}_N$

Petri Net Unfoldings (ii) — Complete Prefixes

- $\mathcal{U}_N$ is the result of unfolding 'as much as possible'
- If you stop: finite unfolding prefix $\mathcal{P}_N$

Definition

Prefix $\mathcal{P}_N$ is marking-complete if:
for all marking m reachable in N, there is marking $\tilde{m}$ reachable in $\mathcal{P}_N$ with

$$h(\tilde{m}) = m.$$

Petri Net Unfoldings (ii) — Complete Prefixes

- $\mathcal{U}_N$ is the result of unfolding 'as much as possible'
- If you stop: finite unfolding prefix $\mathcal{P}_N$

Definition

Prefix $\mathcal{P}_N$ is marking-complete if:
for all marking m reachable in N, there is marking $\tilde{m}$ reachable in $\mathcal{P}_N$ with

$$h(\tilde{m}) = m.$$

- McMillan's algorithm constructs $\mathcal{P}_N$ stopping at the cutoff events
  - Improved by Esparza et al. [ERV02]
  - Tools: Mole, Punf

Petri Net Unfoldings (ii) — Complete Prefixes

- $\mathcal{U}_N$ is the result of unfolding 'as much as possible'
- If you stop: finite unfolding prefix $\mathcal{P}_N$

Definition

Prefix $\mathcal{P}_N$ is marking-complete if:
for all marking m reachable in N, there is marking $\tilde{m}$ reachable in $\mathcal{P}_N$ with

$$h(\tilde{m}) = m.$$

- McMillan's algorithm constructs $\mathcal{P}_N$ stopping at the cutoff events
  - Improved by Esparza et al. [ERV02]
  - Tools: MOLE, PUNF

Reachability in N is

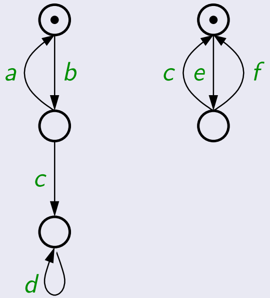
- PSPACE-complete on N
- NP-complete on $\mathcal{P}_N$ (upper bound: $\mathcal{P}_N$ is acyclic!)

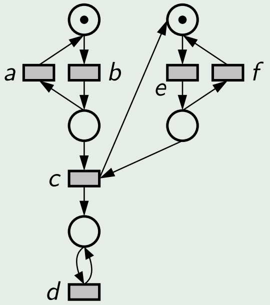
Unfolding Other Models of Concurrency

Unfoldings applicable to other models of concurrency:

- Process algebras
- High-level nets
- Unbounded nets
- Nets with read arcs
- Time Petri nets
- Communicating automata
- Concurrent boolean programs
- ...

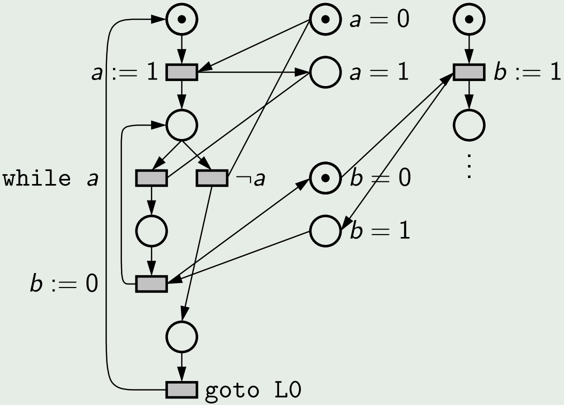
Communicating Automata





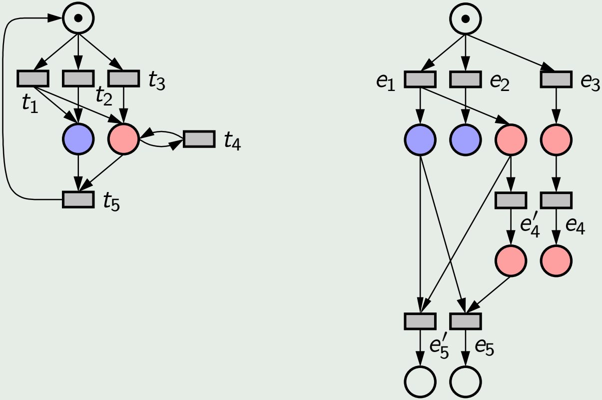
Concurrent Boolean Programs

```
L0:  a := 1;                          L1:  b := 1;
     while (a) b := 0;                     while (b) a := 0;
     goto L0;                              goto L1;
```



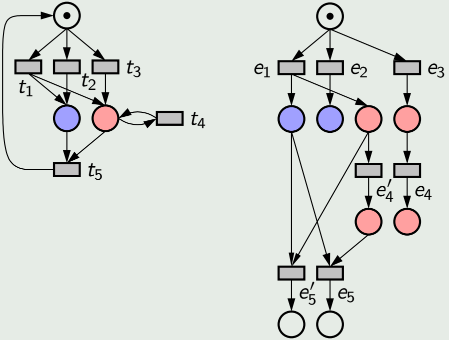
Causality and Conflict

The structure of an unfolding induces three relations over its events:



Causality:	$e < e'$	iff	e' occurs $\Rightarrow$ e occurs before
Conflict:	$e \# e'$	iff	e and e' never occur in the same run
Concurrency:	$e \parallel e'$	iff	not $e < e'$ and not $e' < e$ and not $e \# e'$

Configurations



Configurations:

$\{e_3, e_4\}$

$\{e_1, e'_4, e_5\}$

Not a configuration:

$\{e_4\}$ not causally closed

$\{e_2, e_3, e_4\}$ $e_2 \# e_3$

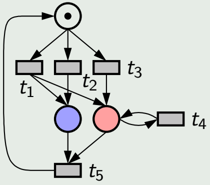
Configurations

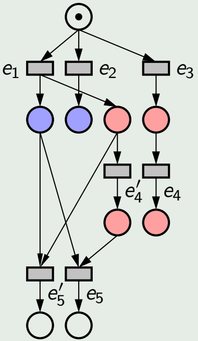
Definition

A set of events $\mathcal{C}$ is a configuration iff:

1. $e \in \mathcal{C} \wedge e' < e \Rightarrow e' \in \mathcal{C}$ — causally closed
2. $\neg e \# e'$ for all $e, e' \in \mathcal{C}$ — conflict free

Intuition: $\mathcal{C}$ configuration iff all its events can be arranged to form a run.





Configurations:
$\{e_3, e_4\}$
$\{e_1, e_4', e_5\}$

Not a configuration:
$\{e_4\}$ — not causally closed
$\{e_2, e_3, e_4\}$ — $e_2 \# e_3$

Computing Prefix Extensions

Main computational problem:

Prefix Extensions

Given $\mathcal{P}_N$ and t, can we extend $\mathcal{P}_N$ with e where $h(e) = t$: NP-complete

Computing Prefix Extensions

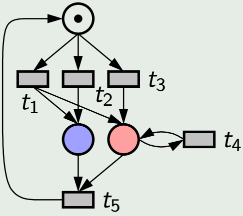
Main computational problem:

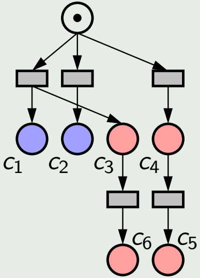
Prefix Extensions

Given $\mathcal{P}_N$ and t, can we extend $\mathcal{P}_N$ with e where $h(e) = t$: NP-complete

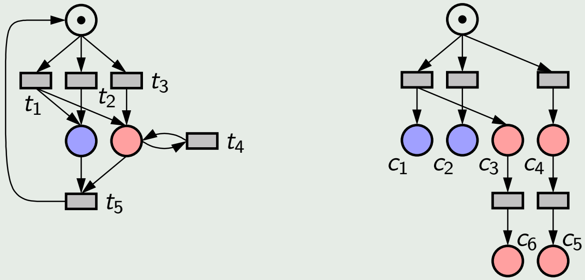
- Enumerate sets of conditions S s.t. $h(S) = {}^{\bullet}t \cup \underline{t}$ (exponential)
- If S is coverable, return YES; otherwise continue (linear)

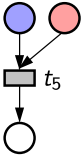
Exploring the "Comb"



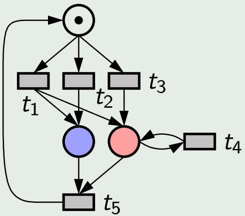


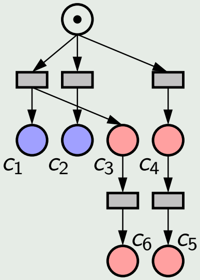
Exploring the "Comb"

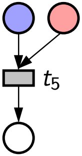


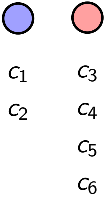


Exploring the “Comb”

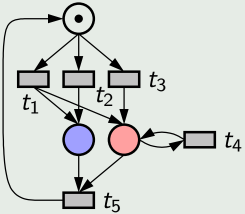


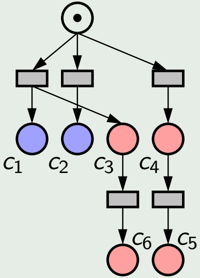


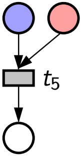


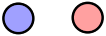


Exploring the "Comb"





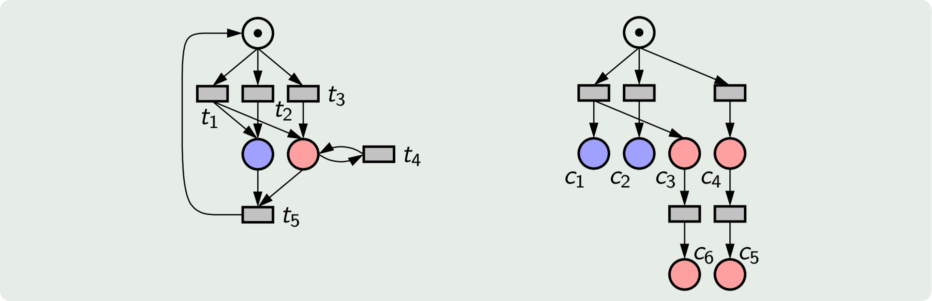


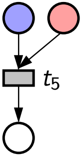


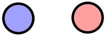
c_1	c_3
c_2	c_4
	c_5
	c_6

1 3

Exploring the "Comb"



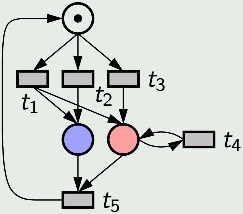


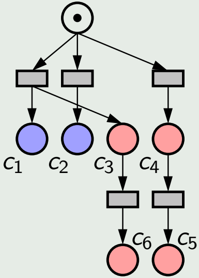


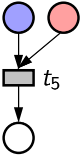
c_1	c_3
c_2	c_4
	c_5
	c_6

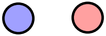
1 3 Yes

Exploring the “Comb”







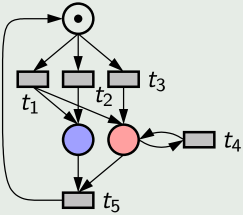


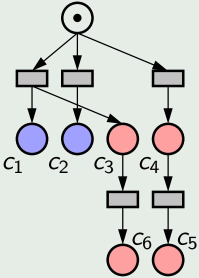
c_1	c_3
c_2	c_4
	c_5
	c_6

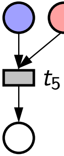
1 3 Yes

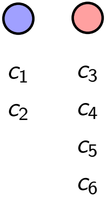
1 4

Exploring the “Comb”



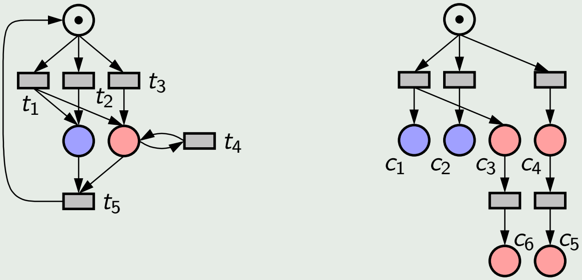


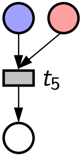




1 3	Yes
1 4	No

Exploring the "Comb"

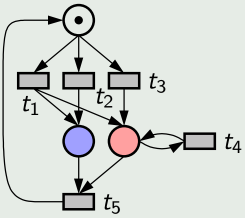


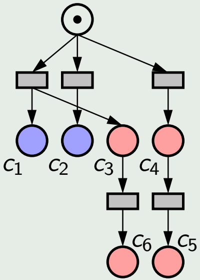


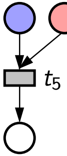
c_1	c_3
c_2	c_4
	c_5
	c_6

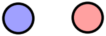
1	3	Yes
1	4	No
1	5	No
1	6	Yes

Exploring the "Comb"





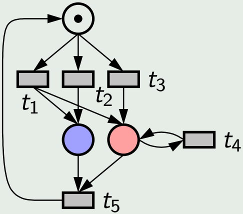


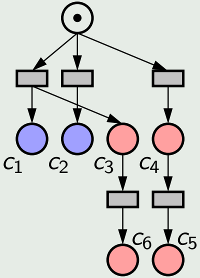


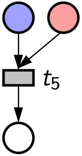
c_1	c_3
c_2	c_4
	c_5
	c_6

1 3	Yes
1 4	No
1 5	No
1 6	Yes
2 3	No
2 4	No
2 5	No
2 6	No

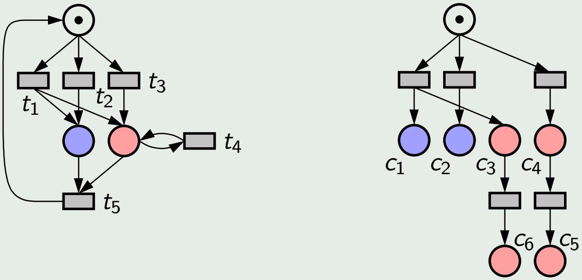
Exploiting Causality

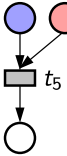


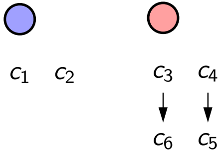




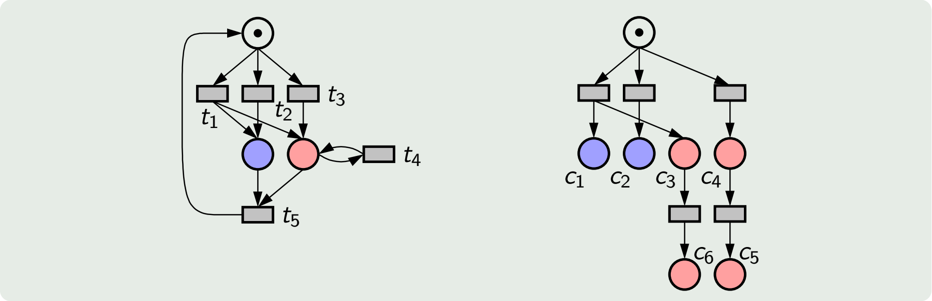
Exploiting Causality

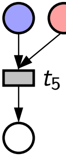


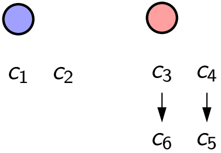




Exploiting Causality







1 3 Yes
1 4 No
~~1 5 No~~
1 6 Yes

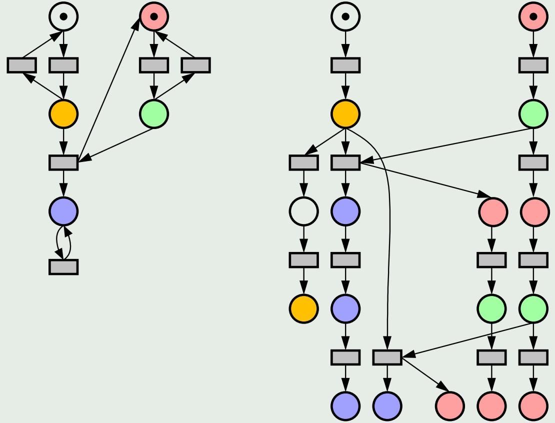
2 3 No
2 4 No
~~2 5 No~~
~~2 6 No~~

p-forest

Definition

The p-forest is the partial order $(h^{-1}(p), <)$, i.e.,

1. Conditions labelled by p
3. Using causality as order

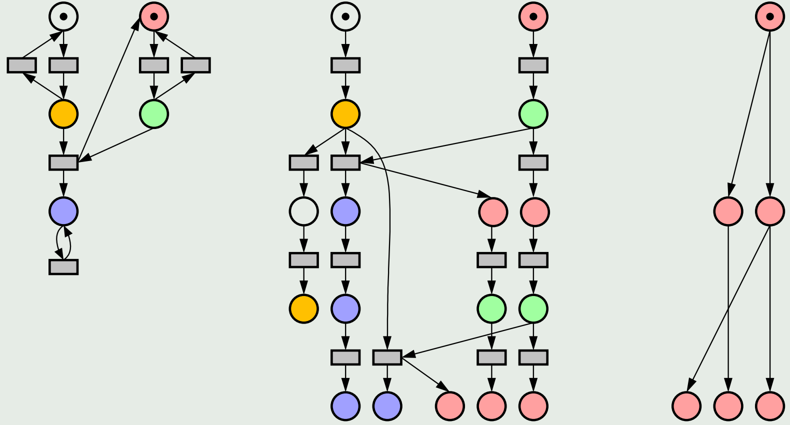


p-forest

Definition

The p-forest is the partial order $(h^{-1}(p), <)$, i.e.,

1. Conditions labelled by p
3. Using causality as order

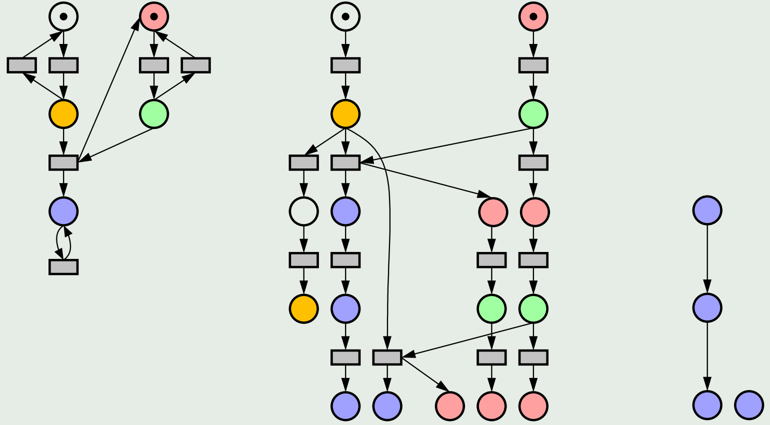


p-forest

Definition

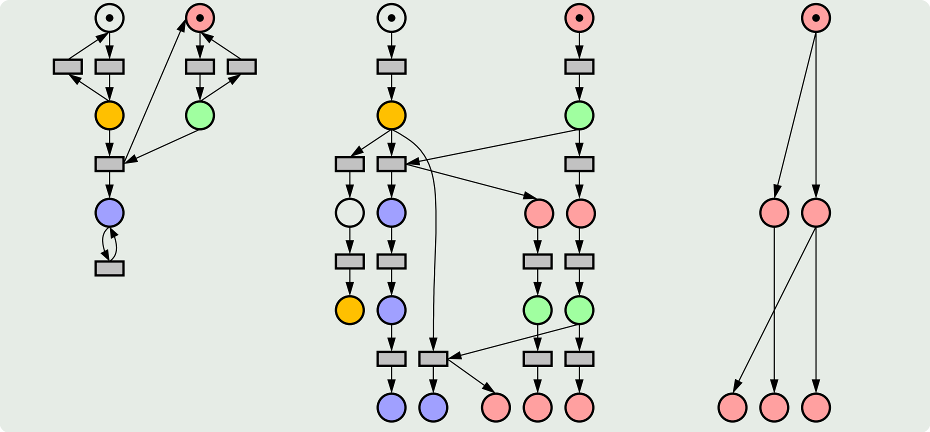
The p-forest is the partial order $(h^{-1}(p), <)$, i.e.,

1. Conditions labelled by p
3. Using causality as order



Computing *p*-forests

- Incremental algorithm
- Cost $\mathcal{O}(1)$ for each condition c appended to the forest
  - Once $[c]$ has been marked



Experiments

Net	Unfolding		New Alg.	Punf	Mole
Name	Events	Cond.	Time	Time (r)	Time (r)
Byz	14724	42276	0.73	11.48	2.66
Q(1)	7469	20969	0.21	6.81	2.14
Elev(4)	16935	32354	0.50	5.06	0.24
Dme(6)	1830	6451	0.04	4.50	3.50
Dme(7)	2737	9542	0.08	4.88	3.88
Dme(9)	5337	18316	0.22	6.64	4.95
Dme(11)	9185	31186	0.53	8.13	5.92
Key(3)	6968	13941	0.23	2.52	0.30
Key(4)	67954	135914	15.94	2.34	0.06
Furn(3)	25394	58897	0.69	3.48	1.01
Furn(4)	146606	342140	25.75	3.02	0.67
Mmgt(3)	5841	11575	0.15	1.93	0.20
Mmgt(4)	46902	92940	9.95	1.68	0.06

Summary

- Algorithmic improvement for constructing net unfoldings
- Preliminary implementation
- Promising results: beats PUNF in almost all examples

Future work

- Generalize the approach to contextual unfoldings

Summary

- Algorithmic improvement for constructing net unfoldings
- Preliminary implementation
- Promising results: beats PUNF in almost all examples

Future work

- Generalize the approach to contextual unfoldings

Thank you for your attention

Javier Esparza, Stefan Römer, and Walter Vogler.
An improvement of McMillan's unfolding algorithm.
Formal Methods in System Design, 20:285–310, 2002.

Kenneth L. McMillan.
Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits.
In Proc. CAV, LNCS 663, pages 164–177, 1992.