Verification Based on Unfoldings of Petri Nets with Read Arcs

César Rodríguez

PhD Thesis defense
Ecole Normale Supérieure de Cachan

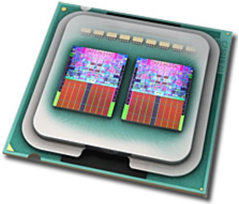
LSV · EDSP

December 12, 2013

Concurrent and Distributed Systems

- System are today increasingly complex and distributed
- Concurrent systems are difficult to reason about

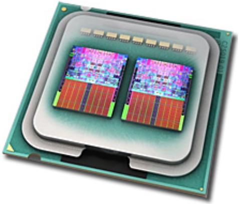
- Avionics
- Traffic control systems
- Multithreading software
- Communication systems
- . . .



Concurrent and Distributed Systems

- System are today increasingly complex and distributed
- Concurrent systems are difficult to reason about

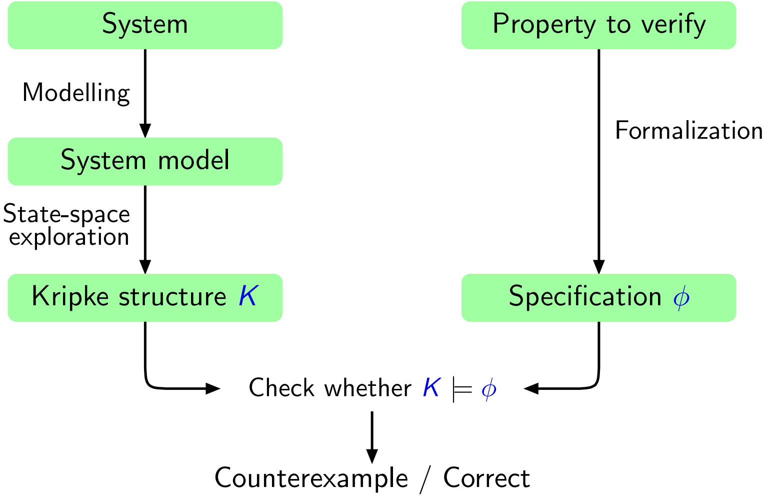
- Avionics
- Traffic control systems
- Multithreading software
- Communication systems
- ...



Ensuring Reliability

- Formal verification: model checking, theorem proving
- Dynamic methods: fault tolerance, runtime verification, fault diagnosis

Model Checking



State-space Explosion

- Interleaving of concurrent actions increase size of state-space
- But many interleavings are uninteresting for target property

```
x := 1                         y := 1
if (x)  y := 0                 if (y)  x := 0
assert (x)
```

State-space Explosion

- Interleaving of concurrent actions increase size of state-space
- But many interleavings are uninteresting for target property

```
x := 1                           y := 1
if (x)  y := 0                   if (y)  x := 0
assert (x)
```

	y := 1
x := 1	
if (x)	
y := 0	
	if (y)
assert (x) ✓	

State-space Explosion

- Interleaving of concurrent actions increase size of state-space
- But many interleavings are uninteresting for target property

```
x := 1                               y := 1
if (x)  y := 0                       if (y)  x := 0
assert (x)
```

`x := 1`	
	`y := 1`
`if (x)`	
`y := 0`	
	`if (y)`
`assert (x)` ✓	

State-space Explosion

- Interleaving of concurrent actions increase size of state-space
- But many interleavings are uninteresting for target property

```
x := 1                              y := 1
if (x)  y := 0                      if (y)  x := 0
assert (x)
```

```
x := 1
------------------------------------------------
                              y := 1
------------------------------------------------
                              if (y)
------------------------------------------------
                              x := 0
------------------------------------------------
if (x)
------------------------------------------------
assert (x) ✗
```

Verification Based on Partial Orders

Concurrent system

- Sequential semantics
- Partial-order semantics

Unfolding semantics of Petri nets

- Compact in presence of concurrency
- But suffer from other sources of explosion such as:

Concurrent read access

Sequences of choices

Verification Based on Partial Orders

Concurrent system

- Sequential semantics
- Partial-order semantics

Unfolding semantics of Petri nets

- Compact in presence of concurrency
- But suffer from other sources of explosion such as:

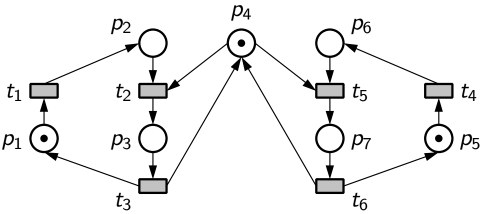
Concurrent read access

Sequences of choices

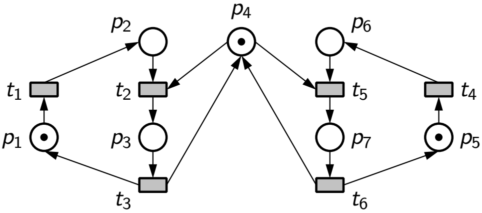
In this thesis

- Study unfoldings of Petri nets with read arcs
- Use them in model checking
- Improve conventional unfoldings for fault diagnosis

Petri Nets — Sequential Semantics



Petri Nets — Sequential Semantics

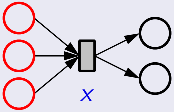


- Preset ${}^\bullet x$ and postset $x^\bullet$

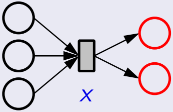
Presets and Postsets

The preset and postset of a transition x (similarly for places) are:

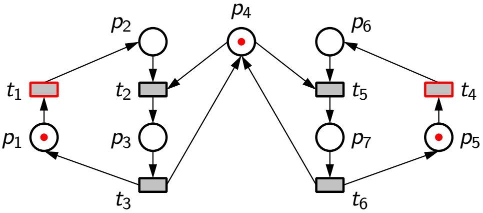
Preset ${}^\bullet x$ of x:



Postset $x^\bullet$ of x:



Petri Nets — Sequential Semantics



- Preset $^\bullet x$ and postset $x^\bullet$
- Firing sequence or run $t_1\, t_2\, t_3 \ldots \in T^* \cup T^\omega$

Run

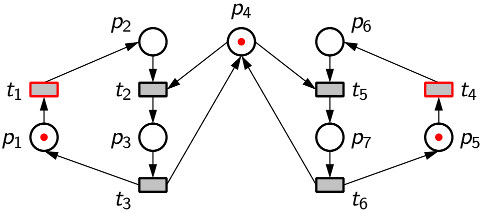
A run, or firing sequence is any sequence of transitions

$$t_1\, t_2\, t_3 \ldots \in T^* \cup T^\omega$$

such that

$$\{p_1, p_4, p_5\}$$

Petri Nets — Sequential Semantics



- Preset $^\bullet x$ and postset $x^\bullet$
- Firing sequence or run $t_1 t_2 t_3 \ldots \in T^* \cup T^\omega$

Run

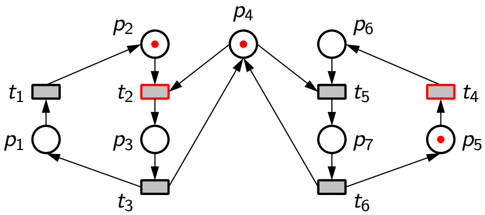
A run, or firing sequence is any sequence of transitions

$$t_1 t_2 t_3 \ldots \in T^* \cup T^\omega$$

such that

$$\{p_1, p_4, p_5\} \quad \xrightarrow{t_1}$$

Petri Nets — Sequential Semantics



- Preset ${}^\bullet x$ and postset $x^\bullet$
- Firing sequence or run $t_1 t_2 t_3 \ldots \in T^* \cup T^\omega$

Run

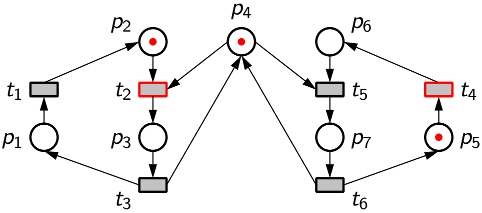
A run, or firing sequence is any sequence of transitions

$$t_1 t_2 t_3 \ldots \in T^* \cup T^\omega$$

such that

$$\{p_1, p_4, p_5\} \xrightarrow{t_1} \{p_2, p_4, p_5\}$$

Petri Nets — Sequential Semantics



- Preset $^\bullet x$ and postset $x^\bullet$
- Firing sequence or run $t_1 t_2 t_3 \ldots \in T^* \cup T^\omega$

Run

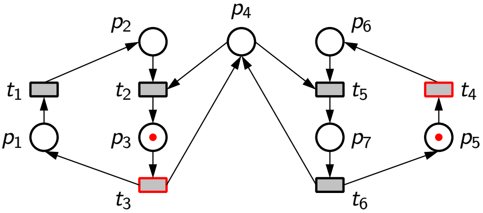
A run, or firing sequence is any sequence of transitions

$$t_1 t_2 t_3 \ldots \in T^* \cup T^\omega$$

such that

$$\{p_1, p_4, p_5\} \xrightarrow{t_1} \{p_2, p_4, p_5\} \xrightarrow{t_2}$$

Petri Nets — Sequential Semantics



- Preset ${}^\bullet x$ and postset $x^\bullet$
- Firing sequence or run $t_1\, t_2\, t_3 \ldots \in T^* \cup T^\omega$

Run

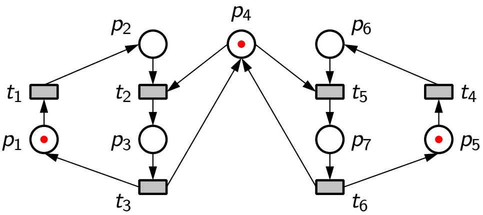
A run, or firing sequence is any sequence of transitions

$$t_1\, t_2\, t_3 \ldots \in T^* \cup T^\omega$$

such that

$$\{p_1, p_4, p_5\} \xrightarrow{t_1} \{p_2, p_4, p_5\} \xrightarrow{t_2} \{p_3, p_5\}$$

Petri Nets — Sequential Semantics



- Preset $^\bullet x$ and postset $x^\bullet$
- Firing sequence or run $t_1 t_2 t_3 \ldots \in T^* \cup T^\omega$

Run

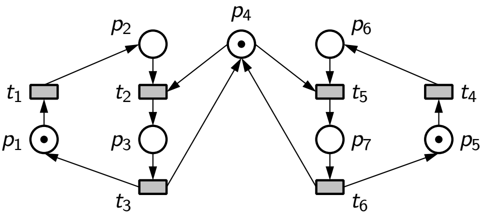
A run, or firing sequence is any sequence of transitions

$$t_1 t_2 t_3 \ldots \in T^* \cup T^\omega$$

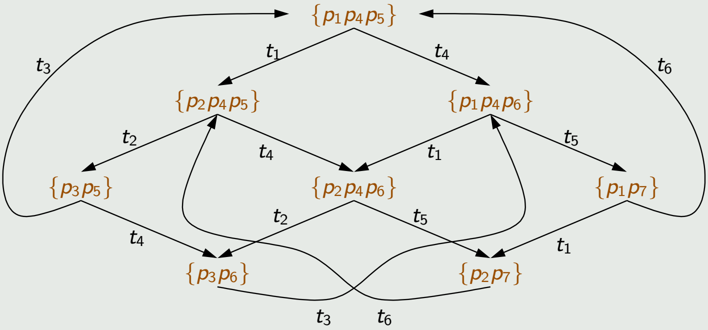
such that

$$\{p_1, p_4, p_5\} \xrightarrow{t_1} \{p_2, p_4, p_5\} \xrightarrow{t_2} \{p_3, p_5\} \xrightarrow{t_3} \ldots$$

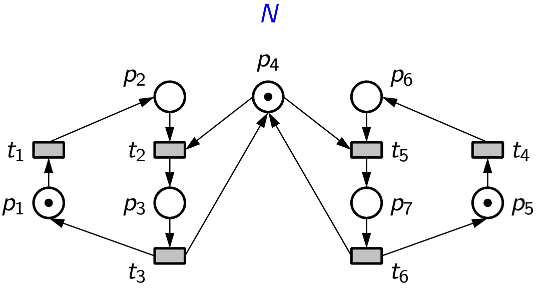
Petri Nets — Sequential Semantics



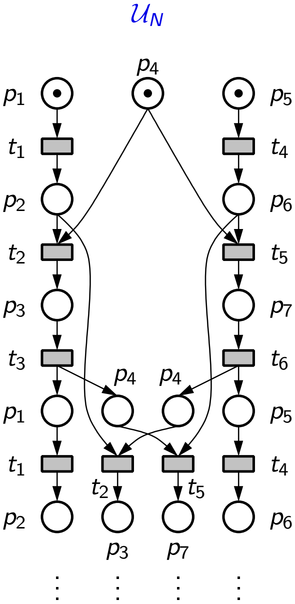
- Preset ${}^\bullet x$ and postset $x^\bullet$
- Firing sequence or run $t_1 t_2 t_3 \ldots \in T^* \cup T^\omega$
- Reachability graph



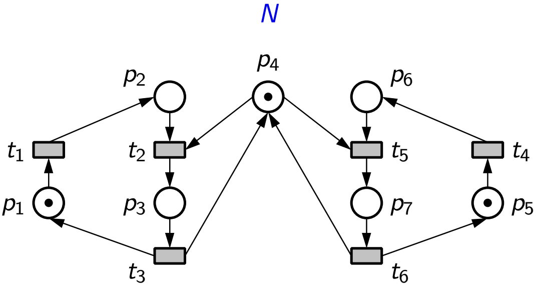
Petri Net — Unfolding Semantics

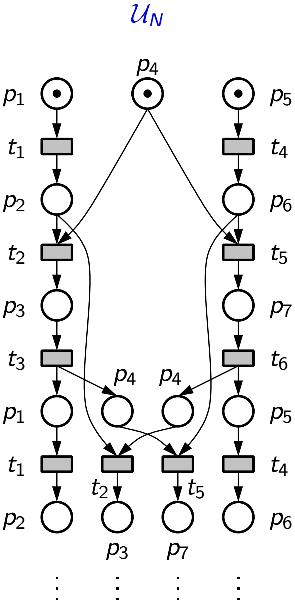


- Unfolding semantics of N is another net $\mathcal{U}_N$
- $\mathcal{U}_N$ is acyclic and labelled
- Transitions are events and places conditions
- Labelling is a homomorphism



Petri Net — Unfolding Semantics

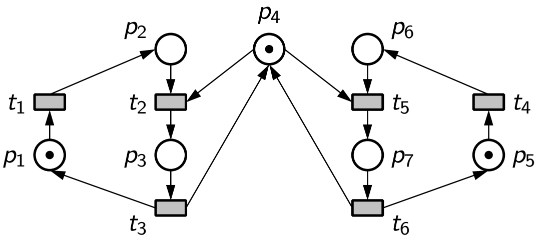




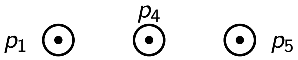
- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^{\bullet}t$
  - Add copy of t, with preset X, and copy of $t^{\bullet}$
- Until no such t and X can be found

Petri Net — Unfolding Semantics

N

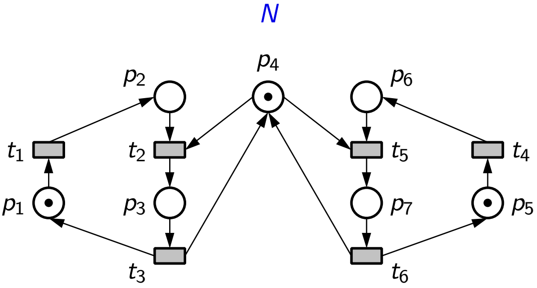


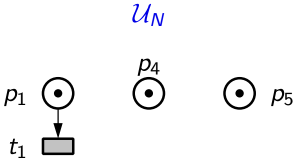
$\mathcal{U}_N$



- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^\bullet t$
  - Add copy of t, with preset X, and copy of $t^\bullet$
- Until no such t and X can be found

Petri Net — Unfolding Semantics

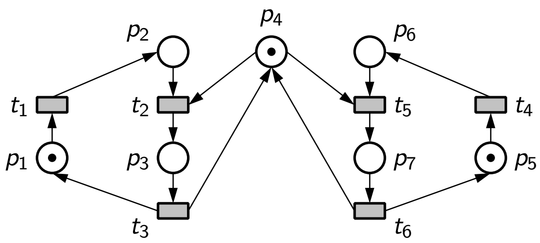




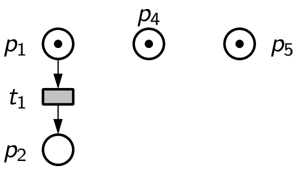
- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^\bullet t$
  - Add copy of t, with preset X, and copy of $t^\bullet$
- Until no such t and X can be found

Petri Net — Unfolding Semantics

N

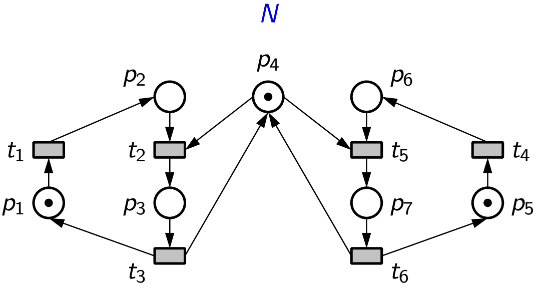


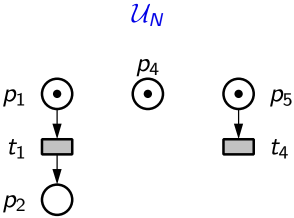
$\mathcal{U}_N$



- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^\bullet t$
  - Add copy of t, with preset X, and copy of $t^\bullet$
- Until no such t and X can be found

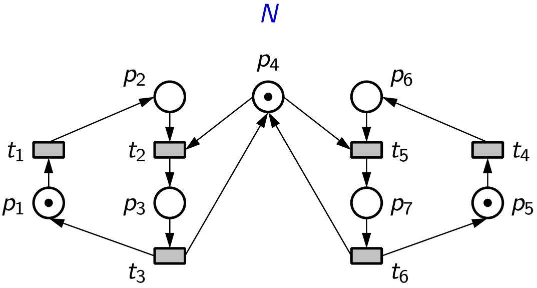
Petri Net — Unfolding Semantics

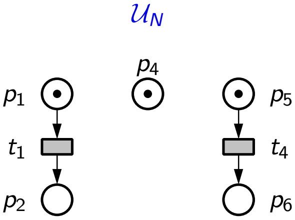




- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^\bullet t$
  - Add copy of t, with preset X, and copy of $t^\bullet$
- Until no such t and X can be found

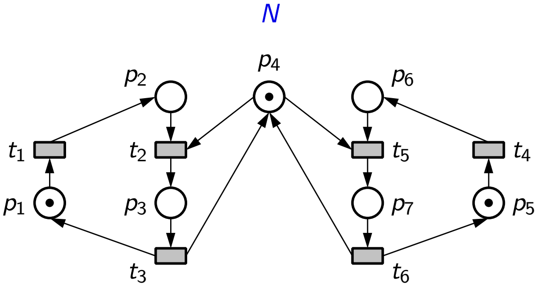
Petri Net — Unfolding Semantics

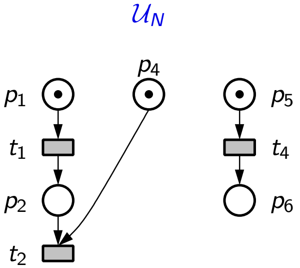




- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^{\bullet}t$
  - Add copy of t, with preset X, and copy of $t^{\bullet}$
- Until no such t and X can be found

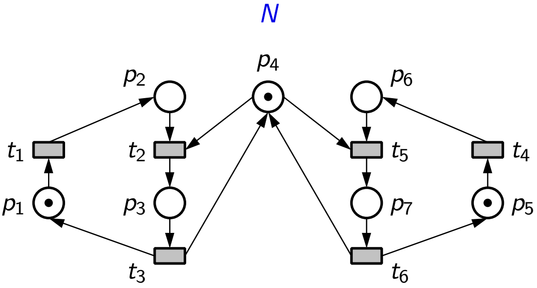
Petri Net — Unfolding Semantics

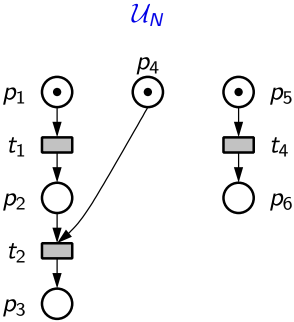




- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^\bullet t$
  - Add copy of t, with preset X, and copy of $t^\bullet$
- Until no such t and X can be found

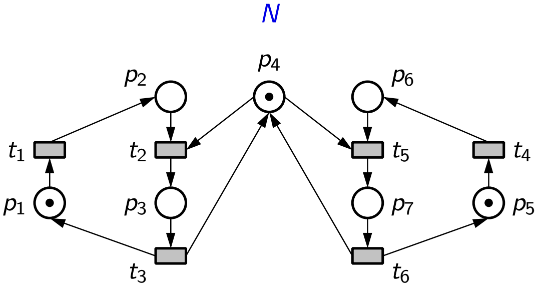
Petri Net — Unfolding Semantics

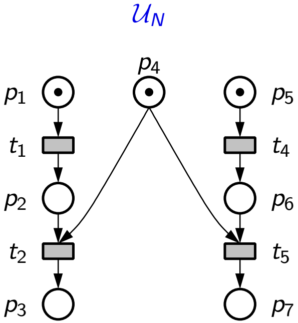




- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^\bullet t$
  - Add copy of t, with preset X, and copy of $t^\bullet$
- Until no such t and X can be found

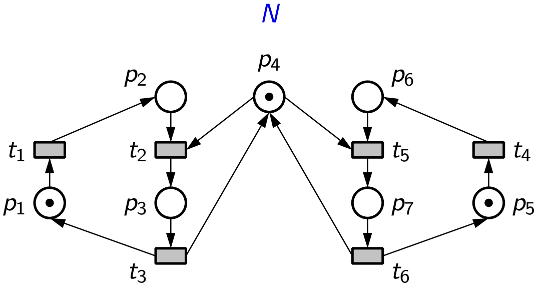
Petri Net — Unfolding Semantics

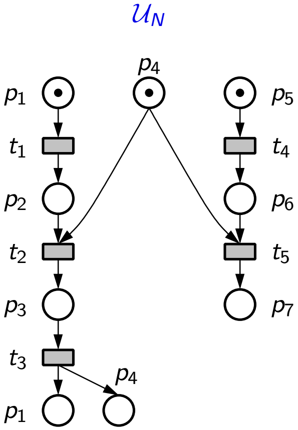




- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^\bullet t$
  - Add copy of t, with preset X, and copy of $t^\bullet$
- Until no such t and X can be found

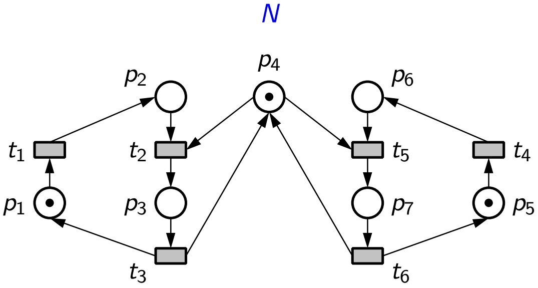
Petri Net — Unfolding Semantics





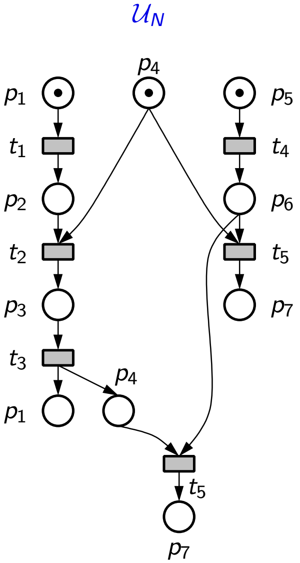
- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^\bullet t$
  - Add copy of t, with preset X, and copy of $t^\bullet$
- Until no such t and X can be found

Petri Net — Unfolding Semantics



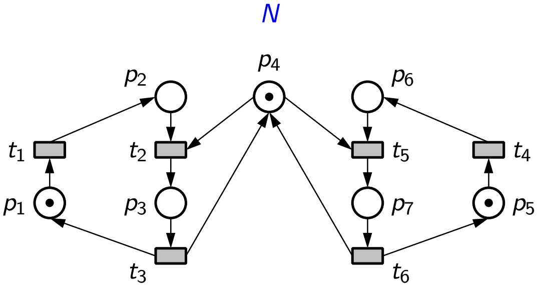
N

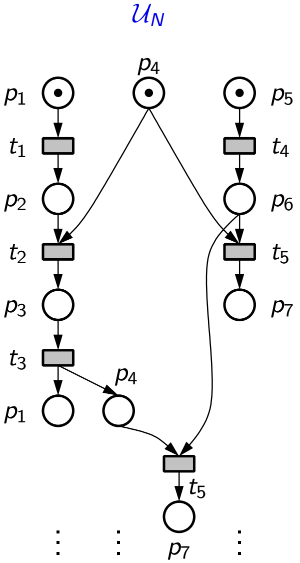
- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^{\bullet}t$
  - Add copy of t, with preset X, and copy of $t^{\bullet}$
- Until no such t and X can be found



$\mathcal{U}_N$

Petri Net — Unfolding Semantics





- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^\bullet t$
  - Add copy of t, with preset X, and copy of $t^\bullet$
- Until no such t and X can be found

Verification with Unfoldings: Finite, Complete Prefixes

- $\mathcal{U}_N$ is the result of unfolding ‘as much as possible’
- Finite unfolding prefix $\mathcal{P}_N$ results if you stop construction

Verification with Unfoldings: Finite, Complete Prefixes

- $\mathcal{U}_N$ is the result of unfolding 'as much as possible'
- Finite unfolding prefix $\mathcal{P}_N$ results if you stop construction

If N has finitely many reachable markings. . .

Verification with Unfoldings: Finite, Complete Prefixes

- $\mathcal{U}_N$ is the result of unfolding 'as much as possible'
- Finite unfolding prefix $\mathcal{P}_N$ results if you stop construction

Definition

Prefix $\mathcal{P}_N$ is marking-complete if:

for all marking m reachable in N, there is marking $\tilde{m}$ reachable in $\mathcal{P}_N$ such that

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

If N has finitely many reachable markings...

Verification with Unfoldings: Finite, Complete Prefixes

- $\mathcal{U}_N$ is the result of unfolding 'as much as possible'
- Finite unfolding prefix $\mathcal{P}_N$ results if you stop construction

Definition

Prefix $\mathcal{P}_N$ is marking-complete if:

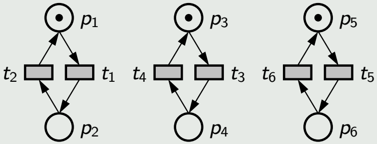
for all marking m reachable in N, there is marking $\tilde{m}$ reachable in $\mathcal{P}_N$ such that

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

If N has finitely many reachable markings. . .

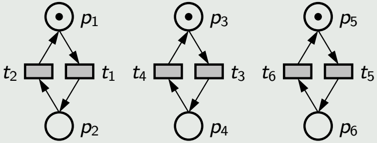
- Some finite and marking-complete $\mathcal{P}_N$ exists
- $\mathcal{P}_N$: symbolic representation of reachability graph
- Reachability of N is:
  - PSPACE-complete in N
  - NP-complete in $\mathcal{P}_N$
  - Linear in reachability graph

Unfoldings Cope with Concurrency

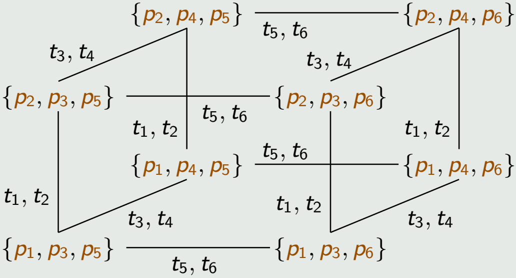


- 2^3 reachable markings

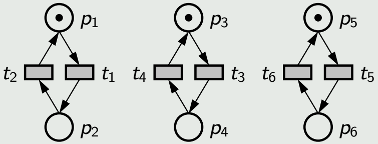
Unfoldings Cope with Concurrency

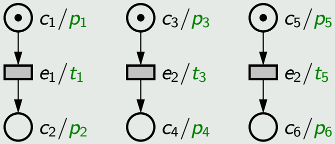


- 2^3 reachable markings
- And 2^n if n processes

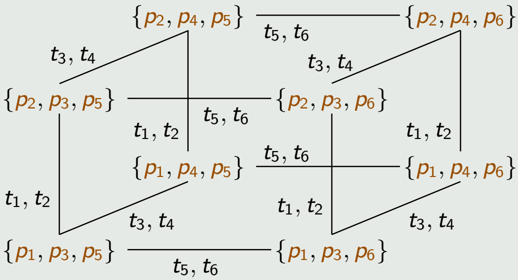


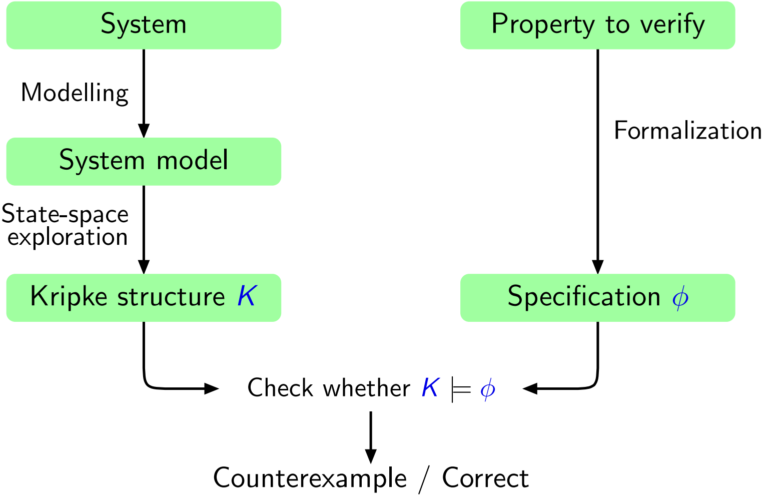
Unfoldings Cope with Concurrency





- 2^3 reachable markings
- And 2^n if n processes
- Unfolding is of linear size





System

Modelling

System model

State-space exploration

Kripke structure K

Property to verify

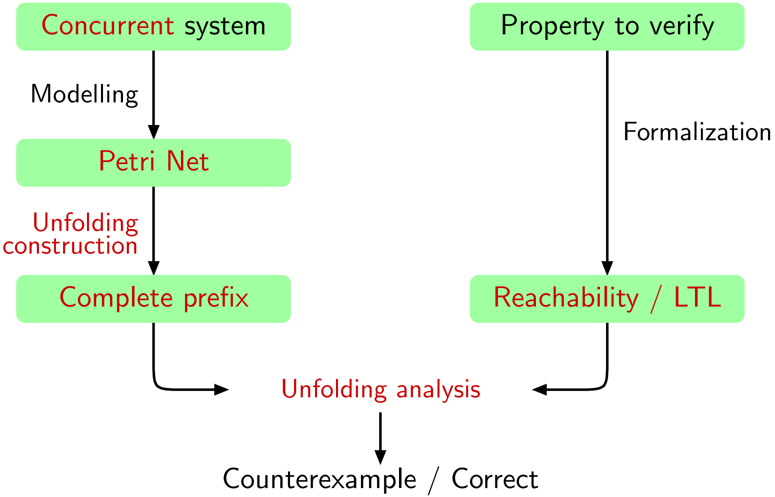
Formalization

Specification ϕ

Check whether $K \models \phi$

Counterexample / Correct

Model Checking with Net Unfoldings



Concurrent system

Modelling

Petri Net

Unfolding construction

Complete prefix

Property to verify

Formalization

Reachability / LTL

Unfolding analysis

Counterexample / Correct

Model Checking with Net Unfoldings

Unfolding construction

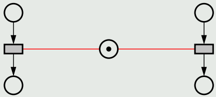
- Initially proposed by Ken McMillan [McMillan 92]
- Size of the prefix reduced [Esparza, Römer, Vogler 96]
- Canonical prefixes [Khomenko, Koutny, Vogler 02]
- Comprehensive account [Esparza, Heljanko 08]

Unfolding analysis

- Reachability and deadlock [McMillan 92], [Melzer, Römer 97], [Heljanko 99], [Khomenko,Koutny 00]
- LTL-X [Esparza, Heljanko 01]

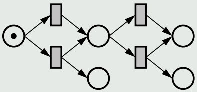
Improving Unfolding-based Verification: Outline

Concurrent read access



- Unfolding construction for nets with read arcs
- SAT-based reachability analysis
- Reduction of size: adequate orders
- Experimental evaluation

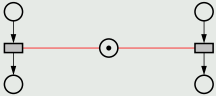
Sequences of choices



- Integration with merged processes
- SAT-based reachability analysis
- Characterization of mp-configurations
- Experimental evaluation

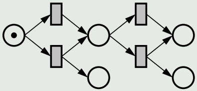
Improving Unfolding-based Verification: Outline

Concurrent read access



- Unfolding construction for nets with read arcs
- SAT-based reachability analysis
- Reduction of size: adequate orders
- Experimental evaluation

Sequences of choices



- Integration with merged processes
- SAT-based reachability analysis
- Characterization of mp-configurations
- Experimental evaluation

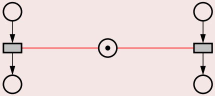
Fault diagnosis

(for conventional Petri nets)

- Generalization to partially-ordered observations
- Integration of fairness assumptions

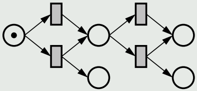
Improving Unfolding-based Verification: Outline

Concurrent read access



- Unfolding construction for nets with read arcs
- SAT-based reachability analysis
- Reduction of size: adequate orders
- Experimental evaluation

Sequences of choices



- Integration with merged processes
- SAT-based reachability analysis
- Characterization of mp-configurations
- Experimental evaluation

Fault diagnosis

(for conventional Petri nets)

- Generalization to partially-ordered observations
- Integration of fairness assumptions

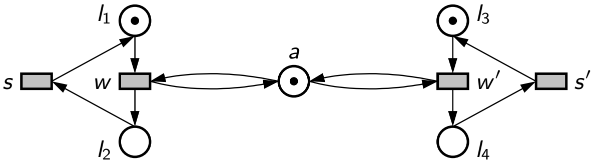
Concurrent Read Access and Unfoldings

Thread 1

```
l1:  while (a)
l2:    work;
```

Thread 2

```
l3:  while (a)
l4:    work;
```



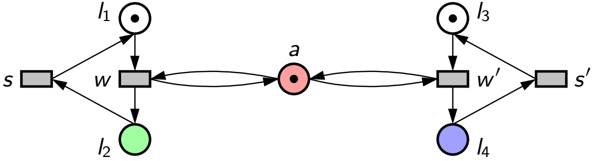
Concurrent Read Access and Unfoldings

Thread 1

```
l1:  while (a)
l2:    work;
```

Thread 2

```
l3:  while (a)
l4:    work;
```



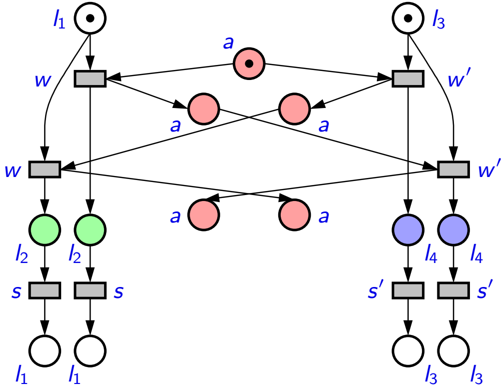
Concurrent Read Access and Unfoldings

Thread 1

```
l1:  while (a)
l2:    work;
```

Thread 2

```
l3:  while (a)
l4:    work;
```



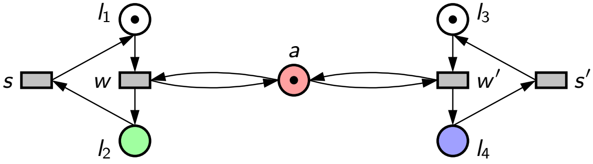
Concurrent Read Access and Unfoldings

Thread 1

```
l1:  while (a)
l2:    work;
```

Thread 2

```
l3:  while (a)
l4:    work;
```



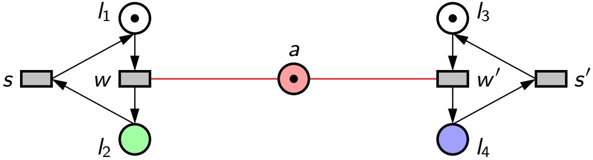
Concurrent Read Access and Unfoldings

Thread 1

```
l1:  while (a)
l2:    work;
```

Thread 2

```
l3:  while (a)
l4:    work;
```



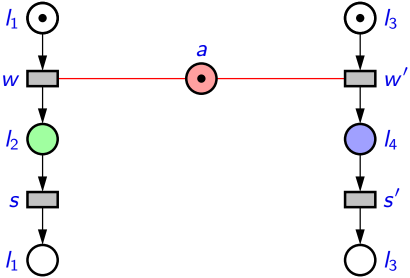
Concurrent Read Access and Unfoldings

Thread 1

```
l1:  while (a)
l2:    work;
```

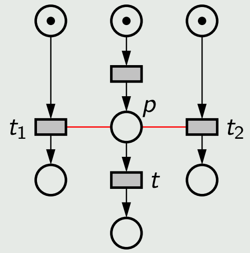
Thread 2

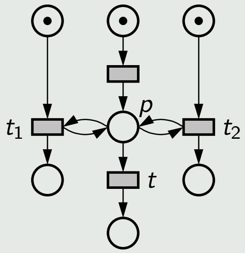
```
l3:  while (a)
l4:    work;
```



Contextual Nets (c-nets)

- Contextual nets: Petri nets + read arcs



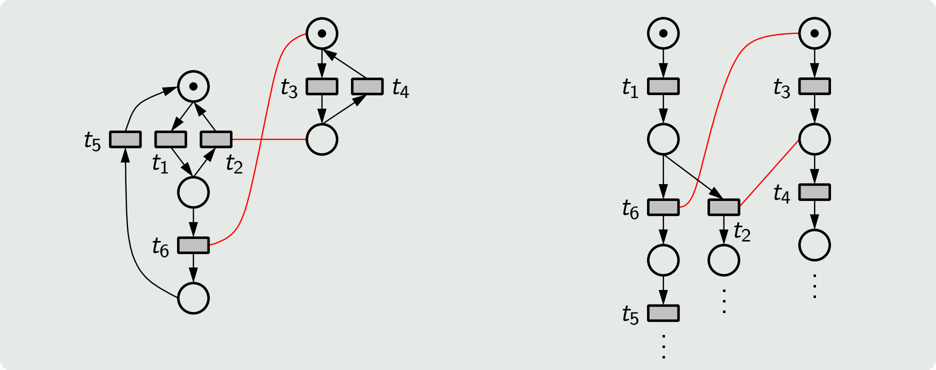


- Transitions (and places) have context: $\underline{t_1} = \{p\}$, $\underline{p} = \{t_1, t_2\}$
- Assumptions: interleaving semantics and finite-state contextual net

[Montanari, Rossi 95]

Contextual Unfoldings

- Contextual unfoldings can be more compact but have richer structure

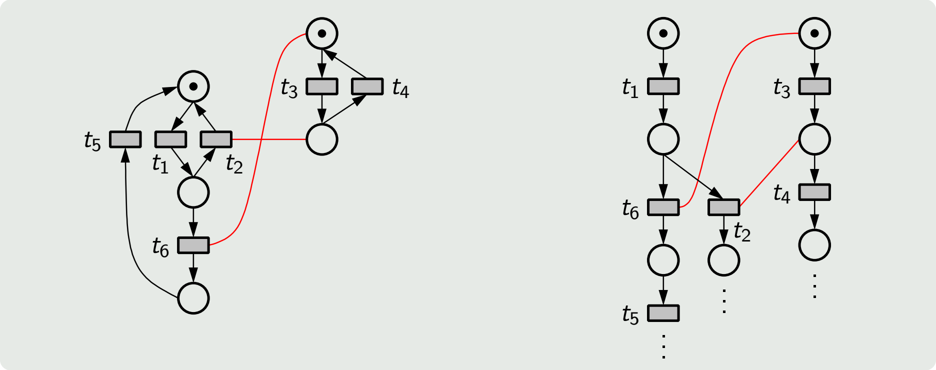


Causality: $e < e'$ iff e' occurs $\Rightarrow$ e occurs before

[Baldan, Corradini, Montanari 98] [Vogler, Semenov, Yakovlev 98]

Contextual Unfoldings

- Contextual unfoldings can be more compact but have richer structure



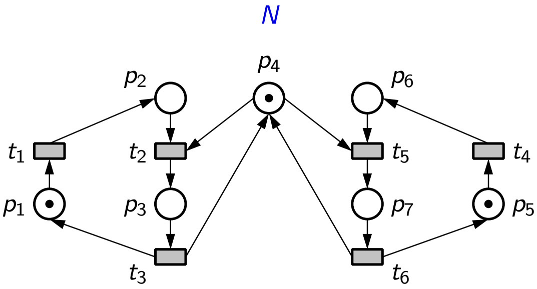
Causality: $e < e'$ iff e' occurs ⇒ e occurs before

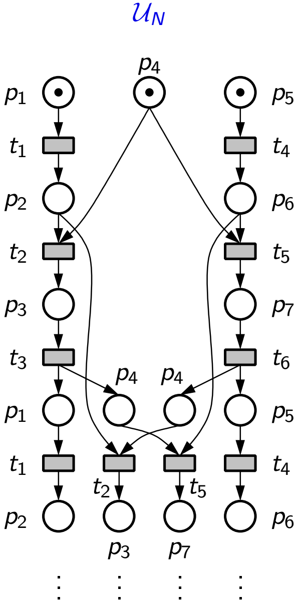
Asymmetric conflict: $e \nearrow e'$ iff e and e' occur ⇒ e occurs before

Configuration: set of events, causally-closed and $\nearrow$-acyclic

[Baldan, Corradini, Montanari 98] [Vogler, Semenov, Yakovlev 98]

Constructing Ordinary Unfoldings





- Copy initial marking
- Repeat:
  - Find transition t and conditions X s.t.:
    - X is coverable
    - $h(X) = {}^\bullet t$
  - Add copy of t, with preset X, and copy of $t^\bullet$
- Until no such t and X can be found

Constructing Ordinary Unfoldings

For ordinary Petri nets,

Definition

Conditions c, c' are concurrent, $c \parallel c'$, iff some run marks them both.

Proposition

Conditions $c_1, \ldots, c_n$ are coverable iff $c_i \parallel c_j$ holds for all $i, j \in \{1, \ldots, n\}$

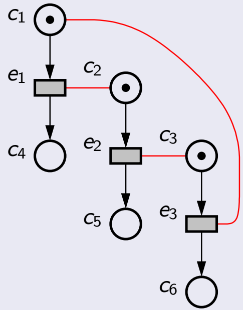
Conventional unfolders:

- Compute and store relation $\parallel$ as the unfolding construction progresses
- Use it to decide coverability of multiple conditions

[Esparza, Römer 99]

However, for contextual unfoldings...

...the same approach does not work:



- $c_4 \parallel c_5$ and $c_4 \parallel c_6$ and $c_5 \parallel c_6$ but $\{c_4, c_5, c_6\}$ is not coverable
- Cycle $e_1 \nearrow e_2 \nearrow e_3 \nearrow e_1$ of asymmetric conflict

Annotating Conditions with Histories [R., Schwoon, Baldan 11]

In short, the solution proposed:

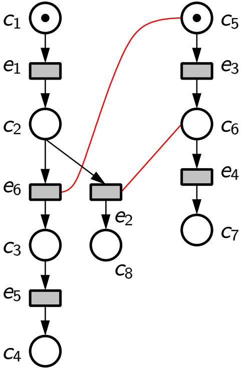
- Keeps track of conditions enriched with histories
- Defines $\|$ on these enriched conditions, instead of plain conditions
- Constructs $\|$ as unfolding progresses thanks to a characterization of $\|$

Annotating Conditions with Histories [R., Schwoon, Baldan 11]

Definition

Any configuration H is a history of e if:

1. $e \in H$
2. Any run of the events of H fires e last



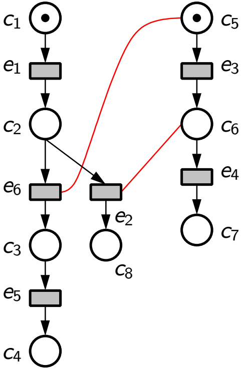
Annotating Conditions with Histories [R., Schwoon, Baldan 11]

Definition

Any configuration H is a history of e if:

1. $e \in H$
2. Any run of the events of H fires e last

$\{e_3, e_4\}$ ✓



Annotating Conditions with Histories [R., Schwoon, Baldan 11]

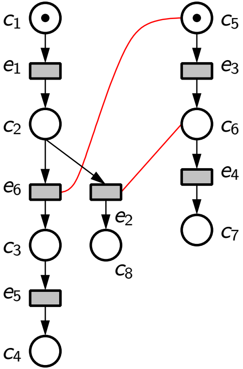
Definition

Any configuration H is a history of e if:

1. $e \in H$
2. Any run of the events of H fires e last

$\{e_3, e_4\}$ ✓

$\{e_1, e_3, e_4\}$ ✗ (run $e_3 e_4 e_1$)



Annotating Conditions with Histories [R., Schwoon, Baldan 11]

Definition

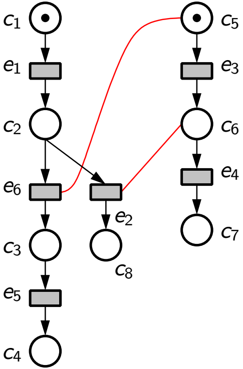
Any configuration H is a history of e if:

1. $e \in H$
2. Any run of the events of H fires e last

$\{e_3, e_4\}$ ✓

$\{e_1, e_3, e_4\}$ ✗ (run $e_3 e_4 e_1$)

$\{e_1, e_6, e_3, e_4\}$ ✓ ($e_6 \nearrow e_3$)



Annotating Conditions with Histories [R., Schwoon, Baldan 11]

Definition

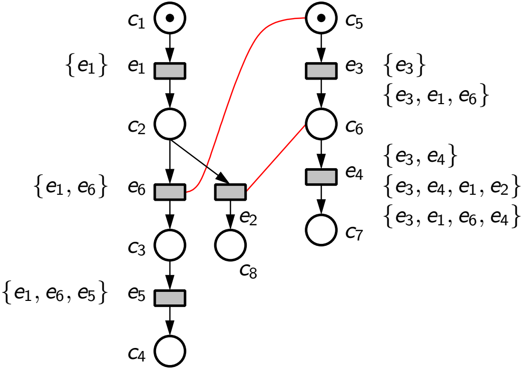
Any configuration H is a history of e if:

1. $e \in H$
2. Any run of the events of H fires e last

$\{e_3, e_4\}$ ✓

$\{e_1, e_3, e_4\}$ ✗ (run $e_3 e_4 e_1$)

$\{e_1, e_6, e_3, e_4\}$ ✓ ($e_6 \nearrow e_3$)



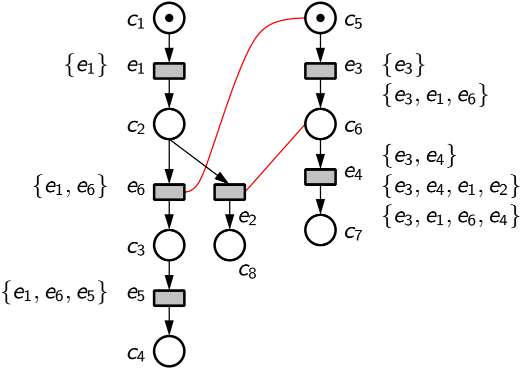
Annotating Conditions with Histories [R., Schwoon, Baldan 11]

Definition

Any configuration H is a history of e if:

1. $e \in H$
2. Any run of the events of H fires e last

- Enriched prefix: label condition c with histories of ${}^\bullet c$ and $\underline{c}$



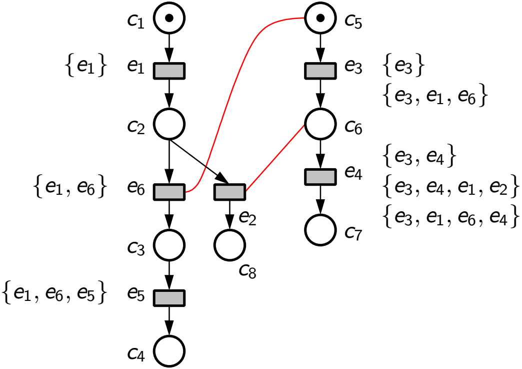
Annotating Conditions with Histories [R., Schwoon, Baldan 11]

Definition

Any configuration H is a history of e if:

1. $e \in H$
2. Any run of the events of H fires e last

- Enriched prefix: label condition c with histories of ${}^\bullet c$ and $\underline{c}$
- Enriched conditions: pairs $\langle c, H \rangle$



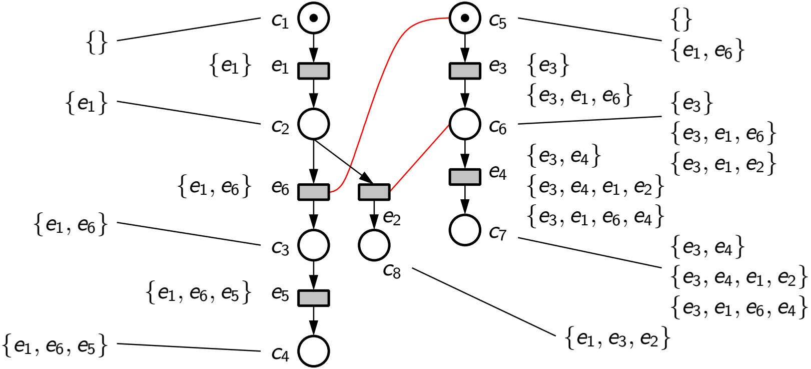
Annotating Conditions with Histories [R., Schwoon, Baldan 11]

Definition

Any configuration H is a history of e if:

1. $e \in H$
2. Any run of the events of H fires e last

- Enriched prefix: label condition c with histories of ${}^{\bullet}c$ and $\underline{c}$
- Enriched conditions: pairs $\langle c, H \rangle$



A Concurrency Relation for c-nets [R., Schwoon, Baldan 11]

Definition

Two enriched conditions $\rho = \langle c, H \rangle$ and $\rho' = \langle c', H' \rangle$ are concurrent, written $\rho \parallel \rho'$, iff:

$$H \text{ not in conflict with } H' \quad \text{and} \quad c, c' \in (H \cup H')^{\bullet}$$

A Concurrency Relation for c-nets [R., Schwoon, Baldan 11]

Definition

Two enriched conditions $\rho = \langle c, H \rangle$ and $\rho' = \langle c', H' \rangle$ are concurrent, written $\rho \parallel \rho'$, iff:

$$H \text{ not in conflict with } H' \quad \text{and} \quad c, c' \in (H \cup H')^\bullet$$

Proposition

Conditions $c_1, \ldots, c_n$ coverable iff there are histories $H_1, \ldots, H_n$ verifying

$$\langle c_i, H_i \rangle \parallel \langle c_j, H_j \rangle \text{ for all } i, j \in \{1, \ldots, n\}.$$

A Concurrency Relation for c-nets [R., Schwoon, Baldan 11]

Definition

Two enriched conditions $\rho = \langle c, H \rangle$ and $\rho' = \langle c', H' \rangle$ are concurrent, written $\rho \parallel \rho'$, iff:

$$H \text{ not in conflict with } H' \quad \text{and} \quad c, c' \in (H \cup H')^\bullet$$

Proposition

Conditions $c_1, \ldots, c_n$ coverable iff there are histories $H_1, \ldots, H_n$ verifying

$$\langle c_i, H_i \rangle \parallel \langle c_j, H_j \rangle \text{ for all } i, j \in \{1, \ldots, n\}.$$

Proposition

Let $\rho = \langle c, H \rangle$ and e be the last enriched condition and event appended to the prefix, let $\rho' = \langle c', H' \rangle$ be an arbitrary enriched condition. Then,

$$\rho \parallel \rho' \iff (c' \in e^\bullet \wedge H = H') \vee \left(c' \notin {}^\bullet e \;\wedge\; \bigwedge_{i=1}^{n} (\rho_i \parallel \rho') \;\wedge\; \underline{{}^\bullet e} \cap H' \subseteq H \right)$$

Challenges and The Cunf Tool [R., Schwoon 13]

Contextual unfoldings can be more compact, but

- Extra bookkeeping work for histories
- Prefix + histories: asymptotically same size as PR-unfolding

Driving questions

- Is contextual unfolding as efficient?
- For realistic cases, more compact?
- How do the various unfolding approaches compare?

The unfolder CUNF

- Asymmetric concurrency + dozen optimizations
- Robust tool, 7KLOC of C
- Integrated in Cosyverif environment (soon: TAPAAL and CPROVER)

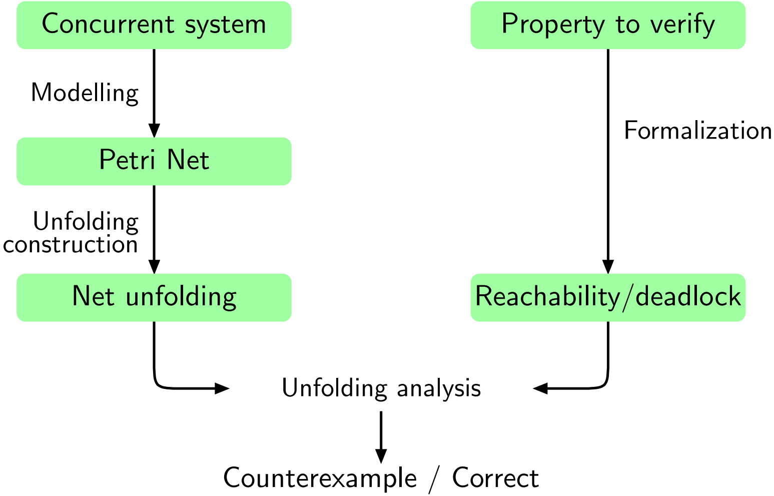
Experimental Results: Unfolding Construction

	Contextual		Ordinary		Ratios	
Net	Events	t_C	Events	t_P	t_C/t_P	t_C/t_R
bds_1.sync	1866	0.14	12900	0.51	0.27	0.54
byzagr4_1b	8044	2.90	14724	3.40	0.85	0.55
ftp_1.sync	50928	34.21	83889	76.74	0.45	0.30
furnace_4	95335	18.34	146606	40.39	0.45	0.42
key_4.fsa	4754	6.33	67954	2.21	2.86	1.47
rw_1w3r	14490	0.45	15401	0.38	1.18	0.65
q_1.sync	10722	1.13	10722	1.21	0.93	0.52
dpd_7.sync	10457	0.91	10457	0.88	1.03	0.92
elevator_4	16856	1.26	16856	2.01	0.63	$>$0.01
rw_12.sync	98361	3.10	98361	3.95	0.78	0.41
rw_2w1r	9241	0.40	9241	0.30	1.33	0.04

- C-net unfolding smaller or equal ordinary unfoldings
- In general faster than plain encoding
- Consistently faster than place-replication (t_R)

[R., Schwoon, Baldan 11] [R., Schwoon 13]

Model Checking with Net Unfoldings



Reachability Analysis with c-net Unfoldings [R., Schwoon 12]

Recall

For marking-complete prefix $\mathcal{P}_N$, deciding reachability of N is NP-complete

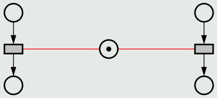
- Reduction to SAT
- Encodes existence of a configuration
- Acyclicity constraint for $\nearrow$ is problematic

Results

- Three optimizations to mitigate effects of acyclicity constraint
- Structural optimizations + logical simplification
- Tool Cna
- Experimental evaluation:
  method is practical and beats established approach on standard benchmark

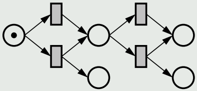
Improving Unfolding-based Verification: Outline

Concurrent read access



- Unfolding construction for nets with read arcs
- SAT-based reachability analysis
- Reduction of size: adequate orders
- Experimental evaluation

Sequences of choices



- Integration with merged processes
- SAT-based reachability analysis
- Characterization of mp-configurations
- Experimental evaluation

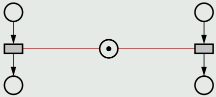
Fault diagnosis

(for conventional Petri nets)

- Generalization to partially-ordered observations
- Integration of fairness assumptions

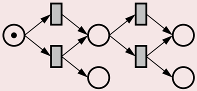
Improving Unfolding-based Verification: Outline

Concurrent read access



- Unfolding construction for nets with read arcs
- SAT-based reachability analysis
- Reduction of size: adequate orders
- Experimental evaluation

Sequences of choices



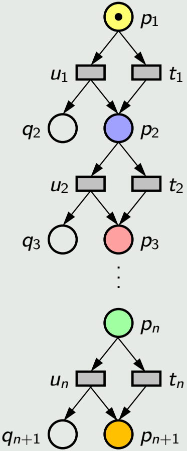
- Integration with merged processes
- SAT-based reachability analysis
- Characterization of mp-configurations
- Experimental evaluation

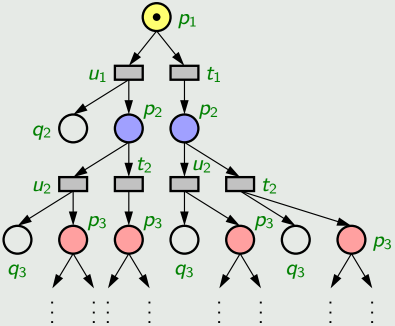
Fault diagnosis

(for conventional Petri nets)

- Generalization to partially-ordered observations
- Integration of fairness assumptions

Unfoldings Suffer from Conflicting Choices





2^n copies of place p_{n+1}

- All events reach different markings, no event is a cutoff
- The prefix is exponential

Combining Two Methods

We integrate two partial-order representations:

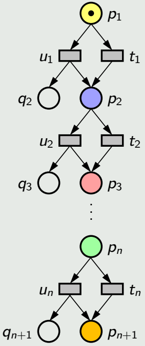
- Contextual unfoldings: address concurrent read access
- Merged Processes: address sequences of conflicts [Khomenko et al. 05]

Combining Two Methods

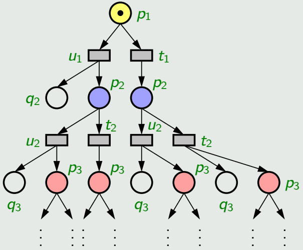
We integrate two partial-order representations:

- Contextual unfoldings: address concurrent read access
- Merged Processes: address sequences of conflicts [Khomenko et al. 05]

These methods address orthogonal sources of state explosion:



Net = Merged Process



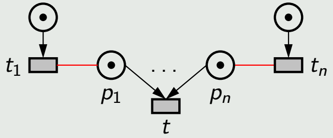
(Contextual) Unfolding

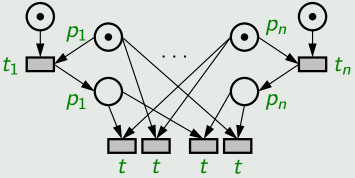
Combining Two Methods

We integrate two partial-order representations:

- Contextual unfoldings: address concurrent read access
- Merged Processes: address sequences of conflicts [Khomenko et al. 05]

These methods address orthogonal sources of state explosion:





C-net = Contextual unfolding

Merged Process

Combining Two Methods

We integrate two partial-order representations:

- Contextual unfoldings: address concurrent read access
- Merged Processes: address sequences of conflicts [Khomenko et al. 05]

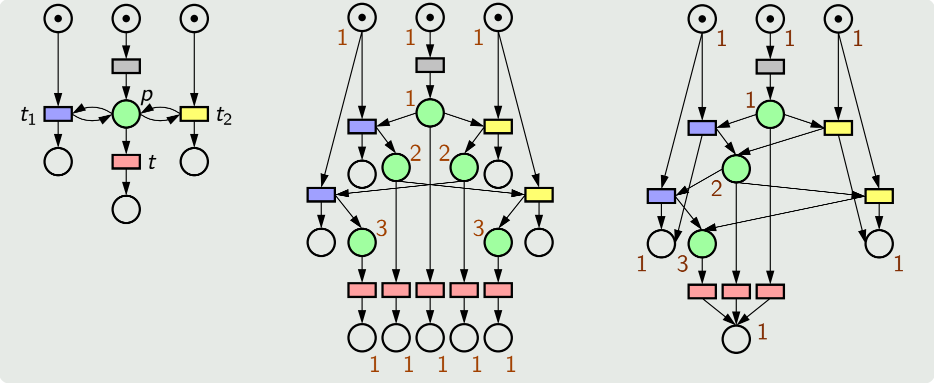
Resulting method: Contextual Merged Processes (CMPs)

Contextual Merged Processes: Main Idea

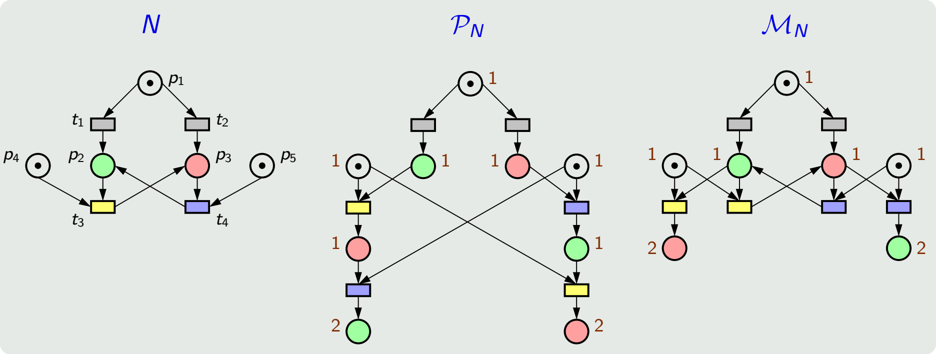
Definition [R., Schwoon, Khomenko 13]

The Contextual Merged Process (CMP) of the unfolding prefix $\mathcal{P}_N$ is the labelled c-net $\mathcal{M}_N$ resulting from

1. Merging all conditions with same occurrence depth and label
2. Eliminating duplicated events

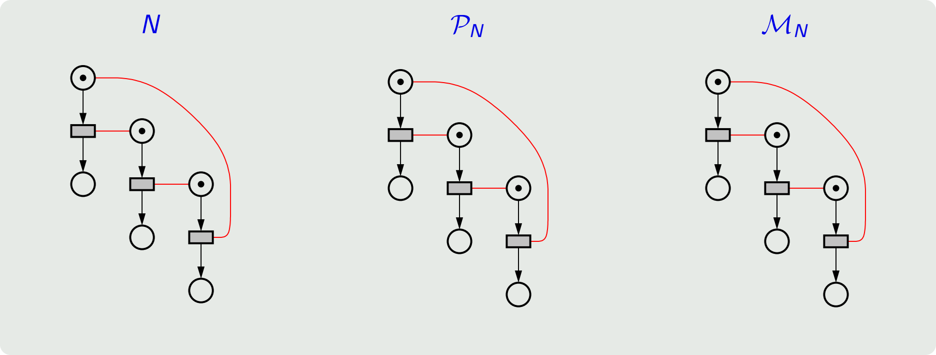


CMPs are in General not Acyclic



- Problem: CMPs have loops, transitions may fire more than once
- Prevents direct application of SAT-based analysis methods

CMPs are in General not Acyclic



- Problem: CMPs have loops, transitions may fire more than once
- Prevents direct application of SAT-based analysis methods

Acyclic Runs in CMPs

[R., Schwoon, Khomenko 13]

Proposition

If $\mathcal{P}_N$ is marking-complete then,

N's state-space is represented by $\mathcal{M}_N$'s $\nearrow$-acyclic runs

- Corollary: reachability of N is NP-complete on $\mathcal{P}_N$

Acyclic Runs in CMPs [R., Schwoon, Khomenko 13]

Proposition

If $\mathcal{P}_N$ is marking-complete then,

N's state-space is represented by $\mathcal{M}_N$'s $\nearrow$-acyclic runs

- Corollary: reachability of N is NP-complete on $\mathcal{P}_N$

Acyclicity of $\nearrow$ prevents both

- Contextual cycles involving read arcs (from c-net unfoldings)
- Cycles of causality (from merging)

Acyclic Runs in CMPs

[R., Schwoon, Khomenko 13]

Proposition

If $\mathcal{P}_N$ is marking-complete then,

N's state-space is represented by $\mathcal{M}_N$'s $\nearrow$-acyclic runs

- Corollary: reachability of N is NP-complete on $\mathcal{P}_N$

Acyclicity of $\nearrow$ prevents both

- Contextual cycles involving read arcs (from c-net unfoldings)
- Cycles of causality (from merging)

Additional results

- Reduction to SAT of reachability queries on N
- Encoding of mp-configurations into SAT (for direct construction)

Experiments with CMPs: Corbett Benchmarks

Benchmark		Unfolding		Merged Process	
Name	$\lvert T \rvert$	Plain	Contextual	Plain	Contextual
Bds	59	21.73	5.73	1.14	44
Brujin	165	3.22	1.64	1.44	127
Byz	409	46.11	25.57	1.03	303
Ftp	529	85.74	82.51	1.05	455
Knuth	137	2.88	1.59	1.31	112
Dme(8)	392	10.64	10.64	1.04	360
Dme(10)	490	15.53	15.53	1.04	450
Elev(3)	783	6.48	6.48	1.00	346
Elev(4)	1939	11.38	11.38	1.00	841
Key(2)	92	3.92	1.82	2.50	105
Key(3)	133	19.93	4.33	4.13	186
Key(4)	174	113.82	12.54	5.26	290
Mmgt(3)	172	4.01	4.01	1.00	355
Mmgt(4)	232	11.68	11.68	1.00	638

[R., Schwoon, Khomenko 13]

CMPs of Dijkstra's Mutual Exclusion Algorithm

```
b[0] = false;
while (k != 0) {
   if (b[k]) k = 0;
}

...
/* critical section */
...
```

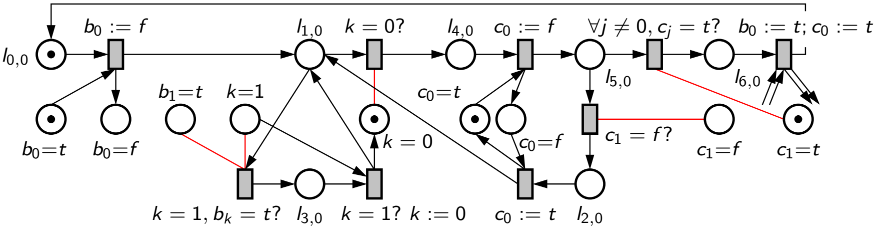
```
b[1] = false;
while (k != 1) {
   if (b[k]) k = 1;
}

...
/* critical section */
...
```

[R., Schwoon, Khomenko 13]

CMPs of Dijkstra's Mutual Exclusion Algorithm

```
b[0] = false;
while (k != 0) {
   if (b[k]) k = 0;
}


...
/* critical section */
...
```

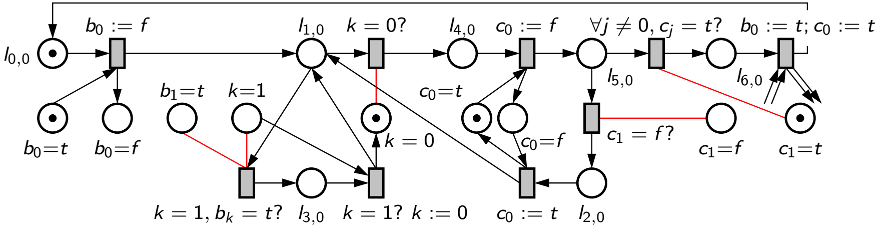
```
b[1] = false;
while (k != 1) {
   if (b[k]) k = 1;
}


...
/* critical section */
...
```



[R., Schwoon, Khomenko 13]

CMPs of Dijkstra's Mutual Exclusion Algorithm

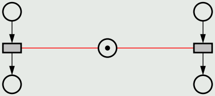
Net		Unfoldings		Merged Processes	
n	$\|T\|$	Petri Net	C-net	Petri Net	C-net
2	18	54	35	42	31
3	36	371	131	113	64
4	60	2080	406	220	105
5	90	10463	1139	375	155
6	126	49331	3000	589	214
	m	$\propto 5^m$	$\propto 3^m$	$\propto m^{1.5}$	$\propto m$



[R., Schwoon, Khomenko 13]

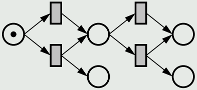
Improving Unfolding-based Verification: Outline

Concurrent read access



- Unfolding construction for nets with read arcs
- SAT-based reachability analysis
- Reduction of size: adequate orders
- Experimental evaluation

Sequences of choices



- Integration with merged processes
- SAT-based reachability analysis
- Characterization of mp-configurations
- Experimental evaluation

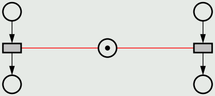
Fault diagnosis

(for conventional Petri nets)

- Generalization to partially-ordered observations
- Integration of fairness assumptions

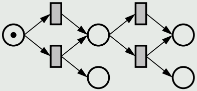
Improving Unfolding-based Verification: Outline

Concurrent read access



- Unfolding construction for nets with read arcs
- SAT-based reachability analysis
- Reduction of size: adequate orders
- Experimental evaluation

Sequences of choices



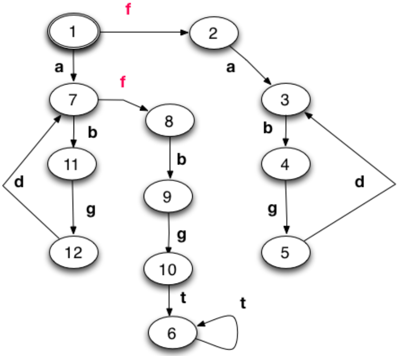
- Integration with merged processes
- SAT-based reachability analysis
- Characterization of mp-configurations
- Experimental evaluation

Fault diagnosis

(for conventional Petri nets)

- Generalization to partially-ordered observations
- Integration of fairness assumptions

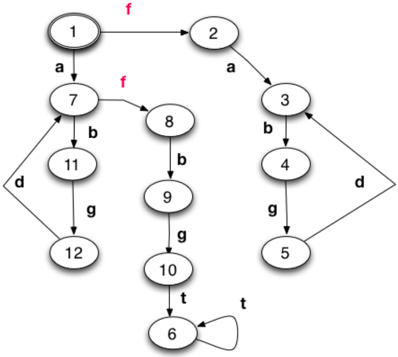
Diagnosis — Classical Approach



Partially-observable system S

[Sampath, Sengupata, Lafortune, Sinnamohideen, Teneketzis 95]

Diagnosis — Classical Approach

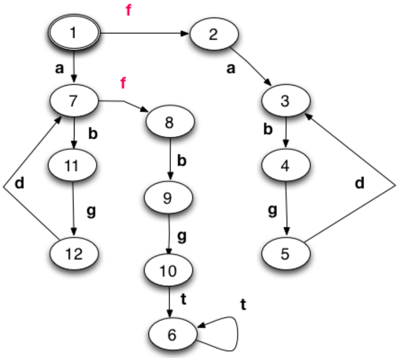


Partially-observable system S

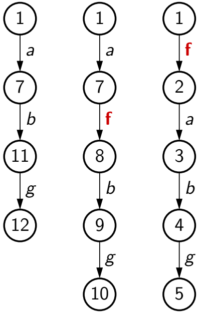
Observation a b g

[Sampath, Sengupata, Lafortune, Sinnamohideen, Teneketzis 95]

Diagnosis — Classical Approach



Partially-observable system S

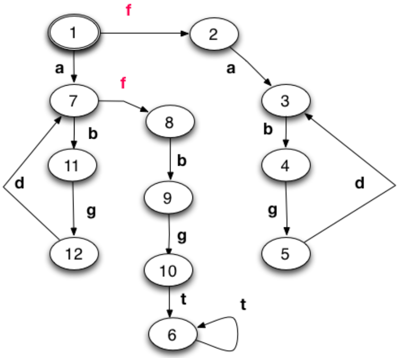


Explanations
expl(*abg*)

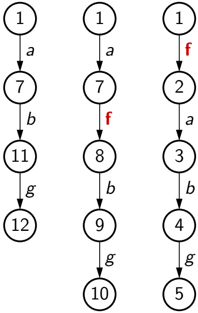
Observation a b g

[Sampath, Sengupata, Lafortune, Sinnamohideen, Teneketzis 95]

Diagnosis — Classical Approach



Partially-observable system *S*



Explanations

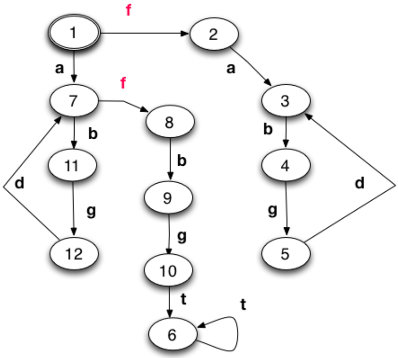
expl(*abg*)

Observation a b g

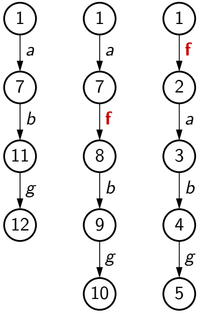
Diagnosis problems: Any/some run that explains the observation contains a fault?

[Sampath, Sengupata, Lafortune, Sinnamohideen, Teneketzis 95]

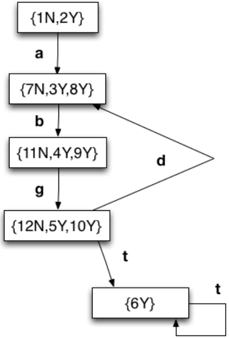
Diagnosis — Classical Approach



Partially-observable system S



Explanations
$expl(abg)$



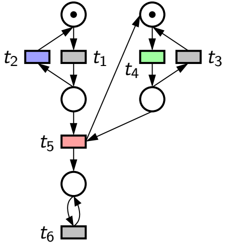
Diagnoser S_d

Observation a b g

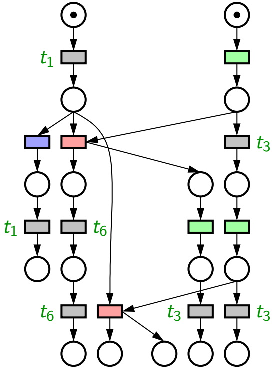
Diagnosis problems: Any/some run that explains the observation contains a fault?

[Sampath, Sengupata, Lafortune, Sinnamohideen, Teneketzis 95]

Diagnosis — Unfolding-based Approach



Partially-observable system S



Explanations

~~Diagnoser S_d~~

Observation: sequential or partially-ordered

[Benveniste, Fabre, Haar, Jard 03]

Contribution

	[SSLST95]	[BFHJ03]
Interleaving explosion	✗	✓
Partial-order observations	✗	✓
Unobservable loops	✗	✗

[SSLST95]: Sampath, Sengupata, Lafortune, Sinnamohideen, Teneketzis 95

[BFHJ03]: Benveniste, Fabre, Haar, Jard 03

Contribution

	[SSLST95]	[BFHJ03]	[EK12]
Interleaving explosion	✗	✓	✓
Partial-order observations	✗	✓	✗
Unobservable loops	✗	✗	✓

[SSLST95]: Sampath, Sengupata, Lafortune, Sinnamohideen, Teneketzis 95

[BFHJ03]: Benveniste, Fabre, Haar, Jard 03

[EK12]: Esparza, Kern 12

Contribution

	[SSLST95]	[BFHJ03]	[EK12]	This thesis
Interleaving explosion	✗	✓	✓	✓
Partial-order observations	✗	✓	✗	✓
Unobservable loops	✗	✗	✓	✓

[SSLST95]: Sampath, Sengupata, Lafortune, Sinnamohideen, Teneketzis 95

[BFHJ03]: Benveniste, Fabre, Haar, Jard 03

[EK12]: Esparza, Kern 12

This thesis: Haar, R., Schwoon 13

Contribution

	[SSLST95]	[BFHJ03]	[EK12]	This thesis
Interleaving explosion	✗	✓	✓	✓
Partial-order observations	✗	✓	✗	✓
Unobservable loops	✗	✗	✓	✓
Fairness	✗	✗	✗	✓

[SSLST95]: Sampath, Sengupata, Lafortune, Sinnamohideen, Teneketzis 95

[BFHJ03]: Benveniste, Fabre, Haar, Jard 03

[EK12]: Esparza, Kern 12

This thesis: Haar, R., Schwoon 13

Diagnosis with Unobservable Loops [Haar, R., Schwoon 13]

Diagnosis Problem

Given observation α, decide whether

all explanations in *expl*(α) contain a fault

Main challenge

- *expl*(α) may be infinite due to unobservable loops

- Define class of succinct explanations
- *expl*(α) contains only finitely many ones
- So they fit in a finite unfolding prefix $\mathcal{P}_\alpha$!

Results

- Cutoff criteria for constructing $\mathcal{P}_\alpha$
- SAT-based decision procedure

- Generalize [EK12] to partially-ordered observations

Weak Diagnosis: Diagnosis + Fairness [Haar, R., Schwoon 13]

Weak fairness: if some transition gets enabled, eventually it is disabled

Weak Diagnosis Problem

Given observation α, decide whether

any fair execution that contains an explanation in *expl*(α),
also contains a fault

Main challenge

- Need finite representation of maximal configurations of the unfolding that permits for checking set inclusion

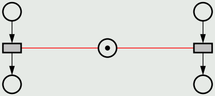
- Maximal configurations repeat spoiling paths that can be cut off

Results

- Cutoff criteria for building the representative prefixes
- SAT-based decision procedure

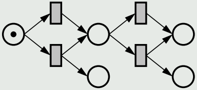
Conclusions

Concurrent read access



- Unfolding construction for nets with read arcs
- SAT-based reachability analysis
- Reduction of size: adequate orders
- Experimental evaluation

Sequences of choices



- Integration with merged processes
- SAT-based reachability analysis
- Characterization of mp-configurations
- Experimental evaluation

Fault diagnosis

(for conventional Petri nets)

- Generalization to partially-ordered observations
- Integration of fairness assumptions

Perspectives

- Unfoldings for other higher-level formalisms
    - Such as software
- Unfoldings vs. partial-order reductions
    - How can each profit from the strengths of the other?
- How much is worth to remember?
    - Contextual Merged Processes: direct construction
- Unfoldings and abstract interpretation
    - Unfoldings are exact abstractions of concurrency