Efficient contextual unfolding

César Rodríguez

(joint work with Stefan Schwoon)

LSV, ENS Cachan

February 2011

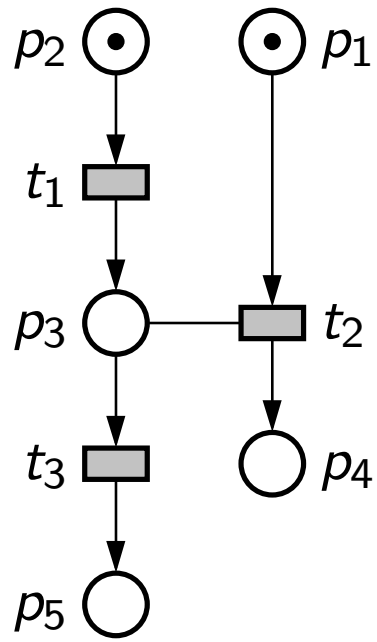
Outline

Contextual Nets

Definition

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

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



Notation

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

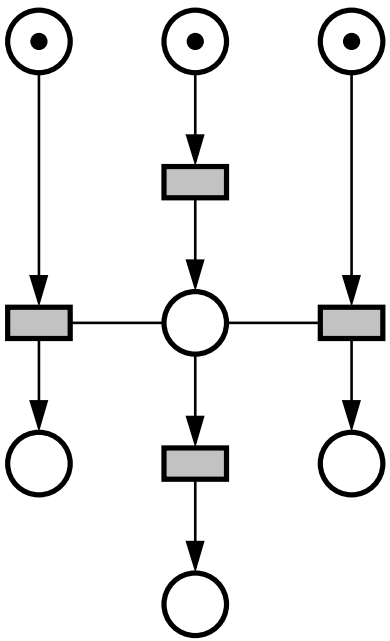
$\underline{t} = \{p \in P \mid (p, t) \in C\}$ for context

Example

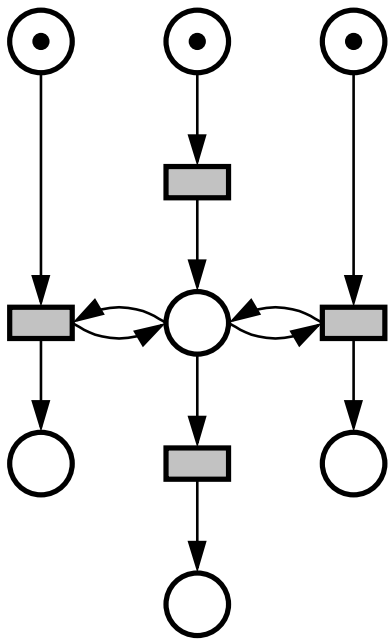
$\underline{t_2} = \{p_3\}$

Encoding contextual nets

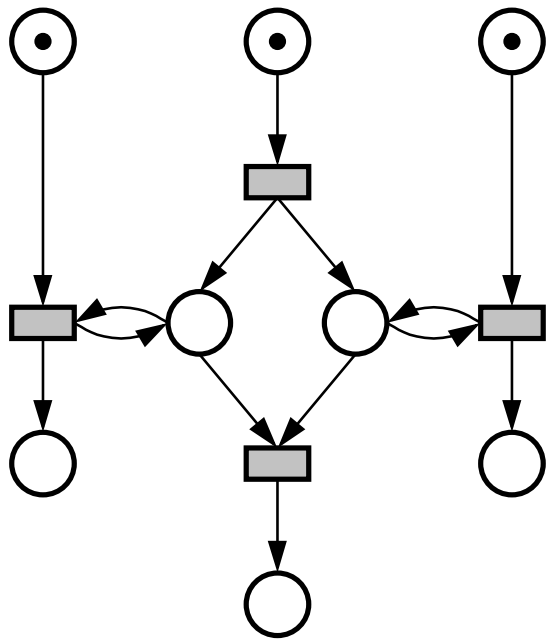
- We can encode a contextual net into a Petri net:



Contextual

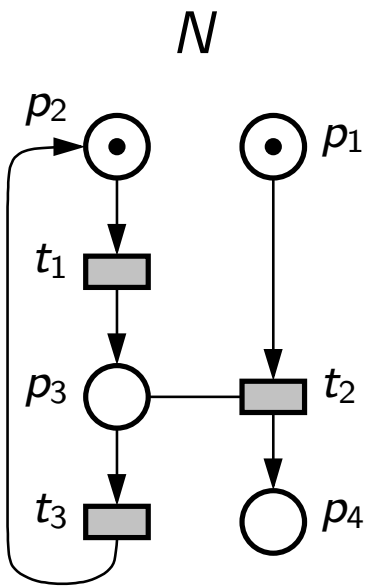


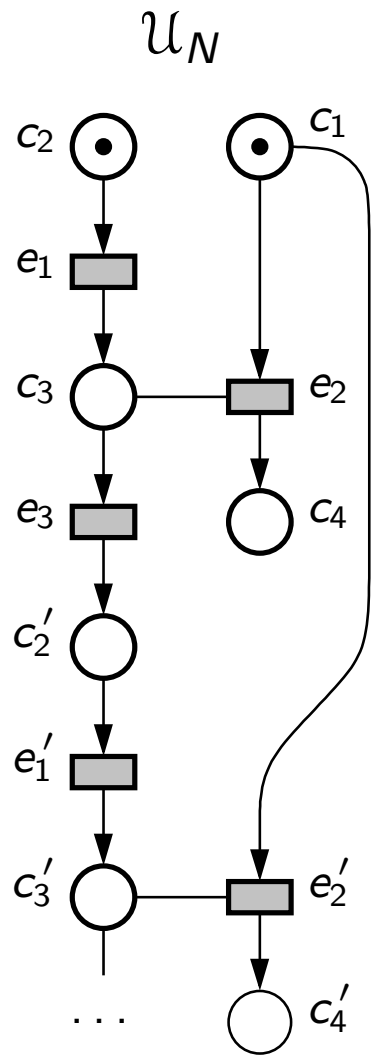
Plain Encoding



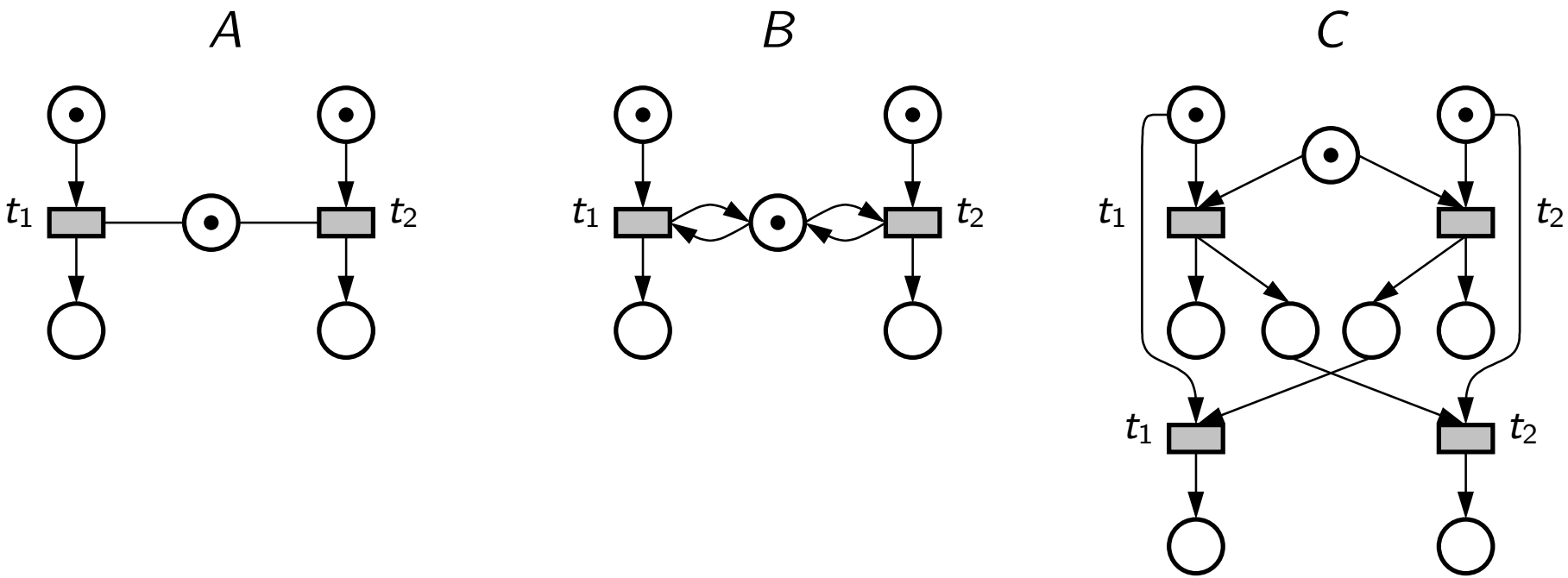
Place Replication Encoding

Contextual unfolding

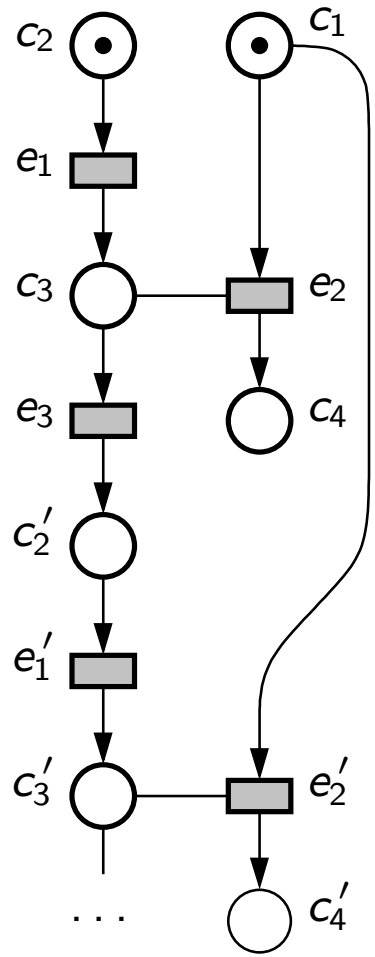




Contextual unfoldings exploit concurrency

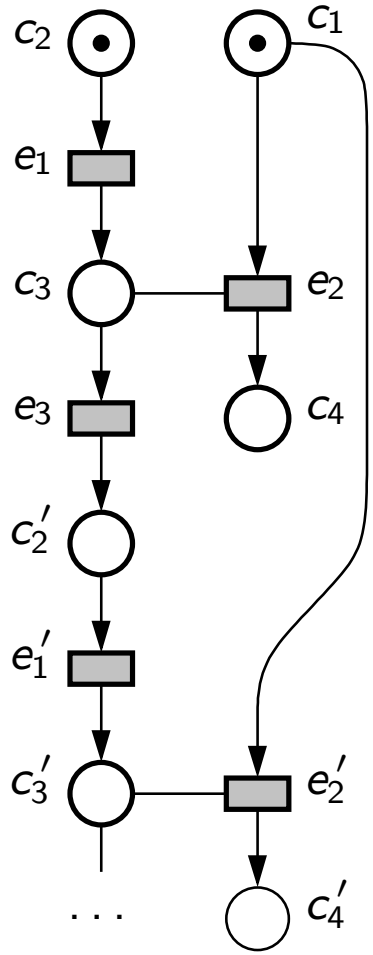


Asymmetric conflict and configurations



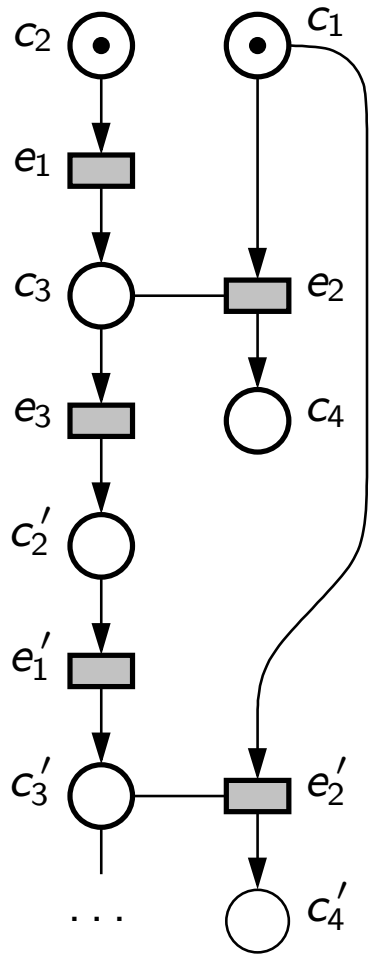
Asymmetric conflict and configurations

- Asymmetric conflict $\nearrow$
  - Two events e, e' are in asymmetric conflict, written $e \nearrow e'$, iff either:
    1. $e < e'$, or
    2. $\underline{e} \cap {}^\bullet e' \neq \emptyset$, or
    3. $e \neq e' \wedge {}^\bullet e \cap {}^\bullet e' \neq \emptyset$
  - Intuition: If both e and e' occur, then e occurs first



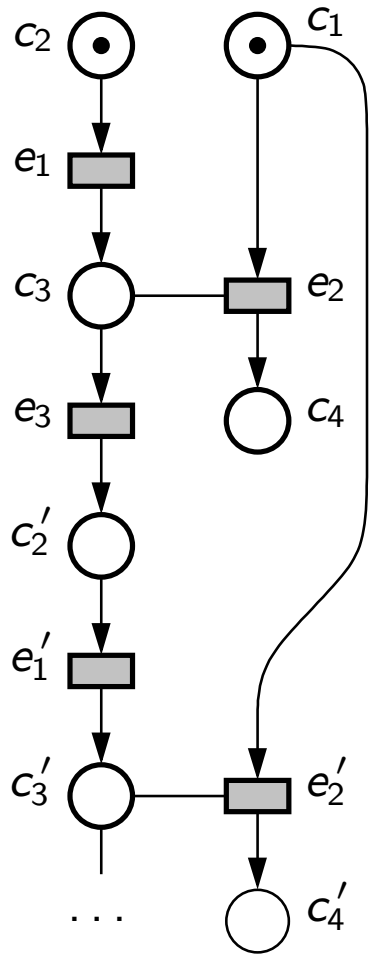
Asymmetric conflict and configurations

- Asymmetric conflict $\nearrow$
- Configuration
  - A set of events $C \subseteq T'$ is a configuration iff
    1. $e \in C$ and $e' < e$ implies $e' \in C$ (C is causally closed), and
    2. Relation $\nearrow \cap\, C \times C$ has no cycles
  - Intuition: C is a configuration iff all its events can be arranged to form a run.



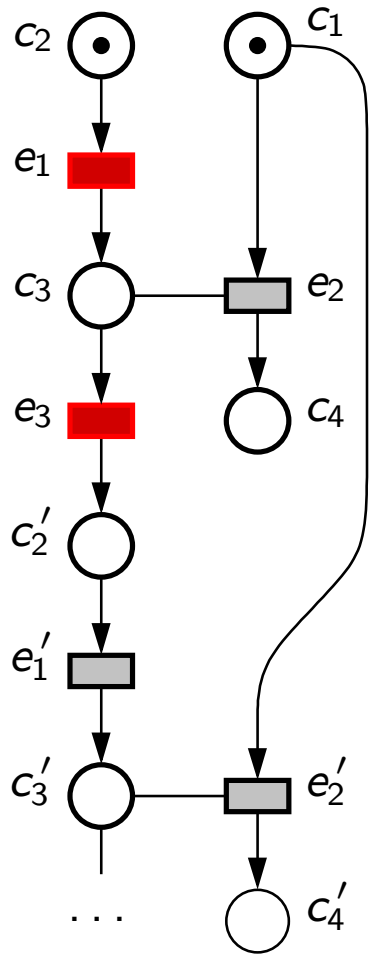
Asymmetric conflict and configurations

- Asymmetric conflict $\nearrow$
- Configuration
- History of an event
  - A configuration H is a history for any $e \in H$ iff any run of *all* events in H fires e last.
  - Example: e_3 has two histories



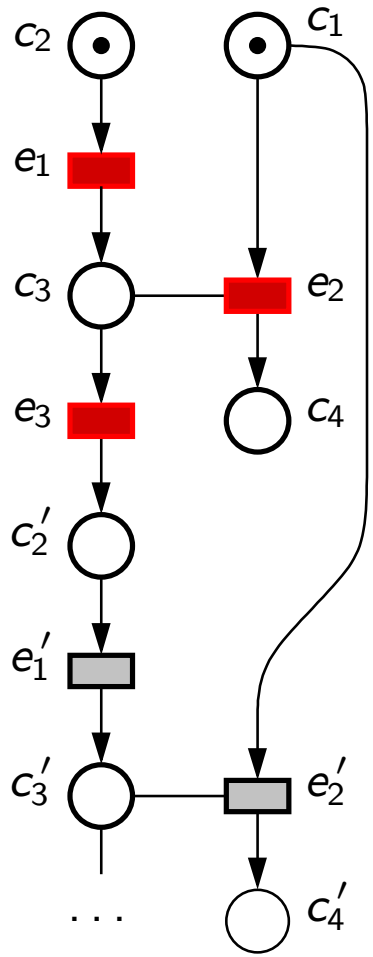
Asymmetric conflict and configurations

- Asymmetric conflict $\nearrow$
- Configuration
- History of an event
  - A configuration H is a history for any $e \in H$ iff any run of *all* events in H fires e last.
  - Example: e_3 has two histories



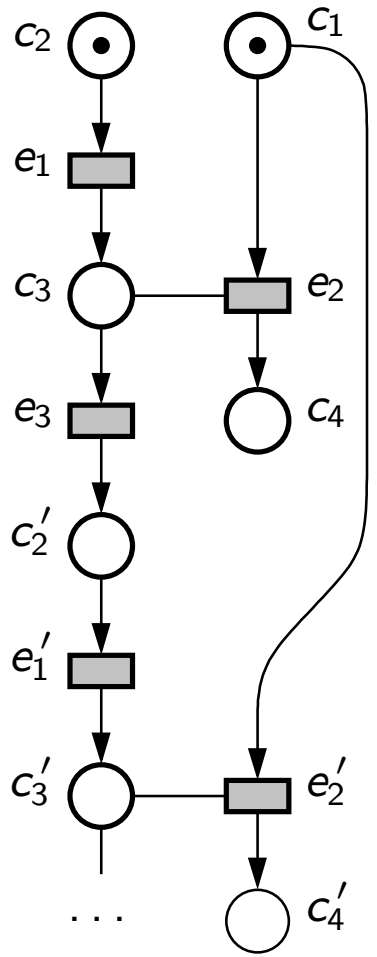
Asymmetric conflict and configurations

- Asymmetric conflict $\nearrow$
- Configuration
- History of an event
  - A configuration H is a history for any $e \in H$ iff any run of *all* events in H fires e last.
  - Example: e_3 has two histories



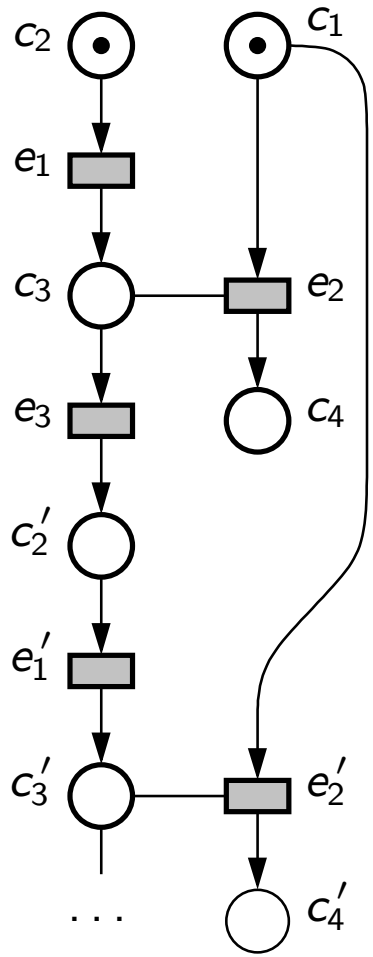
Asymmetric conflict and configurations

- Asymmetric conflict $\nearrow$
- Configuration
- History of an event
- Configuration conflict $\#$
  - Two configurations C_1, C_2 are in conflict, $C_1 \# C_2$, iff there exist events $e \in C_2 \setminus C_1$ and $e' \in C_1$ such that $e \nearrow e'$ (or vice versa).
  - Intuition: $C_1 \# C_2$ iff after firing C_1, some event in $C_2 \setminus C_1$ cannot fire, or vice versa.



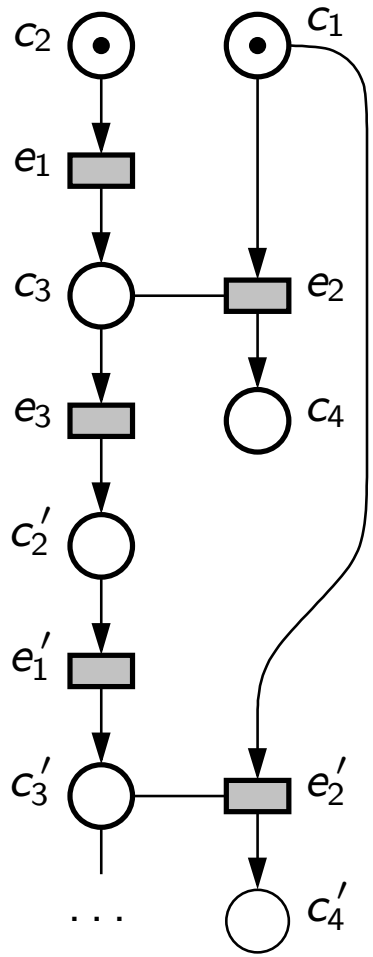
Asymmetric conflict and configurations

- Asymmetric conflict $\nearrow$
- Configuration
- History of an event
- Configuration conflict $\#$
- Cut
  - For a configuration C, the $\mathrm{Cut}(C)$ is the marking reached in $\mathcal{U}_N$ when firing events in C
  - Example: $\mathrm{Cut}(\{e_1\}) = \{c_1, c_3\}$



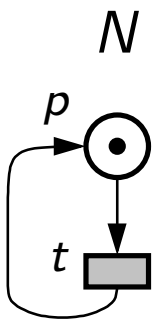
Asymmetric conflict and configurations

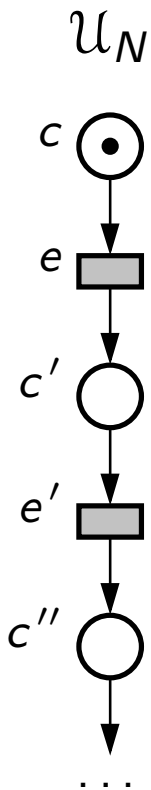
- Asymmetric conflict $\nearrow$
- Configuration
- History of an event
- Configuration conflict $\#$
- Cut
- Marking
  - For a configuration C, we define $\text{Marking}(C) = f_T(\text{Cut}(C))$.



Complete prefixes

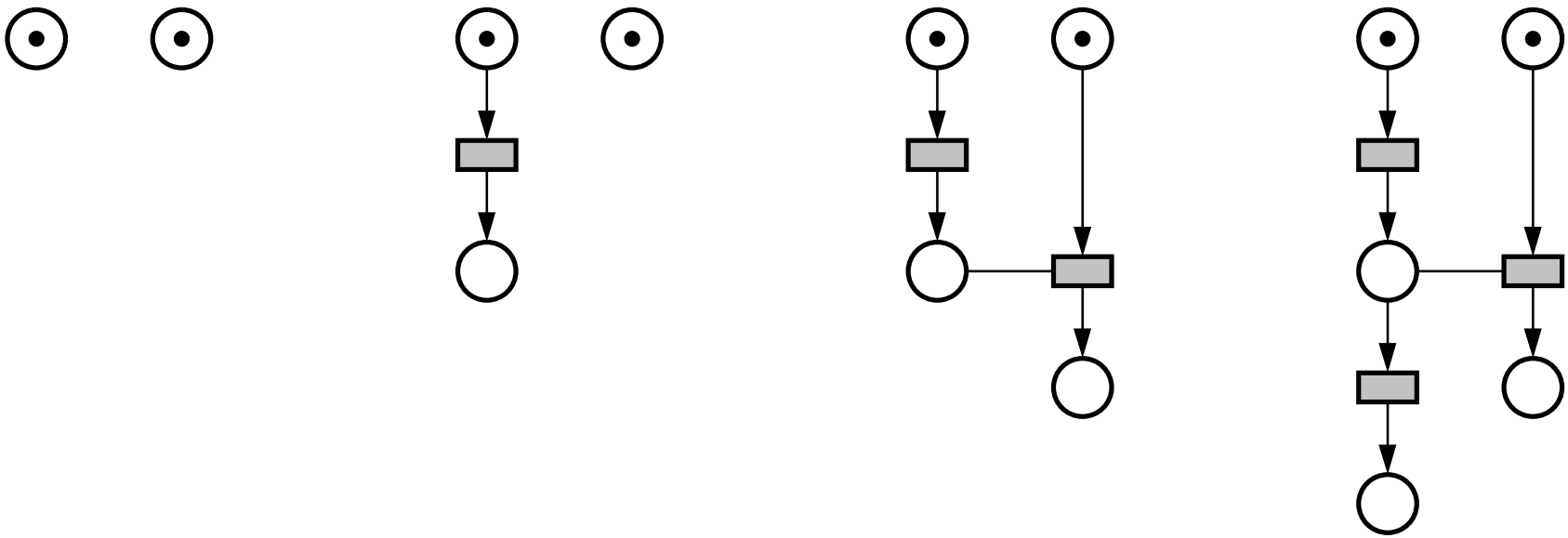
- In general $\mathcal{U}_N$ is infinite.





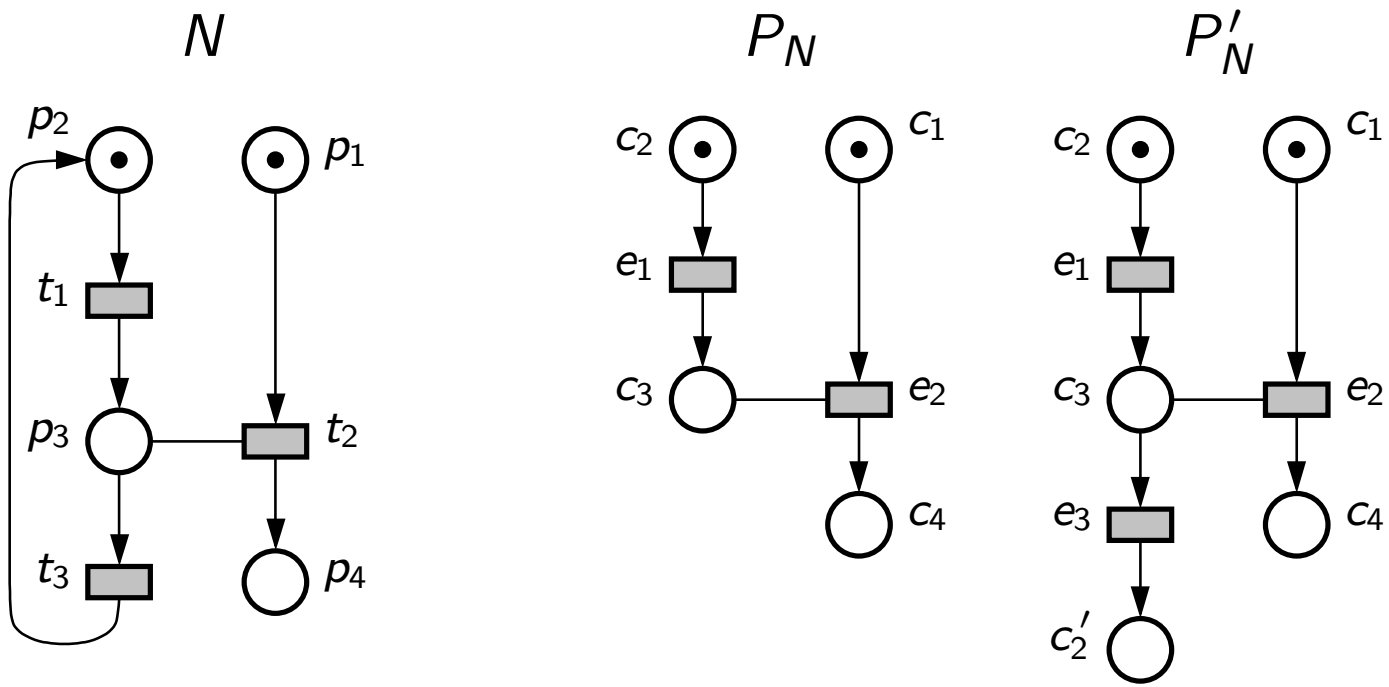
Complete prefixes

- In general $\mathcal{U}_N$ is infinite.
- A prefix $\mathcal{P}_N$ of the full unfolding $\mathcal{U}_N$ is the contextual net resulting from applying the inductive rules above a finite number of times.



Complete prefixes

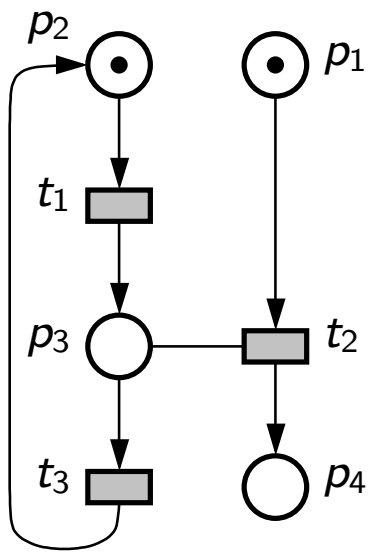
- In general $\mathcal{U}_N$ is infinite.
- A prefix $\mathcal{P}_N$ of the full unfolding $\mathcal{U}_N$ is the contextual net resulting from applying the inductive rules above a finite number of times.
- Prefix $\mathcal{P}_N$ is complete if it represents every reachable marking in N.



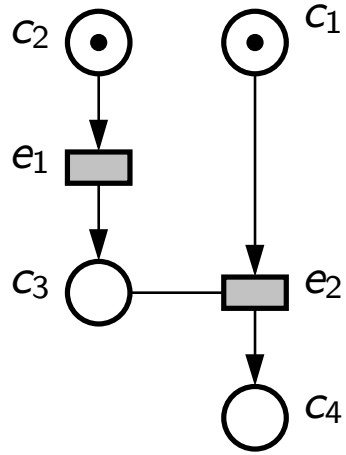
Complete prefixes

- In general $\mathcal{U}_N$ is infinite.
- A prefix $\mathcal{P}_N$ of the full unfolding $\mathcal{U}_N$ is the contextual net resulting from applying the inductive rules above a finite number of times.
- Prefix $\mathcal{P}_N$ is complete if it represents every reachable marking in N.
- Our goal: to (efficiently) compute a finite and complete prefix for N.

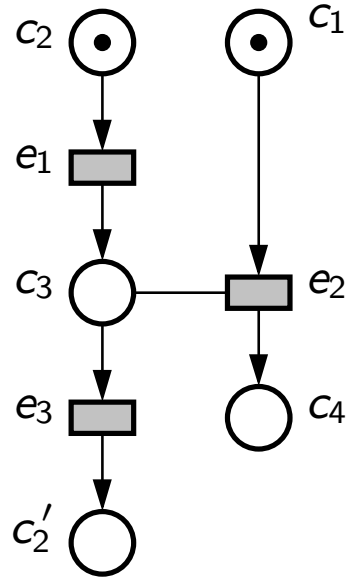
N



P_N



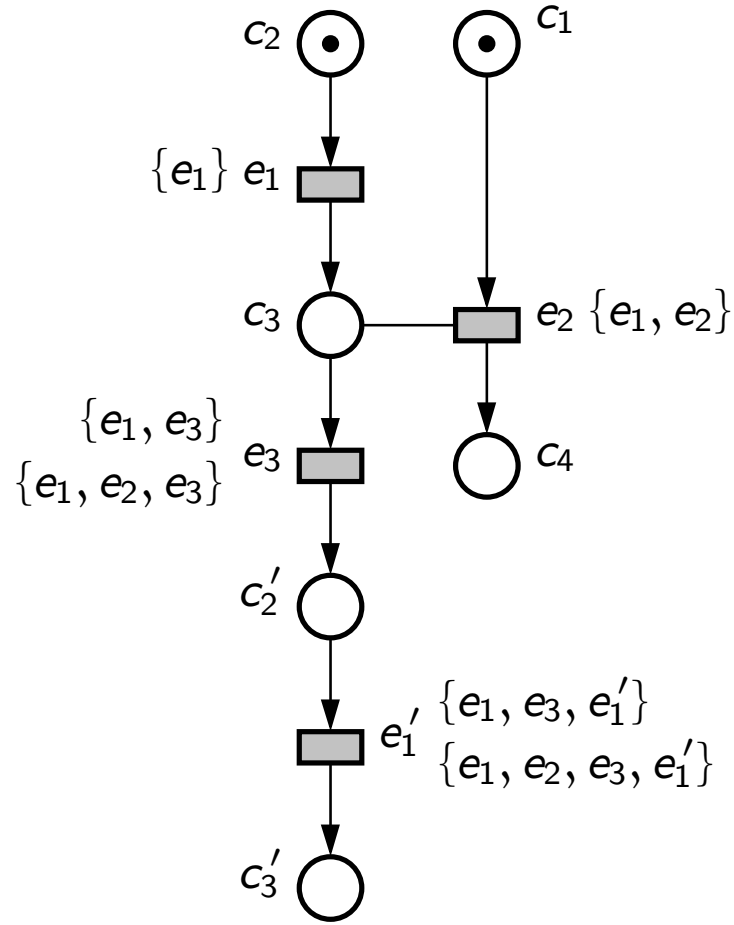
P'_N



Computing a complete prefix

We follow the abstract procedure proposed by:

Baldan et al. McMillan's complete prefix for contextual nets. In *Transactions of PNOMC*, p. 199-220, Berlin, 2008. Springer-Verlag.



Computing a complete prefix

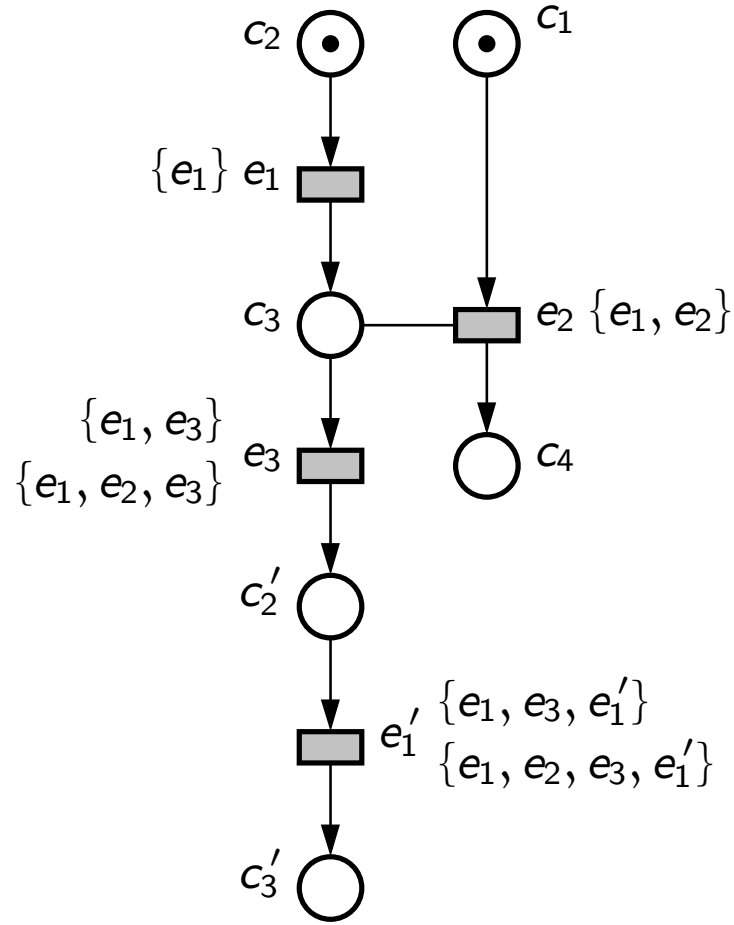
Definition

An enriched prefix $\mathcal{E}_N = \langle \mathcal{P}_N, \chi \rangle$ is a prefix $\mathcal{P}_N$ of the full unfolding whose events e are annotated by a set $\chi(e)$ of histories of e.

A pair (e, H) such that H is a history of e is called enriched event.

Remark

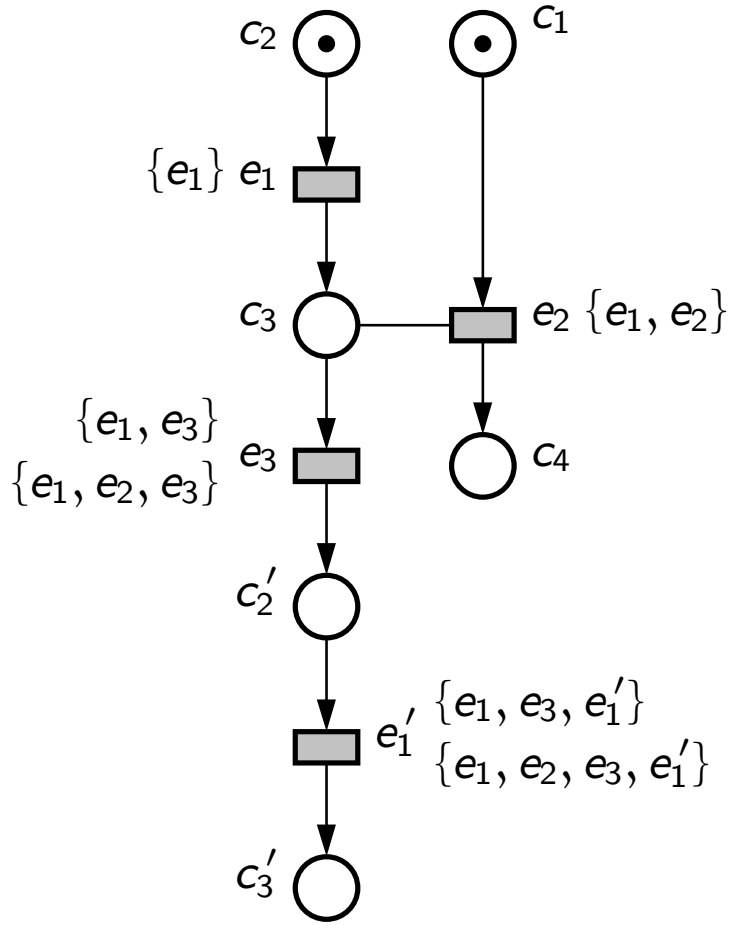
This is our working data structure



Computing a complete prefix

In a nutshell...

- Iterative construction procedure
- Works on enriched prefixes
  1. Start with initial marking.
  2. Extend enriched prefix with one enriched event at a time.
  3. Skip addition of cutoff enriched events.



Challenging problems

- Data structure to store histories
- Efficient procedure to compute extensions to the prefix

Computing prefix extensions

The problem

Given $\mathcal{E}_N$ and conditions $c_1, \ldots, c_n$ of $\mathcal{E}_N$, how to efficiently determine whether $c_1, \ldots c_n$ are coverable

Let's first review the problem for Petri nets...

Definition

Conditions c, c' are concurrent, $c \parallel c'$, iff there exist a firing sequence marking them both.

Proposition

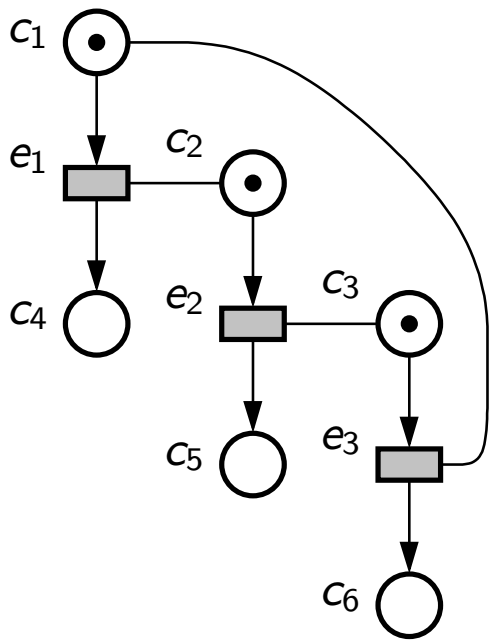
Conditions $c_1, \ldots c_n$ are coverable iff $c_i \parallel c_j$ holds for all $i, j \in \{1, \ldots, n\}$

Remark

We can construct $\parallel$ together with the unfolding.

However, for contextual unfolding...

...the same approach doesn't work:



We have $c_4 \parallel c_5$, and $c_4 \parallel c_6$ and $c_5 \parallel c_6$ but $\{c_4, c_5, c_6\}$ is not coverable.

What to do?

Enriched conditions

Definition

Let c be a condition.

- if ${}^\bullet c = \{e\}$ and H is a history of e, then H is a generating history of c; if ${}^\bullet c = \emptyset$, then $\emptyset$ is.
- if $e \in \underline{c}$ and H is a history of e, then H is a reading history of c.
- A history of c is any
  - generating or reading history of c;
  - union $H_1 \cup H_2$ of non-conflicting histories of c.

We call (c, H) an enriched condition.

Example

$\{e_2\}$ is a generating history for c_5 and a reading history for c_3.

A concurrency relation for contextual nets

Definition

Let $\rho = (c, H)$ and $\rho' = (c', H')$ be two enriched conditions. We say that ρ is concurrent to ρ', written $\rho \parallel \rho'$, iff:

$$\neg(H \# H') \quad \text{and} \quad c, c' \in \mathrm{Cut}(H \cup H')$$

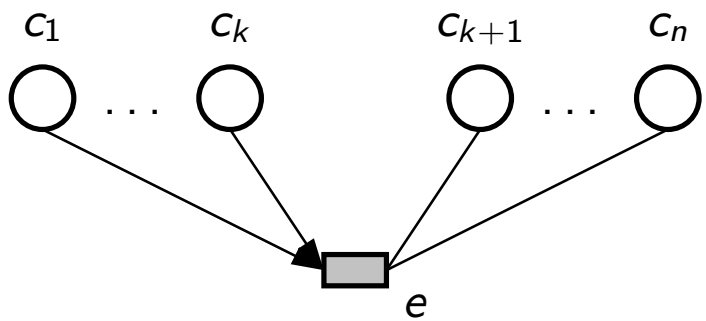
Property

Conditions $c_1, \ldots, c_n$ coverable iff there exist histories $H_1, \ldots, H_n$ verifying $(c_i, H_i) \parallel (c_j, H_j)$ for all $1 \leq i < j \leq n$.

Composing histories

Let e be an event such that ${}^\bullet e = \{c_1, \ldots, c_k\}$ and $\underline{e} = \{c_{k+1}, \ldots, c_n\}$. Then H is a history of e iff there exist *arbitrary* histories $H_1, \ldots, H_k$ for $c_1, \ldots, c_k$ and *generating* histories $H_{k+1}, \ldots, H_n$ for $c_{k+1}, \ldots, c_n$ such that:

- $H = \{e\} \cup \bigcup_{i=1}^{m} H_i$, and
- $(c_i, H_i) \parallel (c_j, H_j)$ for all $1 \leq i < j \leq n$.



Remark

Provides a strategy for unfolding procedure: start with generating histories for initial conditions; for every new enriched condition, use above theorem to construct new event/condition histories.

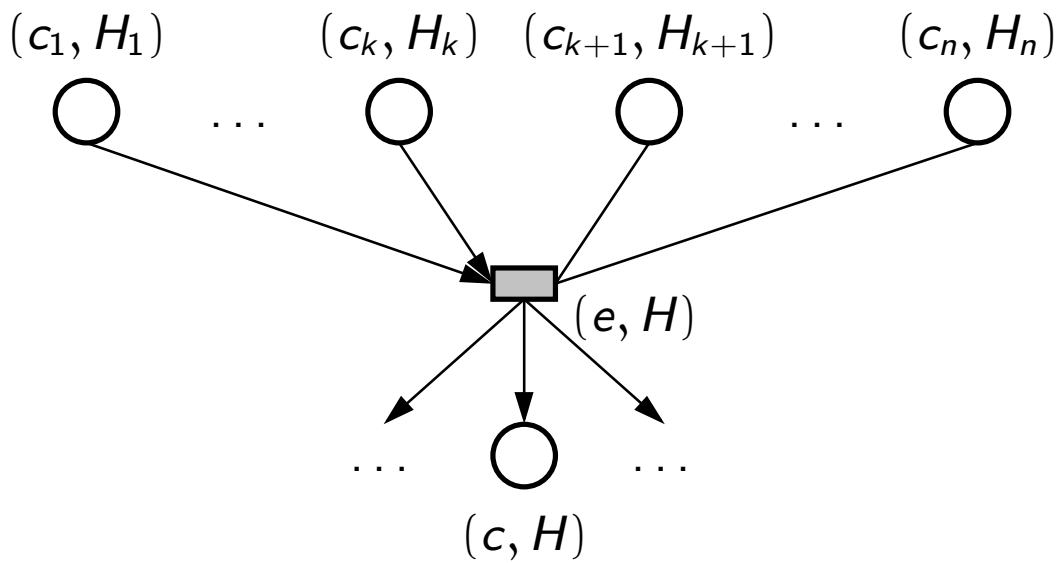
Computing the concurrency relation

Let $\rho = (c, H)$ be a generating condition and let e be such that $\{e\} = {}^\bullet c$. Let $\rho_1, \ldots, \rho_n$ be the enriched conditions used to compose H. Then,

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

Remarks

- Reasonable efficiency when implementing $\underline{{}^\bullet e} \cap H' \subseteq H$
- Efficient detection of coverable sets akin to Petri net method.



History graph

Let $\mathcal{E}_N$ be an enriched prefix of $\mathcal{U}_N$. The history graph $\mathcal{H}_N$ for $\mathcal{E}_N$ is a node-labelled directed graph whose

- Nodes are enriched events (e, H) of $\mathcal{E}_N$.
- Edge $(e, H) \to (e', H')$ exists iff H' has been used to construct H.

Node (e, H) is labelled by e.

Remark

All required operations on histories can be implemented as simple neighbourhood queries on $\mathcal{H}_N$.

Benchmarks

Benchmarks used: Corbett's set of examples

- Standard benchmarks in unfolding literature
- Derived from concurrent finite automata, hence 1-safe
- Different characteristics, fairly sure to exhibit implementation flaws
- Not specifically geared towards contextual nets

Experiments I: Mole vs Cunf

	events	Mole	Cunf
bds_1	12900	0.47	0.52
buf100	5051	2.85	2.10
byzagr4	14724	3.04	3.40
dpd_7	10457	0.93	0.87
dph_7	37272	0.79	0.99
elevator_4	16856	2.00	2.01
fifo20	41792	4.89	4.14
ftp_1	83889	76.02	77.09
furnace_3	25394	1.22	1.10
key_4	67954	1.80	2.18
q_1.sync	10722	1.36	1.22
rw_12.sync	98361	2.89	3.98
rw_1w3r	15401	0.30	0.39
rw_2w1r	9241	0.23	0.29

Mole is an (efficient) unfolder for Petri nets.

Cunf is the new contextual unfolder.

We run both tools on the original Petri nets (no read arcs!) $\Rightarrow$ results are the same

Times given in seconds.

Conclusion: implementation of Cunf is reasonably efficient (factors 0.7 to 1.4 w.r.t. Mole)

Conclusion: histories handled effortlessly

Experiments II: Plain vs contextual

	events	Plain	Cont.	enr. ev.
bds_1	12900	0.52	0.16	4032
buf100	5051	2.10	2.17	5051
byzagr4	14724	3.40	2.59	8044
dpd_7	10457	0.87	0.94	10457
dph_7	37272	0.99	0.99	37272
elevator_4	16856	2.01	1.30	16856
fifo20	41792	4.14	4.14	41792
ftp_1	83889	77.09	34.60	50928
furnace_3	25394	1.10	0.62	16893
key_4	67954	2.18	9.35	21742
q_1.sync	10722	1.22	1.20	10722
rw_12.sync	98361	3.98	3.14	98361
rw_1w3r	15401	0.39	0.43	14982
rw_2w1r	9241	0.29	0.36	9241

Contextual nets obtained by converting read/write loops to read arcs.

Cunf used on both tools.

3 examples w/o read arcs (in italics).

Some savings on time and size (not always on both).

Inefficiency detected in key_4 example, we are working on fixing it.

Experiments III: Contextual vs PR-encoding

	Context.	PR-enc.
bds_1	0.16	0.27
buf100	2.17	2.16
byzagr4	2.59	5.30
dpd_7	0.94	0.98
dph_7	0.99	1.00
elevator_4	1.30	557.06
fifo20	4.14	4.12
ftp_1	34.60	113.71
furnace_3	0.62	0.96
key_4	9.35	4.28
q_1.sync	1.20	2.18
rw_12.sync	3.14	7.66
rw_1w3r	0.43	0.70
rw_2w1r	0.36	8.86

PR-encoding obtained from contextual nets.

Cunf used on both tools.

nr. histories in contextual = nr. events in PR

Nonetheless, contextual is consistently better (except key_4, for now).

Explanation: combinatorial problems in PR due to larger transition presets.

Conclusions and future work

- Contextual unfolding feasible and efficient
- Beats PR-encoding
- Work in progress, further ideas for optimization
- Will look at more extensive benchmarks
- To do: look at the applications in verification, diagnosis, etc.