Unfolding-based Reachability Checking of Petri Nets

César Rodríguez

Dept. of Computer Science, University of Oxford, UK

Group Seminar, University of Oxford, February 20, 2014

Unfoldings: Symbolic Representations

- Compact, symbolic representation of concurrent state-space
- Originated from the partial-order semantics of Petri nets, 1970s-1980s
- Ken McMillan [CAV'92]: use them for practical verification
  - Finite, complete unfolding prefix for finite-state Petri nets
- Reachability, deadlock, LTL, ...

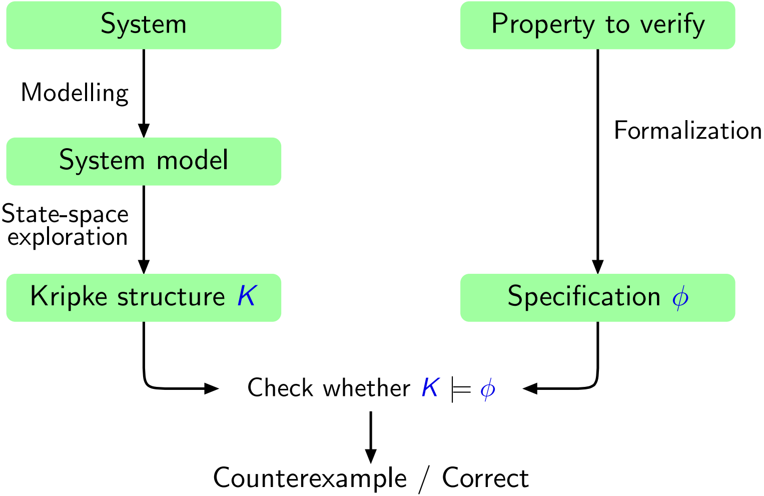
Unfoldings: Symbolic Representations

- Compact, symbolic representation of concurrent state-space
- Originated from the partial-order semantics of Petri nets, 1970s-1980s
- Ken McMillan [CAV'92]: use them for practical verification
  - Finite, complete unfolding prefix for finite-state Petri nets
- Reachability, deadlock, LTL, ...

Here we focus on

- Three semantics of Petri nets
- Unfolding structure and properties
- Unfolding construction and analysis (briefly)

Model Checking



Coping with State-space Explosion

Explosion due to

- Concurrency
- Non-determinism
- Data
- Unsafeness...

Coping with State-space Explosion

Explosion due to

- Concurrency
- Non-determinism
- Data
- Unsafeness...

Alleviating state-space explosion

Abstraction: Aggregate similar states, by throwing away information and possibly repairing inaccuracies
e.g., Abstract Interpretation, CEGAR

Reduction: Discard irrelevant states, by identifying *equivalent* computations and examining only one representative
e.g., Partial-order reduction

Compression: Use compact lossless representation, that handles many states at once without losing any of them
e.g., BDDs, Unfoldings.

Coping with State-space Explosion

Explosion due to

- Concurrency
- Non-determinism
- Data
- Unsafeness...

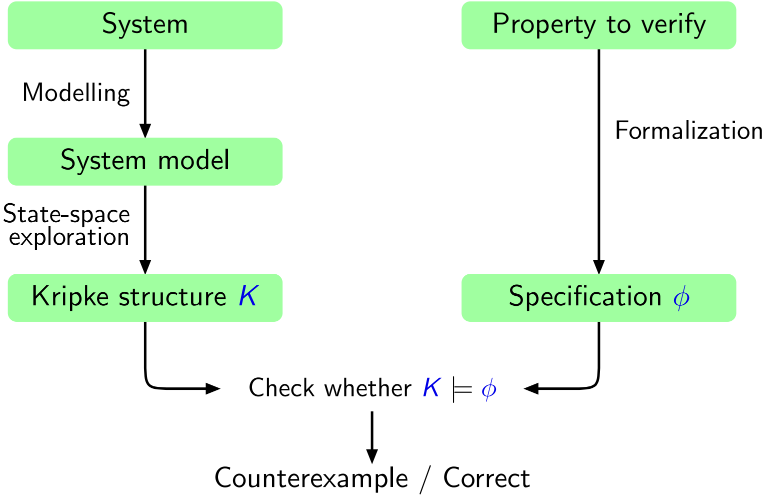
Alleviating state-space explosion

Abstraction: Aggregate similar states, by throwing away information and possibly repairing inaccuracies
e.g., Abstract Interpretation, CEGAR

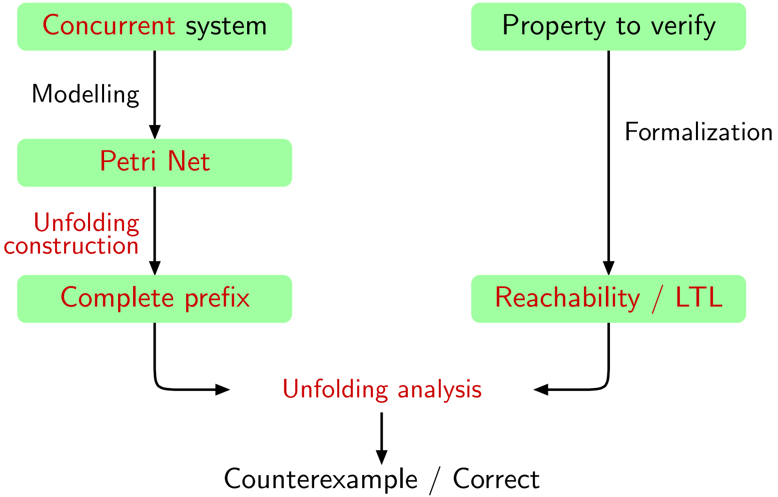
Reduction: Discard irrelevant states, by identifying *equivalent* computations and examining only one representative
e.g., Partial-order reduction

Compression: Use compact lossless representation, that handles many states at once without losing any of them
e.g., BDDs, Unfoldings.

- BDDs: exploit regularity of homogeneous components
- Unfoldings: exploit concurrency of components


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



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]

Outline

1. Petri Nets
2. Non-sequential Semantics
3. Unfolding Semantics
4. Finite, Complete Prefixes
5. Summary

Petri Nets

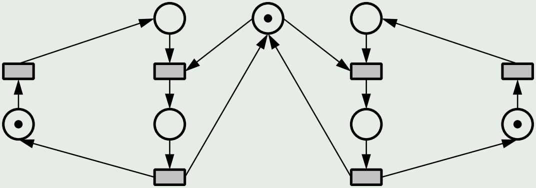
- Petri nets are fundamental model of concurrent and distributed systems
- Invented by Carl Adam Petri in the 1960s (at the age of 12)

- Petri nets contain places and transitions
- Places model states, conditions, or resources
- Transitions model actions carried out on places

A lot of literature available about Petri nets, for instance:

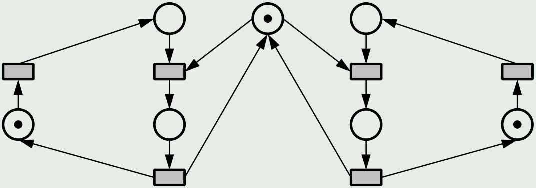
Wolfgang Reisig, *Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets*, Springer, 1998

Petri Nets — Example



The ○ are places
The ▭ are transitions
The • are tokens
The → are arcs

Petri Nets — Example

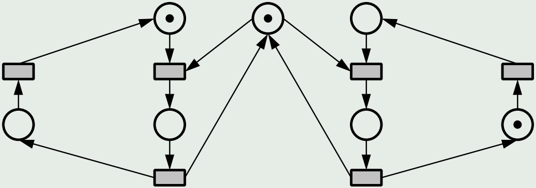


The ○ are places
The ▭ are transitions
The • are tokens
The → are arcs

Allowed patterns:

Forbidden patterns:

Petri Nets — Example

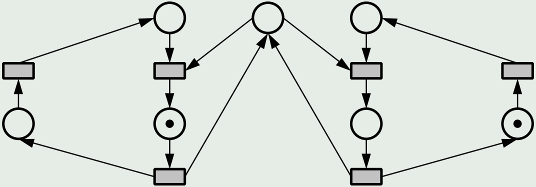


The  are places
The  are transitions
The  are tokens
The  are arcs

Allowed patterns:

Forbidden patterns:

Petri Nets — Example

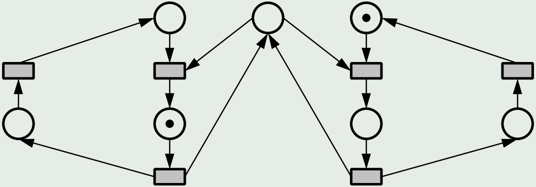


The  are places
The  are transitions
The  are tokens
The  are arcs

Allowed patterns:

Forbidden patterns:

Petri Nets — Example

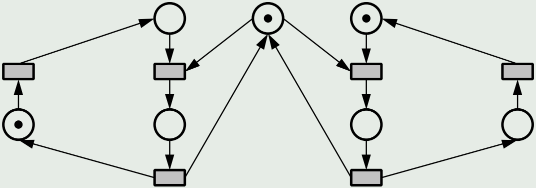


The	○	are places
The	▭	are transitions
The	•	are tokens
The	⟶	are arcs

Allowed patterns:

Forbidden patterns:

Petri Nets — Example



The  are places
The  are transitions
The  are tokens
The  are arcs

Allowed patterns:

Forbidden patterns:

Petri Nets — Formalization

Definition

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

- P: finite set of places
- T: finite set of transitions
- $F \subseteq P \times T \cup T \times P$: flow relation
- $m_0 \colon P \to \{0, 1\}$: initial marking

Definition

The preset and postset of a transition or place x are:

$$\text{Preset:} \quad {}^{\bullet}x := \{y \in P \cup T : (y, x) \in F\}$$

$$\text{Postset:} \quad x^{\bullet} := \{y \in P \cup T : (x, y) \in F\}$$

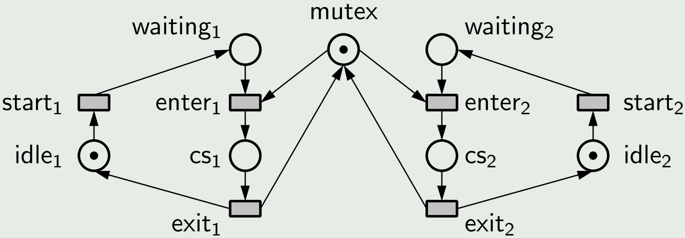
Petri Nets — Markings

Definition

A marking of N is a function

$$m\colon P \to \mathbb{N}$$

that maps places to the number of tokens they contain.



$m(\text{idle}_1) = 1$

$m(\text{mutex}) = 1$

$m(\text{idle}_2) = 1$

$m(p) = 0$ for any other $p \in P$

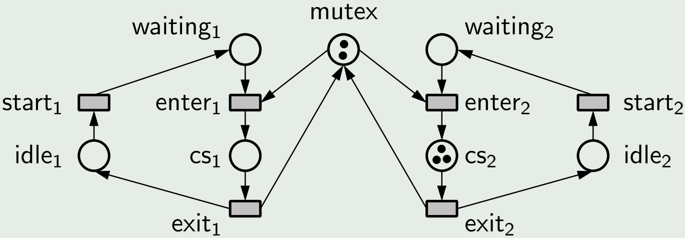
Petri Nets — Markings

Definition

A marking of N is a function

$$m\colon P \to \mathbb{N}$$

that maps places to the number of tokens they contain.



$m(\text{mutex}) = 2$

$m(\text{cs}_2) = 3$

$m(p) = 0$ for any other $p \in P$

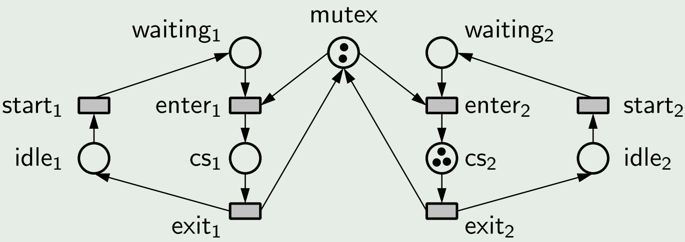
Petri Nets — Enabling Rule

Definition

A transition t is enabled at a marking m iff

$$m(p) \geq 1 \text{ for all } p \in {}^\bullet t,$$

i.e., if the marking covers the preset of t.



exit$_2$ is enabled, but enter$_2$ is not

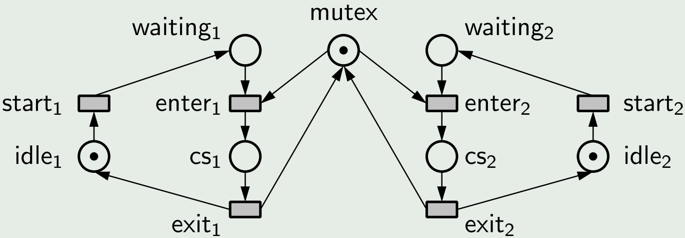
Petri Nets — Enabling Rule

Definition

A transition t is enabled at a marking m iff

$$m(p) \geq 1 \text{ for all } p \in {}^\bullet t,$$

i.e., if the marking covers the preset of t.



Only start$_1$ and start$_2$ are enabled

Petri Nets — Firing a Transition

Definition

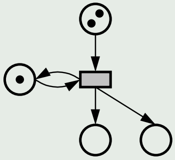
A transition t enabled at marking m can fire, producing a new marking m', denoted as

$$m \xrightarrow{t} m'$$

where m' is defined as

$$m'(p) = m(p) + \begin{cases} 1 & \text{if } p \in t^\bullet \setminus {}^\bullet t \\ -1 & \text{if } p \in {}^\bullet t \setminus t^\bullet \\ 0 & \text{otherwise} \end{cases}$$

for all $p \in P$.



Petri Nets — Firing a Transition

Definition

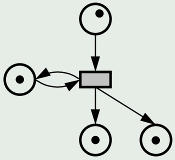
A transition t enabled at marking m can fire, producing a new marking m', denoted as

$$m \xrightarrow{t} m'$$

where m' is defined as

$$m'(p) = m(p) + \begin{cases} 1 & \text{if } p \in t^\bullet \setminus {}^\bullet t \\ -1 & \text{if } p \in {}^\bullet t \setminus t^\bullet \\ 0 & \text{otherwise} \end{cases}$$

for all $p \in P$.



Petri Nets — Operational Semantics

Let $N := \langle P, T, F, m_0 \rangle$ be a Petri net,

Definition: operational semantics

The operational semantics of N is the edge-labelled transition system

$$M_N := \langle S, \Delta, s_0 \rangle$$

defined as

- $S :=$ set of markings $m \colon P \to \mathbb{N}$ of N
- $\Delta := \{\langle m, t, m' \rangle \colon$ there is $t \in T$ such that $m \xrightarrow{t} m'\}$
- $s_0 := m_0$, the initial marking of N

Petri Nets — Operational Semantics

Let $N := \langle P, T, F, m_0 \rangle$ be a Petri net,

Definition: operational semantics

The operational semantics of N is the edge-labelled transition system

$$M_N := \langle S, \Delta, s_0 \rangle$$

defined as

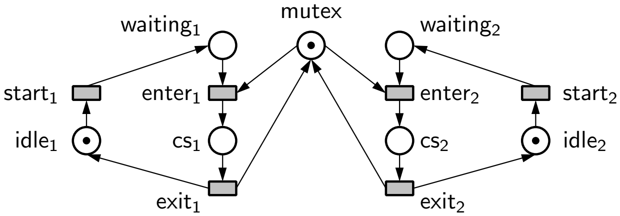
- $S :=$ set of markings $m \colon P \to \mathbb{N}$ of N
- $\Delta := \{\langle m, t, m' \rangle \colon$ there is $t \in T$ such that $m \xrightarrow{t} m'\}$
- $s_0 := m_0$, the initial marking of N

Definition

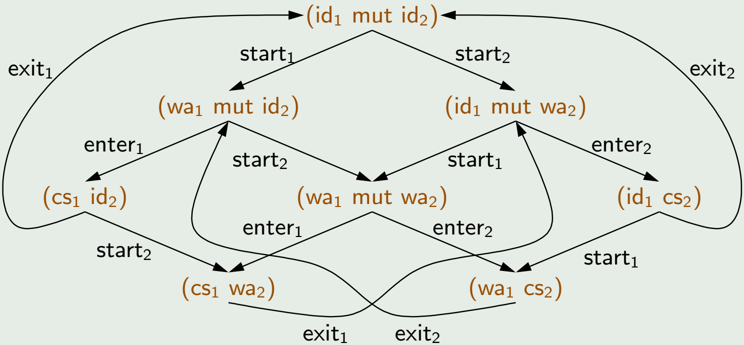
The reachability set of N is the smallest set $reach(N)$ satisfying

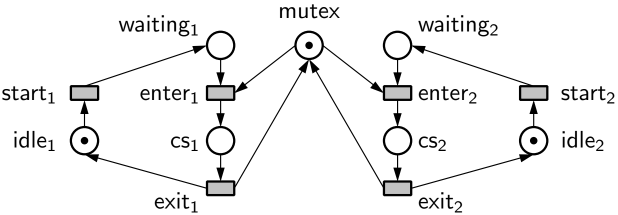
1. $m_0 \in reach(N)$
2. if $m \in reach(N)$ and $m \xrightarrow{t} m'$, for any $t \in T$, then $m' \in reach(N)$.

Petri Nets — Operational Semantics: Example



Petri Nets — Operational Semantics: Example





Petri Nets — Run

Definition

A run, or firing sequence of N is any sequence of transitions

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

which labels at least one path

$$m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} m_2 \xrightarrow{t_3} \ldots$$

in M_N starting from the initial marking m_0. The set of runs of N is denoted by $runs(N)$.

Petri Nets — Boundedness

Definition

A marking m of N is

- k-bounded if $m(p) \leq k$ for all $p \in P$;
- bounded if it is k-bounded for some $k \in \mathbb{N}$;
- safe if it is 1-bounded.

By extension N is safe or bounded if all markings in $reach(N)$ so are.

Petri Nets — Boundedness

Definition

A marking m of N is

- k-bounded if $m(p) \leq k$ for all $p \in P$;
- bounded if it is k-bounded for some $k \in \mathbb{N}$;
- safe if it is 1-bounded.

By extension N is safe or bounded if all markings in $reach(N)$ so are.

Proposition

The Petri net N is bounded iff $reach(N)$ is finite

- All nets we have seen so far were safe
- For the rest of the talk, we focus on bounded Petri nets

Petri Nets — Reachability

Reachability Problem

- Given: a net N and a marking m
- Decide: if $m \in reach(N)$

Coverability Problem

- Given: a net N and a partial function $M \colon P \to \mathbb{N}$
- Decide: if there is $m \in reach(N)$ such that $m(p) \geq M(p)$ for all places $p \in P$

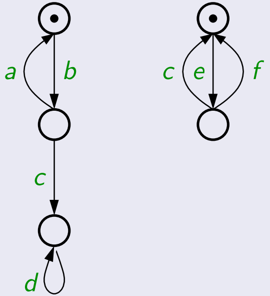
Boundedness Problem

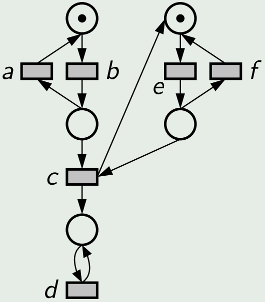
- Given: a net N
- Decide: whether $reach(N)$ is finite, i.e., whether N is bounded

Petri Nets — Decidability and Complexity

	Bounded net	Unbounded net
Reachability	PSPACE-complete	EXPSPACE-hard
Coverability	PSPACE-complete	EXSPACE-complete
LTL model checking	PSPACE-complete	Undecidable
Boundedness	N/A	EXPSPACE-complete

Communicating Automata

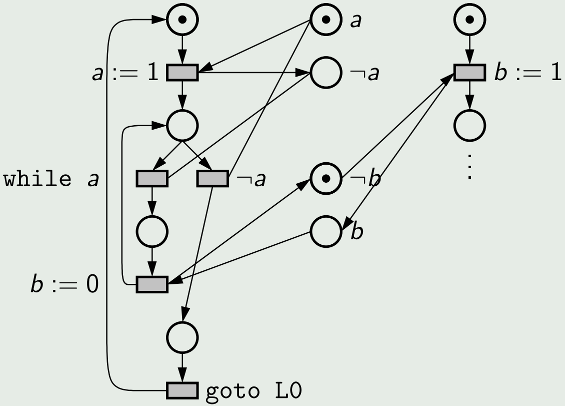




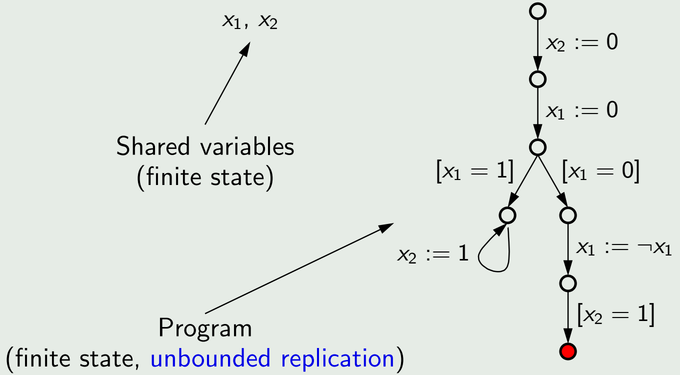
Concurrent Boolean Programs

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

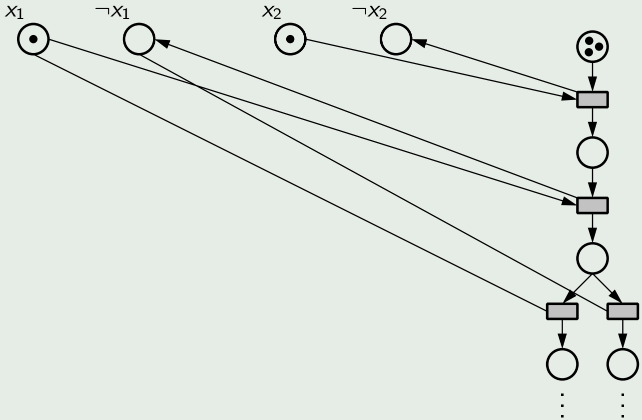
```
L1:  b := 1;
     while (b) a := 0;
     goto L1;
```



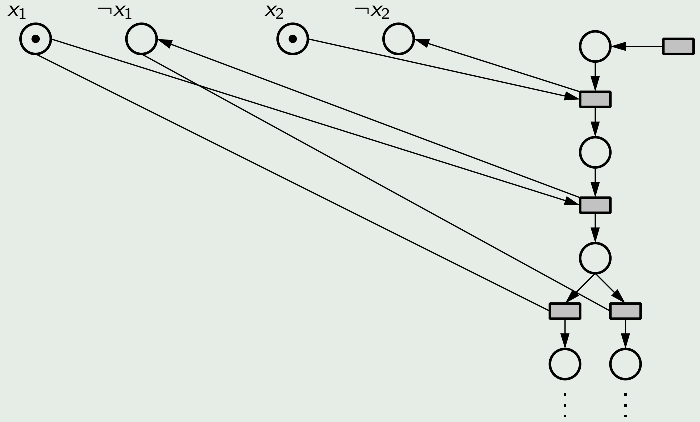
Counter Abstractions



Counter Abstractions

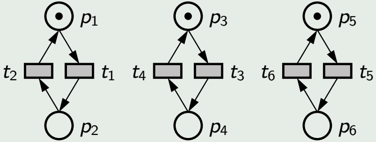


Counter Abstractions



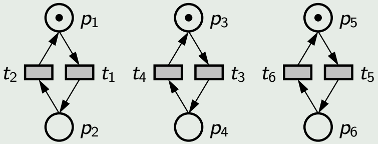
Outline

State-Explosion: Concurrency

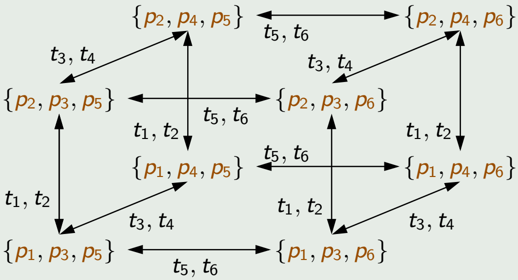


- 2^3 reachable markings

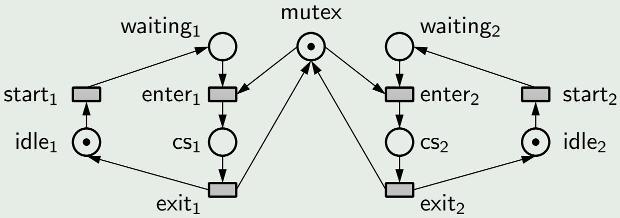
State-Explosion: Concurrency



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

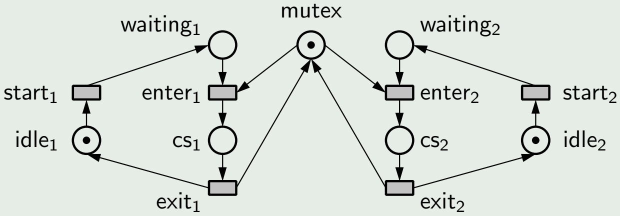


Processes (or configurations) of a Petri Net



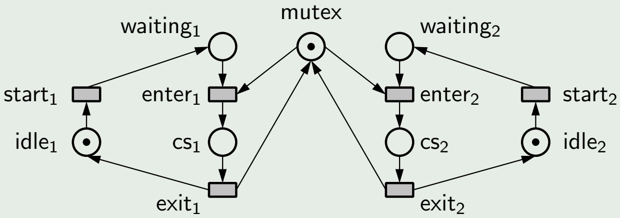
start$_1$, start$_2$, enter$_1$
start$_2$, start$_1$, enter$_1$

Processes (or configurations) of a Petri Net

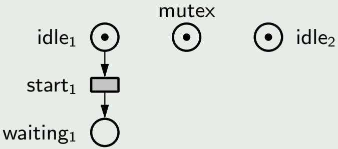


start$_1$, start$_2$, enter$_1$
start$_2$, start$_1$, enter$_1$

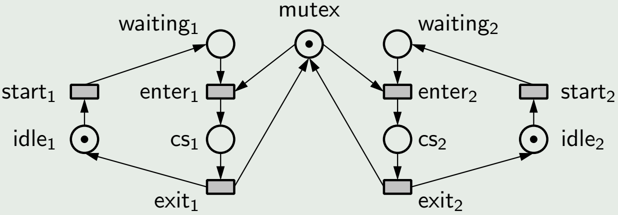
Processes (or configurations) of a Petri Net



start$_1$, start$_2$, enter$_1$
start$_2$, start$_1$, enter$_1$

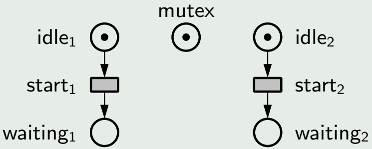


Processes (or configurations) of a Petri Net

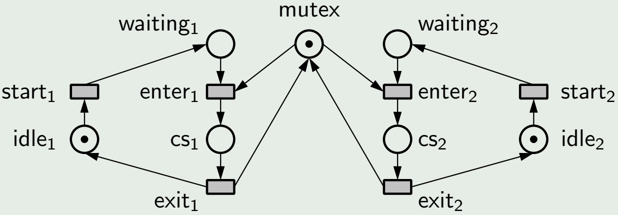


start$_1$, start$_2$, enter$_1$

start$_2$, start$_1$, enter$_1$

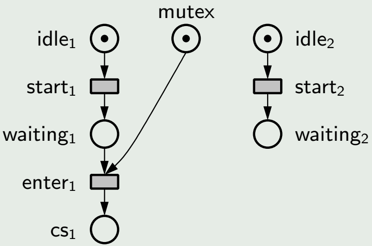


Processes (or configurations) of a Petri Net

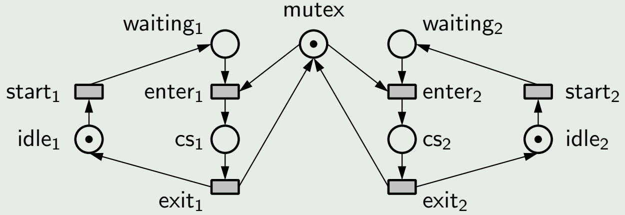


start$_1$, start$_2$, enter$_1$

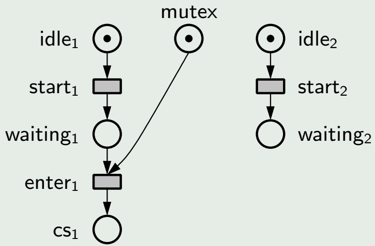
start$_2$, start$_1$, enter$_1$



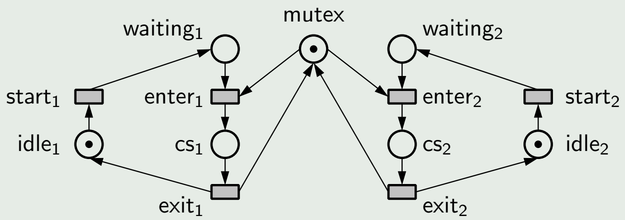
Processes (or configurations) of a Petri Net



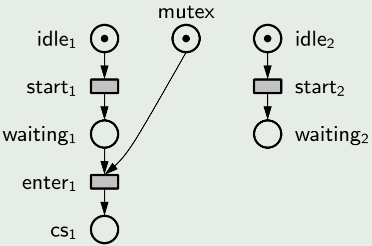
start$_1$, start$_2$, enter$_1$
start$_2$, start$_1$, enter$_1$
start$_1$, enter$_1$, start$_2$



Processes (or configurations) of a Petri Net

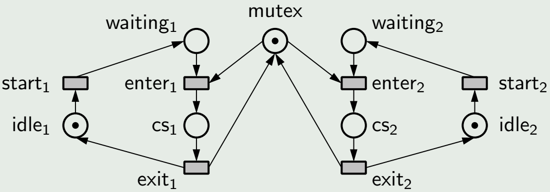


start$_1$, start$_2$, enter$_1$
start$_2$, start$_1$, enter$_1$
start$_1$, enter$_1$, start$_2$

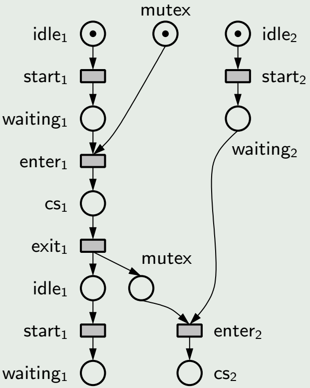


- Events and conditions
- Labelled, acyclic, and safe
- Represents multiple interleavings of the same concurrent behaviour

Structure of Processes

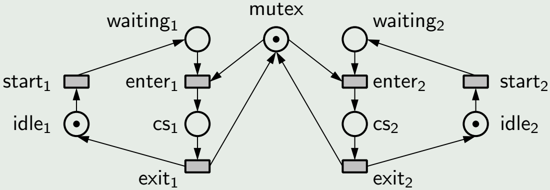


start$_1$, enter$_1$, exit$_1$, start$_2$, enter$_2$, start$_1$,

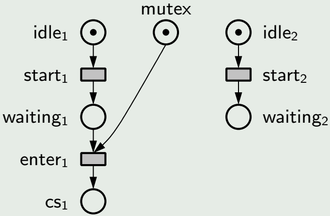


- Processes are acyclic, i.e., partial orders
- Associated to a (set of) run
- Every two events e, e' are either
  1. Concurrent, denoted $e \parallel e'$, as copies of start$_1$ and start$_2$
  2. Causally related, denoted $e < e'$, as start$_1$ and enter$_1$

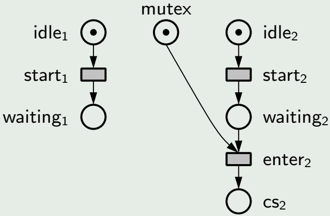
Petri nets — Non-sequential Semantics



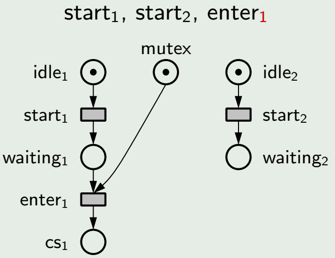
start$_1$, start$_2$, enter$_1$



start$_1$, start$_2$, enter$_2$



Petri nets — Non-sequential Semantics



Non-sequential Semantics

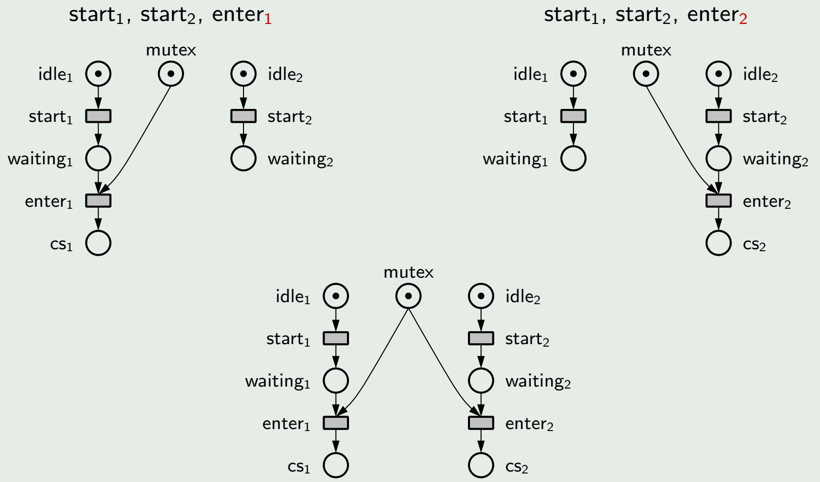
The non-sequential semantics of N is the set $conf(N)$ of all processes associated to the runs of N, i.e.,

$$conf(N) := \{\mathcal{C}_\sigma : \mathcal{C}_\sigma \text{ is the process of some } \sigma \in runs(N)\}$$

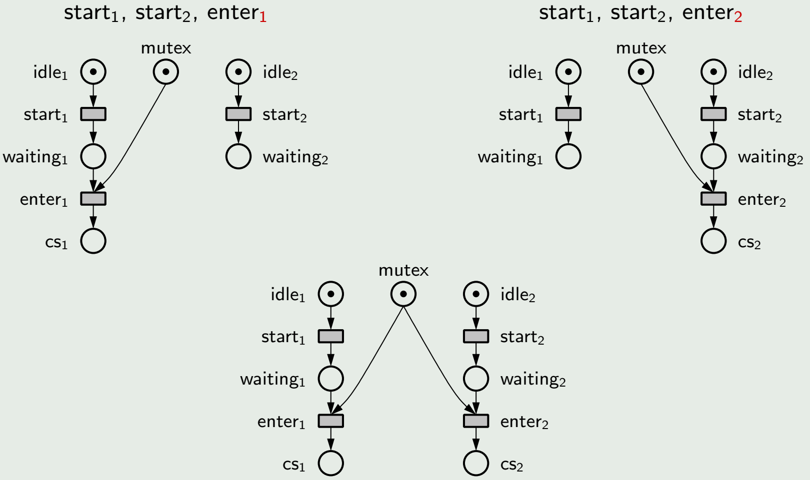
- Each process is a Mazurkiewicz trace or a labelled partial order or ...

Outline

What if we fuse common parts of multiple processes?

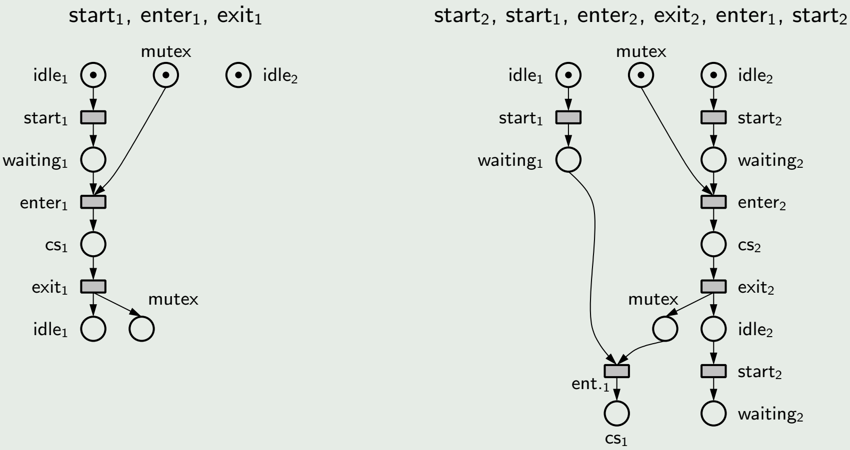


What if we fuse common parts of multiple processes?



- We get a branching process or unfolding prefix
- Events may now be in conflict, denoted by $e \# e'$, as enter_1 and enter_2

What if we fuse common parts of multiple processes?

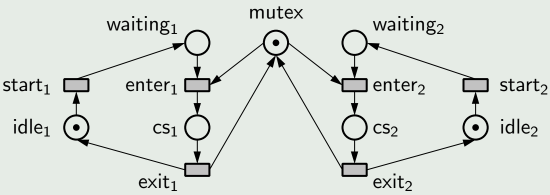


- We get a branching process or unfolding prefix
- Events may now be in conflict, denoted by $e \# e'$, as enter_1 and enter_2

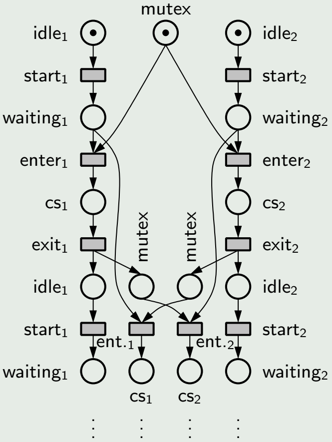
Petri Nets — Unfolding Semantics

Unfolding Semantics

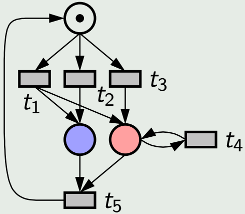
The unfolding $\mathcal{U}_N$ is the net that results from fusing together the common parts of all configurations in $conf(N)$.

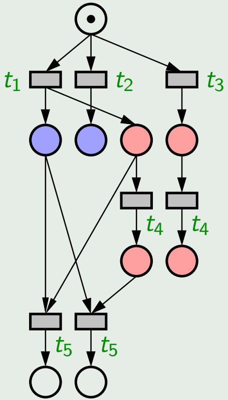


- Acyclic and safe
- Labelling is a homomorphism
- Infinite in general



Inductive Definition — Example

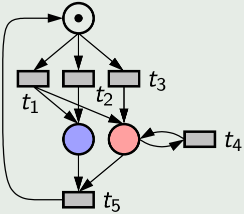




Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Labelling is a homomorphism
- Infinite in general
- Finite, complete unfolding prefix

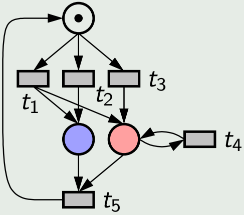
Inductive Definition — Example

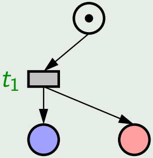


Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Labelling is a homomorphism
- Infinite in general
- Finite, complete unfolding prefix

Inductive Definition — Example

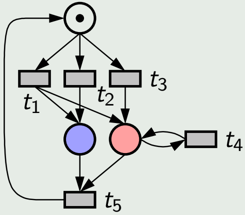


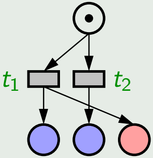


Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Labelling is a homomorphism
- Infinite in general
- Finite, complete unfolding prefix

Inductive Definition — Example

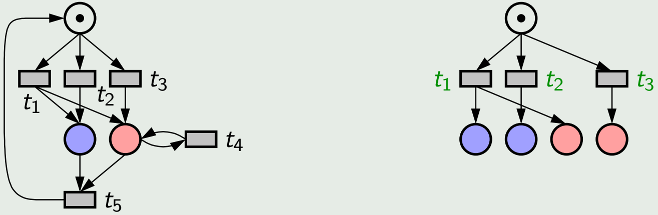




Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Labelling is a homomorphism
- Infinite in general
- Finite, complete unfolding prefix

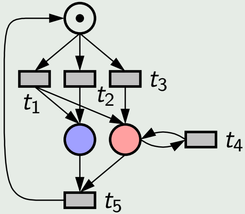
Inductive Definition — Example

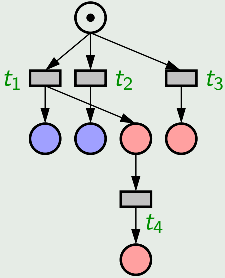


Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Labelling is a homomorphism
- Infinite in general
- Finite, complete unfolding prefix

Inductive Definition — Example

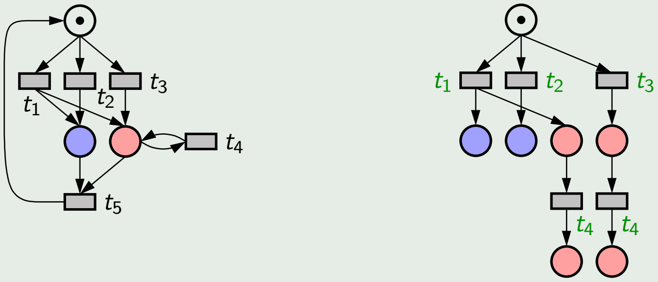




Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Labelling is a homomorphism
- Infinite in general
- Finite, complete unfolding prefix

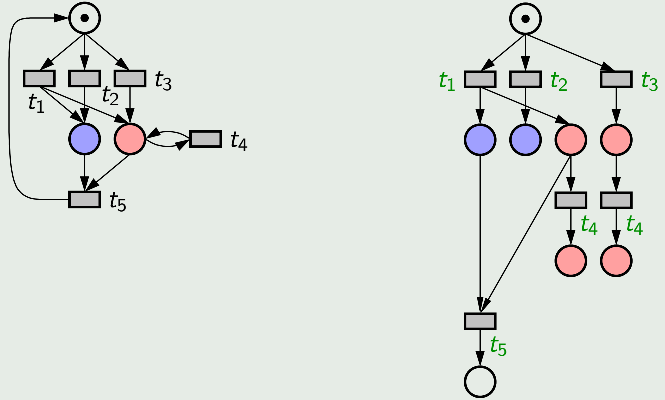
Inductive Definition — Example



Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Labelling is a homomorphism
- Infinite in general
- Finite, complete unfolding prefix

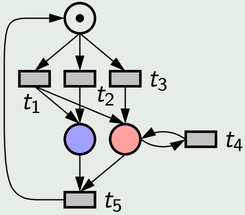
Inductive Definition — Example

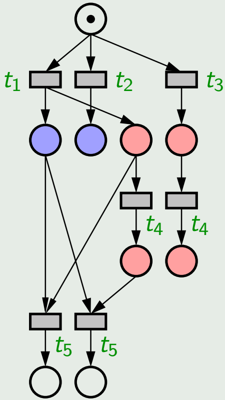


Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Labelling is a homomorphism
- Infinite in general
- Finite, complete unfolding prefix

Inductive Definition — Example





Remarks

- $\mathcal{U}_N$ is acyclic, 1-safe
- Labelling is a homomorphism
- Infinite in general
- Finite, complete unfolding prefix

Petri Nets — Unfolding Semantics (Inductive Definition)

Let $N := \langle P, T, F, m_0 \rangle$ be a safe Petri net. The unfolding

$$\mathcal{U}_N := \langle B, E, G, D, \tilde{m}_0 \rangle$$

is the safe, acyclic net defined by:

$$\frac{p \in m_0}{c = \langle \bot, p \rangle \in B \qquad h(c) = p \qquad c \in \tilde{m}_0}$$

$$\frac{t \in T \qquad X \subseteq B \qquad h(X) = {}^\bullet t \qquad X \text{ is coverable}}{e = \langle X, t \rangle \in E \qquad {}^\bullet e = X \qquad h(e) = t}$$

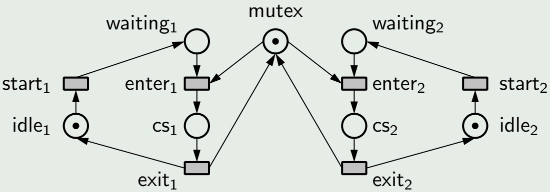
$$\frac{e \in E \qquad h(e) = t \qquad t^\bullet = \{p_1, \ldots, p_n\}}{c_i = \langle e, p_i \rangle \in B \qquad e^\bullet = \{c_1, \ldots, c_n\} \qquad h(c_i) = p_i}$$

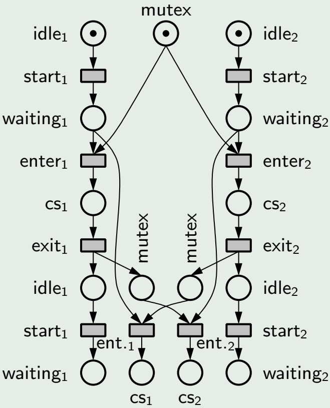
- h is a Petri net homomorphism.

Structural Relations

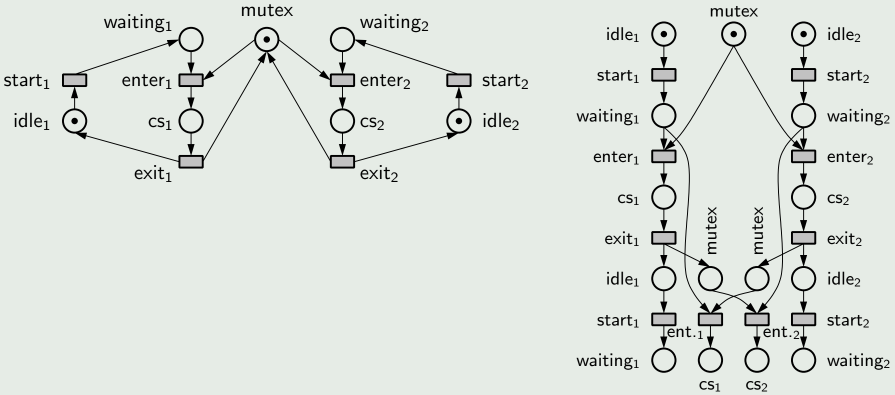
Definition

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



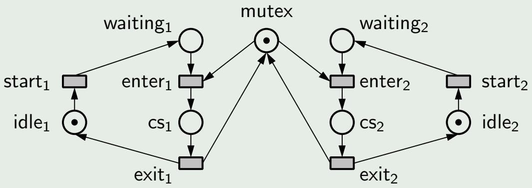
Configurations

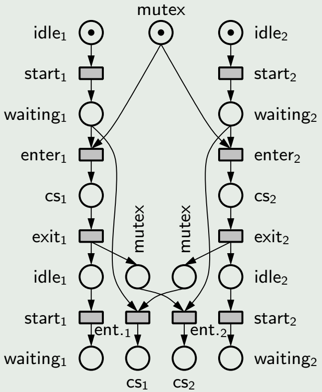
(Re)definition

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

1. $e \in \mathcal{C} \land e' < e \Rightarrow e' \in \mathcal{C}$ (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 sorted to form a run.





Outline

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:

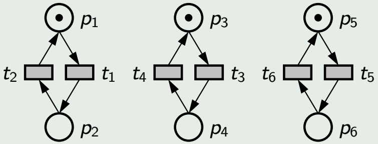
for all marking m reachable in N, there is marking $\tilde{m}$ reachable in $\mathcal{P}_N$ such that

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

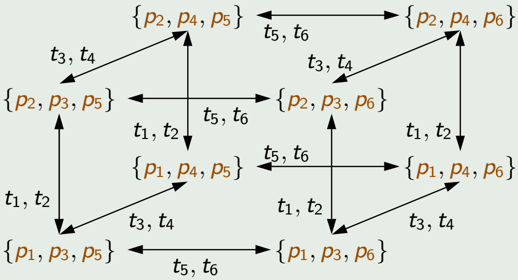
If N has finitely many reachable markings. . .

- 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

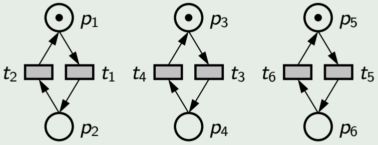
Unfoldings Cope with Concurrency

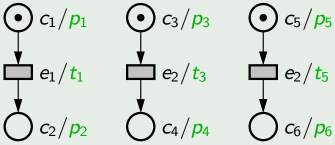


- 2^3 reachable markings
- And 2^n if n processes

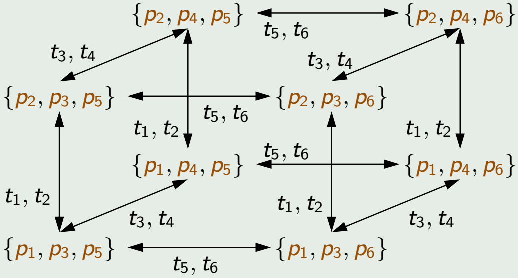


Unfoldings Cope with Concurrency





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



Cutoff Events

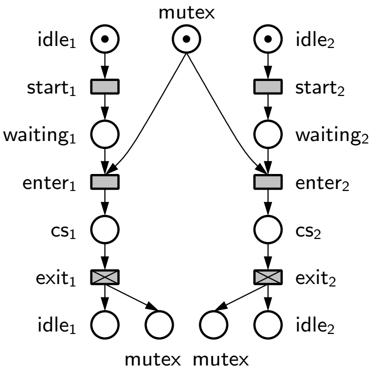
Pruning the unfolding

An event e is a cutoff if either there is an event e' such that

- $|[e']| < |[e]|$ and
- $mark([e]) = mark([e'])$.

Remarks

- Requires building prefixes breadth-first
- Cutoff criteria relates to completeness
- Proposed by McMillan; improved by Esparza et al., among others



Unfolding Analysis — Reachability

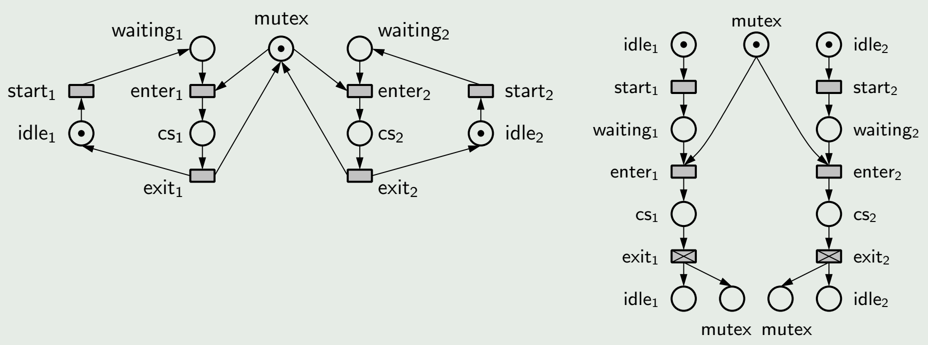
Let $\mathcal{P}_N$ be a complete unfolding prefix of N:

- The reachability problem in $\mathcal{P}_N$ can be solved in polynomial time
- Every reachable marking of $\mathcal{P}_N$ is labelled by a marking reachable in N
- And all markings of N are represented in $\mathcal{P}_N$

So given $\mathcal{P}_N$ and a marking m of N, checking whether m is reachable in N is NP-complete in $\mathcal{P}_N$

- Reductions to SAT, linear programming, stable models, ...
- Analysis time generally much smaller than unfolding time

Unfolding Analysis — Reachability



Given a set of places M of the net, generate

- $\phi^{\text{reach}, M}$ satisfiable iff places M reachable in N
- Encodes existence of a configuration (partially-ordered run) that marks M

Partial-order Reduction vs Unfoldings

	Partial-order reduction	**Unfoldings**
Underlying structure	Interleavings	Partial order
Idea	Discard equivalent states	Compress equivalent states
Cycles	Allowed	Unfolded
Independence	Static	Dynamic
Analysis	Linear time	NP-complete
Mainstream	✓	✗

cf. ongoing work with Subodh

Unfolding Other Models of Concurrency

Unfoldings applicable to other models of concurrency:

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

Unfolding Other Models of Concurrency

Unfoldings applicable to other models of concurrency:

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

cf. work with Bjoern and Subodh

Summary

- Compact representation of a finite, concurrent state spaces
- Structure, properties, and construction of unfoldings
- Reachability analysis: based on SAT or on-the-fly
- Applicable to other formalisms with notion of concurrency

Summary

- Compact representation of a finite, concurrent state spaces
- Structure, properties, and construction of unfoldings
- Reachability analysis: based on SAT or on-the-fly
- Applicable to other formalisms with notion of concurrency

Unfoldings do not address other sources of explosion:

- Non-deterministic choices ($\rightarrow$ merged processes)
- Concurrent read access ($\rightarrow$ contextual unfoldings)
- Non-safe or unbounded nets ($\rightarrow$ currently working on it)
- Data