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

- **System**: Model the system
  - **Modelling**: Formalization of the system model
  - **State-space exploration**: Generation of the Kripke structure $K$
  - **Kripke structure $K$**: Check whether $K \models \phi$
- **Property to verify**: Specification $\phi$
  - **Formalization**: Unfolding-based Reachability of Petri Nets

Counterexample / Correct
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
Check whether $K \models \phi$

Counterexample / Correct
Model Checking with Net Unfoldings

- **Concurrent system**
  - Modelling
  - Petri Net
  - Unfolding construction
  - Complete prefix
  - Unfolding analysis
  - Counterexample / Correct

- **Property to verify**
  - Formalization
  - Reachability / LTL
## 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:

Petri Nets — Example

- **Places** are represented by circles.
- **Transitions** are represented by rectangles.
- **Tokens** are represented by dots.
- **Arcs** are represented by arrows connecting places and transitions.

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 circles are places
The rectangles are transitions
The dots are tokens
The arrows are arcs

Allowed patterns:

Forbidden patterns:
Petri Nets — Example

The \( \bigcirc \) are places
The \( \square \) are transitions
The \( \bullet \) are tokens
The \( \rightarrow \) 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:
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
- $m_0: P \to \{0, 1\}$: initial marking

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

**Preset:**
\[ \cdot x := \{ y \in P \cup T : (y, x) \in F \} \]

**Postset:**
\[ x^\bullet := \{ y \in P \cup T : (x, y) \in F \} \]
A marking of $N$ is a function $m: P \rightarrow \mathbb{N}$ that maps places to the number of tokens they contain.

$m(idle_1) = 1$
$m(mux) = 1$
$m(idle_2) = 1$

$m(p) = 0$ for any other $p \in P$
A marking of $N$ is a function $m: P \rightarrow \mathbb{N}$ that maps places to the number of tokens they contain.

- $m$ (mutex) = 2
- $m$ (cs$_2$) = 3
- $m(p) = 0$ for any other $p \in P$
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.
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 \( \text{start}_1 \) and \( \text{start}_2 \) are enabled.
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 \( p \in P \).
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

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

for all $p \in P$. 

![Petri Net Diagram](image)
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 \( m: P \rightarrow \mathbb{N} \) of \( N \)
- \( \Delta := \{ \langle m, t, m' \rangle: \text{there is } t \in T \text{ such that } m \xrightarrow{t} m' \} \)
- \( s_0 := m_0 \), the initial marking of \( N \)
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 \( m: P \to \mathbb{N} \) of \( N \)
- \( \Delta := \{\langle m, t, m'\rangle: \) 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 \( \text{reach}(N) \) satisfying

1. \( m_0 \in \text{reach}(N) \)
2. If \( m \in \text{reach}(N) \) and \( m \xrightarrow{t} m' \), for any \( t \in T \), then \( m' \in \text{reach}(N) \).
Petri Nets — Operational Semantics: Example

\[ \text{mutex} \]

\[ \text{cs}_2 \]

\[ \text{exit}_2 \]

\[ \text{waiting}_2 \]

\[ \text{enter}_2 \]

\[ \text{start}_2 \]

\[ \text{idle}_2 \]

\[ \text{cs}_1 \]

\[ \text{exit}_1 \]

\[ \text{start}_1 \]

\[ \text{idle}_1 \]

\[ \text{waiting}_1 \]

\[ \text{enter}_1 \]
Petri Nets — Operational Semantics: Example
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 $\text{runs}(N)$. 
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 \( \text{reach}(N) \) so are.
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 $\text{reach}(N)$ so are.

The Petri net $N$ is bounded iff $\text{reach}(N)$ is finite.

- All nets we have seen so far were safe
- For the rest of the talk, we focus on bounded Petri nets
### Reachability Problem
- Given: a net \( N \) and a marking \( m \)
- Decide: if \( m \in \text{reach}(N) \)

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

### Boundedness Problem
- Given: a net \( N \)
- Decide: whether \( \text{reach}(N) \) is finite, i.e., whether \( N \) is bounded
## Petri Nets — Decidability and Complexity

<table>
<thead>
<tr>
<th></th>
<th>Bounded net</th>
<th>Unbounded net</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reachability</strong></td>
<td>PSPACE-complete</td>
<td>EXPSPACE-hard</td>
</tr>
<tr>
<td><strong>Coverability</strong></td>
<td>PSPACE-complete</td>
<td>EXSPACE-complete</td>
</tr>
<tr>
<td><strong>LTL model checking</strong></td>
<td>PSPACE-complete</td>
<td>Undecidable</td>
</tr>
<tr>
<td><strong>Boundedness</strong></td>
<td>N/A</td>
<td>EXPSPACE-complete</td>
</tr>
</tbody>
</table>
Communicating Automata

\[ a \rightarrow b \rightarrow c \rightarrow d \]

\[ c \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow c \rightarrow d \]

\[ c \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]

\[ a \rightarrow b \rightarrow e \rightarrow f \]
Concurrent Boolean Programs

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

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

\[ a := 1 \]
\[ \sim a \]
\[ b := 0 \]
\[ \sim b \]
\[ \text{while } a \]

\[ \text{goto L0} \]
Counter Abstractions

\[ x_1, x_2 \]

\( x_1 := 0 \)
\( x_2 := 0 \)
\[ x_1 = 0 \]
\( x_1 := \neg x_1 \)
\( x_2 := 1 \)
\[ x_2 = 1 \]
\[ x_1 = 0 \]

Shared variables (finite state)

Program (finite state, unbounded replication)
Counter Abstractions
Counter Abstractions

\[ x_1, \neg x_1, x_2, \neg x_2 \]
Outline

1 Petri Nets

2 Non-sequential Semantics

3 Unfolding Semantics

4 Finite, Complete Prefixes

5 Summary
State-Explosion: Concurrency

2^3 reachable markings

\[\begin{align*}
\text{p}_1 & \xleftarrow{t_2} \text{p}_2 \\
\text{p}_3 & \xleftarrow{t_4} \text{p}_4 \\
\text{p}_5 & \xleftarrow{t_6} \text{p}_6
\end{align*}\]
State-Explosion: Concurrency

- $2^3$ reachable markings
- And $2^n$ if $n$ processes instead of 3
Processes (or configurations) of a Petri Net

![Petri Net Diagram]

- start₁, start₂, enter₁
- start₂, start₁, enter₁
Processes (or configurations) of a Petri Net

Labels and conditions

Labelled, acyclic, and safe

Represents multiple interleavings of the same concurrent behaviour

\[ \text{start}_1, \text{start}_2, \text{enter}_1 \]

\[ \text{start}_2, \text{start}_1, \text{enter}_1 \]

\[ \text{idle}_1, \text{mutex}, \text{idle}_2 \]
Processes (or configurations) of a Petri Net

![Petri Net Diagram]

- **start**\(_1\), **start**\(_2\), **enter**\(_1\)
- **start**\(_2\), **start**\(_1\), **enter**\(_1\)

**Events and conditions**

- Labelled, acyclic, and safe
- Represents multiple interleavings of the same concurrent behaviour
Processes (or configurations) of a Petri Net

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

César Rodríguez (Oxford)
Processes (or configurations) of a Petri Net

\[ \text{start}_1, \text{start}_2, \text{enter}_1, \text{start}_2, \text{start}_1, \text{enter}_1 \]
Processes (or configurations) of a Petri Net

\begin{align*}
\text{start}_1, \text{start}_2, \text{enter}_1, \\
\text{start}_2, \text{start}_1, \text{enter}_1, \\
\text{start}_1, \text{enter}_1, \text{start}_2
\end{align*}
Processes (or configurations) of a Petri Net

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

start₁, start₂, enter₁
start₂, start₁, enter₁
start₁, enter₁, start₂
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$_1$
Non-sequential Semantics

The non-sequential semantics of $N$ is the set $\text{conf}(N)$ of all processes associated to the runs of $N$, i.e., $\text{conf}(N) := \{ C_\sigma : C_\sigma$ is the process of some $\sigma \in \text{runs}(N) \}$.
The non-sequential semantics of $N$ is the set $\text{conf}(N)$ of all processes associated to the runs of $N$, i.e.,

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

- Each process is a Mazurkiewicz trace or a labelled partial order or...
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 \( e \not\equiv e' \), as \( \text{enter}_1 \) and \( \text{enter}_2 \).
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 $e \not\equiv e'$, as enter$_1$ and enter$_2$
The unfolding $\mathcal{U}_N$ is the net that results from fusing together the common parts of all configurations in $\text{conf}(\mathcal{N})$.

- 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
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
Remarks

- $U_N$ 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
Remarks

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

\[ p \in m_0 \]
\[ c = \langle \bot, p \rangle \in B \quad h(c) = p \quad c \in \tilde{m}_0 \]

\[ t \in T \quad X \subseteq B \quad h(X) = \bullet t \quad X \text{ is coverable} \]
\[ e = \langle X, t \rangle \in E \quad \bullet e = X \quad h(e) = t \]

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

- \( h \) 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

A set of events $C$ is a configuration iff:

1. $e, e' \in C$ and $e' < e$ implies $e' \in C$ (causally closed)
2. $\neg e \# e'$ for all $e, e' \in C$ (conflict free)

Intuition: $C$ configuration iff all its events can be sorted to form a run.
A set of events $C$ is a configuration iff:

1. $e \in C \land e' < e \Rightarrow e' \in C$ (causally closed)
2. $\neg e \neq e'$ for all $e, e' \in C$ (conflict free)

Intuition: $C$ configuration iff all its events can be sorted to form a run.
Outline

1. Petri Nets
2. Non-sequential Semantics
3. Unfolding Semantics
4. Finite, Complete Prefixes
5. Summary
\( \mathcal{U}_N \) is the result of unfolding ‘as much as possible’

Finite unfolding prefix \( \mathcal{P}_N \) results if you stop construction
\( U_N \) 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

- $U_N$ is the result of unfolding ‘as much as possible’
- Finite unfolding prefix $P_N$ results if you stop construction

**Definition**

Prefix $P_N$ is marking-complete if:
for all marking $m$ reachable in $N$, there is marking $\tilde{m}$ reachable in $P_N$ such that

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

If $N$ has finitely many reachable markings...
\( \mathcal{U}_N \) 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
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
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
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

<table>
<thead>
<tr>
<th></th>
<th>Partial-order reduction</th>
<th>Unfoldings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Underlying structure</strong></td>
<td>Interleavings</td>
<td>Partial order</td>
</tr>
<tr>
<td><strong>Idea</strong></td>
<td>Discard equivalent states</td>
<td>Compress equivalent states</td>
</tr>
<tr>
<td><strong>Cycles</strong></td>
<td>Allowed</td>
<td>Unfolded</td>
</tr>
<tr>
<td><strong>Independence</strong></td>
<td>Static</td>
<td>Dynamic</td>
</tr>
<tr>
<td><strong>Analysis</strong></td>
<td>Linear time</td>
<td>NP-complete</td>
</tr>
<tr>
<td><strong>Mainstream</strong></td>
<td>✔</td>
<td>✗</td>
</tr>
</tbody>
</table>

cf. ongoing work with Subodh
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
- ...
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

Unfoldings do not address other sources of explosion:
- Non-deterministic choices (→ merged processes)
- Concurrent read access (→ contextual unfoldings)
- Non-safe or unbounded nets (→ currently working on it)
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 (→ merged processes)
- Concurrent read access (→ contextual unfoldings)
- Non-safe or unbounded nets (→ currently working on it)
- Data