Using Unfoldings of Petri Nets for Verification of Systems

Dominik Drexler

27.01.2018
Introduction to System Verification
Given a system (e.g. software system). Most prominent question:
Given a system (e.g. software system). Most prominent question:
- Does the system never reach a deadlock? (deadlock checking)
Given a system (e.g. software system). Most prominent question:
- Does the system never reach a deadlock? (deadlock checking)
- Traditional approach: Create Transition System and do a state space search
Example

- Given \( n \) processes all doing the same following task:

```c
bool x = false;
while true do
  x = true;  # \( t_i \)
  wait(random)
  x = false;  # \( \overline{t_i} \)
  wait(random)
end while
```
Example

- Given \( n \) processes all doing the same following task:

```c
bool x = false;
while true do
    x = true;    \# \( t_i \)
    wait(random)
    x = false;    \# \( \bar{t}_i \)
    wait(random)
end while
```

- Each state represented by the set of processes where \( x = \) true
Example

- Given \( n \) processes all doing the same following task:

```
bool x = false;
while true do
    x = true;  // \( t_i \)
    wait(random)
    x = false;  // \( \overline{t}_i \)
    wait(random
end while
```

- Each state represented by the set of processes where \( x = \text{true} \)
Example

- Given \( n \) processes all doing the same following task:

```plaintext
bool x = false;
while true do
    x = true;  # \( t_i \)
    wait(random)
    x = false;  # \( \bar{t}_i \)
    wait(random)
end while
```

Each state represented by the set of processes where \( x = \text{true} \)

In this example \( |P(\{p_1, \ldots, p_n\})| = 2^n \) states.
Example

- Given $n$ processes all doing the same following task:

```plaintext
bool x = false;
while true do
    x = true; # $t_i$
    wait(random)
    x = false; # $\bar{t}_i$
    wait(random)
end while
```

- Each state represented by the set of processes where $x = true$
- In this example $|\mathcal{P}\{p_1, \ldots, p_n\}| = 2^n$ states.
- Searching for deadlock: Find state with no outgoing transitions which is no terminal state.
Example

Given \( n \) processes all doing the same following task:

```java
bool x = false;
while true do
  x = true;  # \( t_i \)
  wait(random)
  x = false;  # \( \overline{t}_i \)
  wait(random)
end while
```

**Figure:** Petri net

Alternative approach: Petri nets to model concurrency

How to check for deadlocks?
Cycles in Petri Nets are problematic
Alternative approach: Petri nets to model concurrency

Example

Given \( n \) processes all doing the same following task:

```
bool x = false;
while true do
    x = true;  # \( t_i \)
    wait(random)
    x = false;  # \( \bar{t}_i \)
    wait(random)
end while
```

- Size is linear in \( n \)
Alternative approach: Petri nets to model concurrency

### Example

Given $n$ processes all doing the same following task:

```plaintext
bool x = false;
while true do
    x = true; # $t_i$
    wait(random)
    x = false; # $\bar{t}_i$
    wait(random)
end while
```

- Size is linear in $n$
- How to check for deadlocks?

**Figure: Petri net**
Alternative approach: Petri nets to model concurrency

Example

Given $n$ processes all doing the same following task:

```c
bool x = false;
while true do
    x = true;  # $t_i$
    wait(random)
    x = false;  # $\bar{t}_i$
    wait(random)
end while
```

- Size is linear in $n$
- How to check for deadlocks?
- Cycles in Petri Nets are problematic

Example

Figure: Petri net
An Unfolding of a Petri Net is a Branching Process where:

1. All reachable markings are present.
2. All transitions enabled by a marking are present.
Definition: Unfolding

An Unfolding of a Petri Net is a Branching Process where
1. All reachable markings are present
2. All transitions enabled by a marking are present

A Branching Process is an Occurrence Net with a labelling function
An Unfolding of a Petri Net is a Branching Process where
1. All reachable markings are present
2. All transitions enabled by a marking are present

A Branching Process is an Occurrence Net with a labelling function

An Occurrence Net is a net with a simpler structure
An Unfolding of a Petri Net is a Branching Process where
1. All reachable markings are present
2. All transitions enabled by a marking are present

A Branching Process is an Occurrence Net with a labelling function

An Occurrence Net is a net with a simpler structure

The labelling functions assigns each node in the occurrence net a label of the original net
Intuitively: Occurrence Nets can be seen as 1-safe Nets.

**Definition**

An Occurrence Net is a net $O = (B, E, F)$ with the following properties:

1. $|\bullet b| \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No event $e \in E$ is in conflict with itself (no backward conflicts)
Intuitively: Occurence Nets can be seen as 1-safe Nets.
Initially one token at each Condition of $Min(O) = \{ b \in B | 0 = |\bullet b| \}$

Definition

An Occurence Net is a net $O = (B, E, F)$ with the following properties:

1. $|\bullet b| \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No event $e \in E$ is in conflict with itself (no backward conflicts)
Intuitively: Occurrence Nets can be seen as 1-safe Nets.

Initially one token at each Condition of \( \text{Min}(O) = \{ b \in B | 0 = |\bullet b| \} \)

**Definition**

An Occurrence Net is a net \( O = (B, E, F) \) with the following properties:

1. \( |\bullet b| \leq 1 \) for all \( b \in B \)
2. \( O \) is acyclic
3. Every \( x \in B \cup E \) has finitely many predecessors
4. No event \( e \in E \) is in conflict with itself (no backward conflicts)

**Example (Property 1)**

![Counterexample](image1)

![Example](image2)

**Figure: Counterexample**

**Figure: Example**
Intuitively: Occurrence Nets can be seen as 1-safe Nets.

Initially one token at each Condition of $Min(O) = \{ b \in B | 0 = \bullet b \}$

**Definition**

An Occurrence Net is a net $O = (B, E, F)$ with the following properties:

1. $|\bullet b| \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No event $e \in E$ is in conflict with itself (no backward conflicts)

**Example (Property 1)**

*Figure: Counterexample*

*Figure: Example*
Definition of Occurrence Nets (1)

- Intuitively: Occurrence Nets can be seen as 1-safe Nets.
- Initially one token at each Condition of $\text{Min}(O) = \{b \in B | 0 = |\bullet b|\}$

**Definition**

An Occurrence Net is a net $O = (B, E, F)$ with the following properties:

1. $|\bullet b| \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No event $e \in E$ is in conflict with itself (no backward conflicts)

**Example (Property 1)**

- Figure: Counterexample
- Figure: Example
Definition of Occurrence Nets (1)

- Intuitively: Occurrence Nets can be seen as 1-safe Nets.
- Initially one token at each Condition of $Min(O) = \{b \in B | 0 = |\bullet b|\}$

Definition

An Occurrence Net is a net $O = (B, E, F)$ with the following properties:
1. $|\bullet b| \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No event $e \in E$ is in conflict with itself (no backward conflicts)

Example (Property 1)

![Counterexample](image1)

![Example](image2)
Definition of Occurrence Nets (2)

An Occurrence Net is a net $O = (B, E, F)$ with the following properties:

1. $\vert \bullet b \vert \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No event $e \in E$ is in conflict with itself (no backward conflicts)

Example (Property 2)

Figure: Counterexample
Definition of Occurrence Nets (3)

Definition

An Occurrence Net is a net \( O = (B, E, F) \) with the following properties:

1. \(|\bullet b| \leq 1\) for all \( b \in B \)
2. \(O\) is acyclic
3. Every \( x \in B \cup E\) has finitely many predecessors
4. No event \( e \in E\) is in conflict with itself (no backward conflicts)
Definition

An Occurrence Net is a net $O = (B, E, F)$ with the following properties:

1. $|\bullet b| \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No event $e \in E$ is in conflict with itself (no backward conflicts)

Definition

$x$ is in conflict with $y$ denoted by $x \# y$ iff there exists two paths $p_1 = b, e_1, \ldots, x$ and $p_2 = b, e_2, \ldots, y$ for $b \in B$ and $e_1 \neq e_2 \in E$.

Example (Property 4)

- Either $e_1$ or $e_2$ can fire
- $\Rightarrow$ Either $b_2$ or $b_3$ receives a token
- $\Rightarrow$ $e_3$ cannot be fired at any time
- $\Rightarrow$ makes no sense to allow such thing

Figure: Counterexample
Configurations describe the possible events that can be fired in an Occurrence Net.
Configurations describe the possible events that can be fired in an Occurrence Net.

**Definition**

A set of events is a configuration $C$ iff the following properties are satisfied:

1. $e \in C \Rightarrow \forall e' < e : e' \in C$
2. $\forall e, e' : e, e' \in C : \neg(e \# e')$
Configurations describe the possible events that can be fired in an Occurrence Net.

**Definition**

A set of events is a configuration $C$ iff the following properties are satisfied:

1. $e \in C \Rightarrow \forall e' : e' \in C$
2. $\forall e, e' \in C : \neg (e \# e')$

**Example**

- $C_0 = \emptyset$

---

Verification using net unfoldings
• Configurations describe the possible events that can be fired in an Occurrence Net

**Definition**

A set of events is a configuration \( C \) iff the following properties are satisfied:

1. \( e \in C \Rightarrow \forall e' < e : e' \in C \)
2. \( \forall e, e' \in C : \neg(e \# e') \)

**Example**

- \( C_0 = \emptyset \)
- \( C_1 = \{e_2, e_3\} \)
Configurations describe the possible events that can be fired in an Occurrence Net.

**Definition**

A set of events is a configuration $C$ iff the following properties are satisfied:

1. $e \in C \Rightarrow \forall e' < e : e' \in C$
2. $\forall e, e' \in C : \neg (e \# e')$

**Example**

- $C_0 = \emptyset$
- $C_1 = \{e_2, e_3\}$
- $\{e_4\}$ is not a configuration!
Configurations describe the possible events that can be fired in an Occurrence Net.

**Definition**

A set of events is a configuration $C$ iff the following properties are satisfied:

1. $e \in C \Rightarrow \forall e' < e : e' \in C$
2. $\forall e, e' \in C : \neg (e \# e')$

**Example**

- $C_0 = \emptyset$
- $C_1 = \{e_2, e_3\}$
- $\{e_4\}$ is not a configuration!
- $\{e_1, e_2\}$ is not a configuration!
Definition: Branching Process (1)

Let $N = (P, T, W, M_0)$ be a Petri net. A Branching Process is a labelled occurrence net $\beta = (O, p) = (B, E, F, p)$ where $p$ is the labelling function with the following properties:

1. $p(B) \subseteq P$ and $P(E) \subseteq T$
2. For every $e \in E$, the restriction of $p$ to $\bullet e$ is a bijection between $\bullet e$ (in $\beta$) and $\bullet p(e)$ (in $N$)
3. The restriction of $p$ to $\text{Min}(O) := \{b \in B | 0 = |\bullet b|\}$ is a bijection between $\text{Min}(O)$ and $M_0$
4. For every $e_1, e_2 \in E$ if $\bullet e_1 = \bullet e_2$ and $p(e_1) = p(e_2)$ then $e_1 = e_2$

Example (Property 1: Preservation of the nature of nodes)

\begin{figure}
\centering
\begin{tikzpicture}
  \node[state, initial] (s1) {$s_1$};
  \node[place] (b1) [below of=s1] {$b_1$};
  \node[place] (b2) [below left of=s1] {$b_2$};
  \node[place] (b3) [below right of=s1] {$b_3$};
  \node[place] (s2) [below of=s1] {$s_2$};

  \path[->] (s1) edge (b1)
  (s1) edge (b2)
  (s1) edge (b3)
  (b1) edge (t1)
  (b2) edge (t1)
  (b3) edge (t1)
  (s2) edge (t2)
  (b1) edge (s2)
  (b2) edge (s2)
  (b3) edge (s2);
\end{tikzpicture}
\caption{$N = (P, T, W, M_0)$}
\end{figure}

\begin{figure}
\centering
\begin{tikzpicture}
  \node[state, initial] (s1) {$s_1$};
  \node[place, below of=s1] (b1) {$b_1$};
  \node[place, left of=s1] (b2) {$b_2$};
  \node[place, right of=s1] (b3) {$b_3$};
  \node[place] (s2) [below of=s1] {$s_2$};

  \path[->] (s1) edge (b1)
  (s1) edge (b2)
  (s1) edge (b3)
  (b1) edge (t1)
  (b2) edge (t1)
  (b3) edge (t1)
  (s2) edge (t2)
  (b1) edge (s2)
  (b2) edge (s2)
  (b3) edge (s2);
\end{tikzpicture}
\caption{$\beta = (B, E, F, p)$}
\end{figure}
Definition

Let $\mathcal{N} = (P, T, W, M_0)$ be a Petri net. A Branching Process is a labelled occurrence net $\beta = (O, p) = (B, E, F, p)$ where $p$ is the labelling function with the following properties:

1. $p(B) \subseteq P$ and $P(E) \subseteq T$
2. For every $e \in E$, the restriction of $p$ to $\bullet e$ is a bijection between $\bullet e$ (in $\beta$) and $\bullet p(e)$ (in $\mathcal{N}$)
3. The restriction of $p$ to $\text{Min}(O) := \{b \in B \mid 0 = |\bullet b|\}$ is a bijection between $\text{Min}(O)$ and $M_0$
4. For every $e_1, e_2 \in E$ if $\bullet e_1 = \bullet e_2$ and $p(e_1) = p(e_2)$ then $e_1 = e_2$

Example (Property 2: Preservation of transition environment)

Figure: $\mathcal{N} = (P, T, W, M_0)$

- $p(e_1) = t_1$
- $p(\bullet e_1) = p(\{b_1\}) = \{s_1\} = \bullet t_1$
- $p(e_1^\bullet) = p(\{b_2\}) = \{s_2\} = t_1^\bullet$

Figure: $\beta = (B, E, F, p)$
Definition: Branching Process (3)

Let $\mathcal{N} = (P, T, W, M_0)$ be a Petri net. A Branching Process is a labelled occurrence net $\beta = (O, p) = (B, E, F, p)$ where $p$ is the labelling function with the following properties:

1. $p(B) \subseteq P$ and $P(E) \subseteq T$
2. For every $e \in E$, the restriction of $p$ to $\cdot e$ is a bijection between $\cdot e$ (in $\beta$) and $\cdot p(e)$ (in $\mathcal{N}$)
3. The restriction of $p$ to $\text{Min}(O) := \{ b \in B | 0 = |\cdot b| \}$ is a bijection between $\text{Min}(O)$ and $M_0$
4. For every $e_1, e_2 \in E$ if $\cdot e_1 = \cdot e_2$ and $p(e_1) = p(e_2)$ then $e_1 = e_2$

Example (Property 3: $\beta$ starts at $M_0$)

- $M_0 = \{s_1, s_1, s_2\}$
- $\beta$ has three minimal nodes $b_1, b_2, b_3$ without incoming edges.
- $p(b_1) = s_1$, $p(b_2) = s_1$ and $p(b_3) = s_2$
Definition: Branching Process (4)

Let $\mathcal{N} = (P, T, W, M_0)$ be a Petri net. A Branching Process is a labelled occurrence net $\beta = (O, p) = (B, E, F, p)$ where $p$ is the labelling function with the following properties:

1. $p(B) \subseteq P$ and $P(E) \subseteq T$
2. For every $e \in E$, the restriction of $p$ to $\cdot e$ is a bijection between $\cdot e$ (in $\beta$) and $\cdot p(e)$ (in $\mathcal{N}$)
3. The restriction of $p$ to $Min(O) := \{b \in B | 0 = |\cdot b|\}$ is a bijection between $Min(O)$ and $M_0$
4. For every $e_1, e_2 \in E$ if $\cdot e_1 = \cdot e_2$ and $p(e_1) = p(e_2)$ then $e_1 = e_2$

Example (Property 4: $\beta$ does not duplicate transitions)

Figure: Counterexample
What is the marking reached by a configuration?

**Definition**

A set of events is a configuration $C$ iff the following properties are satisfied:

1. $e \in C \Rightarrow \forall e' \leq e : e' \in C$
2. $\forall e, e' \in C : \neg(e \# e')$
What is the marking reached by a configuration?

**Definition**

A set of events is a configuration $C$ iff the following properties are satisfied:

1. $e \in C \Rightarrow \forall e' \leq e : e' \in C$
2. $\forall e, e' \in C : \neg(e \neq e')$

$\text{Mark}(C)$ denotes the marking reached by firing all events in $C$. 
What is the marking reached by a configuration?

**Definition**

A set of events is a configuration $C$ iff the following properties are satisfied:

1. $e \in C \Rightarrow \forall e' \leq e : e' \in C$
2. $\forall e, e' \in C : \neg(e \# e')$

$\text{Mark}(C)$ denotes the marking reached by firing all events in $C$.

**Example**

$\text{Mark}(\emptyset) = p(\text{Min}(O)) = p(\{b_1, b_2\}) = \{s_1, s_2\}$
What is the marking reached by a configuration?

**Definition**

A set of events is a configuration $C$ iff the following properties are satisfied:

1. $e \in C \Rightarrow \forall e' \leq e : e' \in C$
2. $\forall e, e' \in C : \neg(e \neq e')$

$\text{Mark}(C)$ denotes the marking reached by firing all events in $C$.

**Example**

- $\text{Mark}(\emptyset) = p(\text{Min}(O)) = p\{b_1, b_2\} = \{s_1, s_2\}$
- $\text{Mark}\{e_2, e_3\} = p\{b_2, b_5, b_6, b_7\} = \{s_2, s_4, s_5, s_5\}$
Example: Mutual Exclusion Model

Example

- Given: The Petri Net \( \mathcal{N} \) which models mutual exclusion

![Petri Net \( \mathcal{N} \)](image)

**Figure:** Petri Net \( \mathcal{N} \)
Example: Mutual Exclusion Model

- Given: The Petri Net $\mathcal{N}$ which models mutual exclusion

- Goal: Compute Unfolding of $\mathcal{N}$
Example: Mutual Exclusion Model

Example

- Given: The Petri Net $\mathcal{N}$ which models mutual exclusion

![Petri Net $\mathcal{N}$](image)

**Figure**: Petri Net $\mathcal{N}$

- Goal: Compute Unfolding of $\mathcal{N}$
- Unfolding is a Branching Process where:
  1. All reachable markings are present
  2. All transitions enabled by a marking are present
Definition (Occurrence Net $O = (B, E, F)$)

1. $|\bullet b| \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No backward conflicts

Definition (Labelling function $p$)

1. $p(B) \subseteq P$ and $p(E) \subseteq T$
2. Preserve environment of transitions
3. Minimal conditions correspond to $M_0$
4. No duplicate transitions

Example: Unfolding of Mutual Exclusion Model

Figure: Petri Net $\mathcal{N}$

Figure: Current State while unfolding

Verification using net unfoldings
Definition (Occurrence Net $O = (B, E, F)$)

1. $|\bullet b| \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No backward conflicts

Definition (Labelling function $p$)

1. $p(B) \subseteq P$ and $p(E) \subseteq T$
2. Preserve environment of transitions
3. Minimal conditions correspond to $M_0$
4. No duplicate transitions

Example

Figure: Petri Net $\mathcal{N}$

Figure: Current State while unfolding
**Definition (Occurrence Net)**

\[ O = (B, E, F) \]

1. \(|\cdot b| \leq 1 \) for all \( b \in B \)
2. \( O \) is acyclic
3. Every \( x \in B \cup E \) has finitely many predecessors
4. No backward conflicts

**Definition (Labelling function \( p \))**

1. \( p(B) \subseteq P \) and \( p(E) \subseteq T \)
2. Preserve environment of transitions
3. Minimal conditions correspond to \( M_0 \)
4. No duplicate transitions

**Example:**

- \( t_{P1Enter} \) to \( s_{P1Wait} \)
- \( t_{P1Leave} \) to \( s_{P1Crit} \)
- \( s_{Mutex} \)
- \( t_{P2Enter} \) to \( s_{P2Wait} \)
- \( b_1 \) to \( s_{P1W} \)
- \( b_2 \) to \( s_{M} \)
- \( b_3 \) to \( s_{P2W} \)
- \( b_4 \) to \( s_{P1C} \)
- \( b_5 \) to \( s_{P2C} \)

**Figure:** Petri Net \( \mathcal{N} \)

**Figure:** Current State while unfolding
Example: Unfolding of Mutual Exclusion Model

Definition (Occurrence Net $O = (B, E, F)$)

1. $|\bullet b| \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No backward conflicts

Definition (Labelling function $p$)

1. $p(B) \subseteq P$ and $p(E) \subseteq T$
2. Preserve environment of transitions
3. Minimal conditions correspond to $M_0$
4. No duplicate transitions

Example

Figure: Petri Net $\mathcal{N}$

Figure: Current State while unfolding
Example: Unfolding of Mutual Exclusion Model

**Definition (Occurrence Net)**

\[ O = (B, E, F) \]

1. \(|\bullet b| \leq 1\) for all \( b \in B \)
2. \( O \) is acyclic
3. Every \( x \in B \cup E \) has finitely many predecessors
4. No backward conflicts

**Definition (Labelling function \( p \))**

1. \( p(B) \subseteq P \) and \( p(E) \subseteq T \)
2. Preserve environment of transitions
3. Minimal conditions correspond to \( M_0 \)
4. No duplicate transitions

**Example**

**Figure: Petri Net \( \mathcal{N} \)**

**Figure: Current State while unfolding**
Example: Unfolding of Mutual Exclusion Model

**Definition (Occurrence Net)**

\[ O = (B, E, F) \]

1. \(|b| \leq 1 \) for all \( b \in B \)
2. \( O \) is acyclic
3. Every \( x \in B \cup E \) has finitely many predecessors
4. No backward conflicts

**Definition (Labelling function \( p \))**

1. \( p(B) \subseteq P \) and \( p(E) \subseteq T \)
2. Preserve environment of transitions
3. Minimal conditions correspond to \( M_0 \)
4. No duplicate transitions

Figure: Petri Net \( N \)
Example: Unfolding of Mutual Exclusion Model

Definition (Occurrence Net $O = (B, E, F)$)

1. $|b| \leq 1$ for all $b \in B$
2. $O$ is acyclic
3. Every $x \in B \cup E$ has finitely many predecessors
4. No backward conflicts

Definition (Labelling function $p$)

1. $p(B) \subseteq P$ and $p(E) \subseteq T$
2. Preserve environment of transitions
3. Minimal conditions correspond to $M_0$
4. No duplicate transitions

Example

Figure: Petri Net $\mathcal{N}$
Definition (Deadlock: Petri Net vs Unfolding)

Let $\mathcal{N}$ be a Petri Net and $\beta$ its Unfolding. $\mathcal{N}$ has a deadlock if there exists a reachable marking $M$ which is no terminal marking and no transition is enabled.

$\iff$ There exists a configuration $C$ in $\beta$ for which $Mark(C) = M$ and $M$ is no terminal marking of $\mathcal{N}$ and $C$ is in conflict with all cutoff events.
Configuration \( C = \{e_2, e_4\} \) with \( \text{Mark}(C) = \{s_{P1W}, s_{P2W}\} \) is a deadlock!

Search techniques:
- Backtracking search
- Exponential time complexity for search

Figure: Petri Net \( \mathcal{N} \) with deadlock

Figure: Unfolding of \( \mathcal{N} \)
Conclusion

- Structure of Unfoldings was presented
Conclusion

- Structure of Unfoldings was presented
- Construction of Unfoldings was presented
Conclusion

- Structure of Unfoldings was presented
- Construction of Unfoldings was presented
- Unfolding not necessarily smaller than the Transition System

More concurrency $\Rightarrow$ Much smaller Unfolding
Size reduction up to an exponential factor possible
Search in Transition System linear running time in its size
Search in Unfolding exponential running time in its size
Overall verification speed is increased
Conclusion

- Structure of Unfoldings was presented
- Construction of Unfoldings was presented
- Unfolding not necessarily smaller than the Transition System
- More concurrency $\Rightarrow$ Much smaller Unfolding
- Structure of Unfoldings was presented
- Construction of Unfoldings was presented
- Unfolding not necessarily smaller than the Transition System
- More concurrency $\Rightarrow$ Much smaller Unfolding
- Size reduction up to an exponential factor possible
Conclusion

- Structure of Unfoldings was presented
- Construction of Unfoldings was presented
- Unfolding not necessarily smaller than the Transition System
- More concurrency \(\Rightarrow\) Much smaller Unfolding
- Size reduction up to an exponential factor possible
- Search in Transition System linear running time in its size
• Structure of Unfoldings was presented
• Construction of Unfoldings was presented
• Unfolding not necessarily smaller than the Transition System
• More concurrency \( \Rightarrow \) Much smaller Unfolding
• Size reduction up to an exponential factor possible
• Search in Transition System linear running time in its size
• Search in Unfolding exponential running time in its size
• Structure of Unfoldings was presented
• Construction of Unfoldings was presented
• Unfolding not necessarily smaller than the Transition System
• More concurrency ⇒ Much smaller Unfolding
• Size reduction up to an exponential factor possible
• Search in Transition System linear running time in its size
• Search in Unfolding exponential running time in its size
• Overall verification speed is increased
- Using Unfoldings to avoid the State Explosion Problem - K.L. McMillan
- An improvement of McMillan's Net Unfolding - J. Esparza, S. Römer, W. Vögler