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:

```
bool x = false;
while true do
    x = true;            # t_i
    wait(random)
    x = false;           # t̄_i
    wait(random
end while
```

Transition System of Software with concurrent processes

Example

- Given n processes all doing the same following task:

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

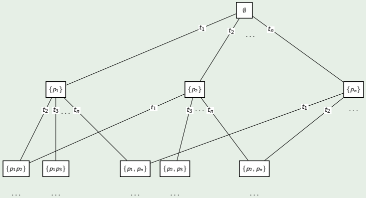


Figure: Transition System

- 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;           # t̄_i
    wait(random
end while
```

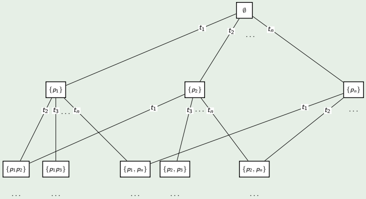


Figure: Transition System

- Each state represented by the set of processes where x = true
- In this example $|\mathcal{P}(\{p_1, \ldots, p_n\})| = 2^n$ states.

Example

- Given n processes all doing the same following task:

```
bool x = false;
while true do
    x = true;            # t_i
    wait(random)
    x = false;           # t̄_i
    wait(random
end while
```

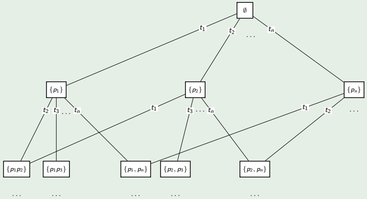


Figure: Transition System

- Each state represented by the set of processes where x = true
- In this example $|\mathcal{P}(\{p_1, \dots, p_n\})| = 2^n$ 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:

```
bool x = false;
while true do
    x = true;                # t_i
    wait(random)
    x = false;               # t̄_i
    wait(random)
end while
```

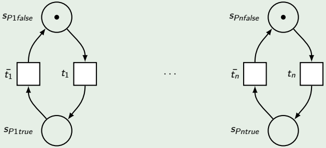


Figure: Petri net

Example

Given n processes all doing the same following task:

```
bool x = false;
while true do
    x = true;            # t_i
    wait(random)
    x = false;           # t̄_i
    wait(random)
end while
```

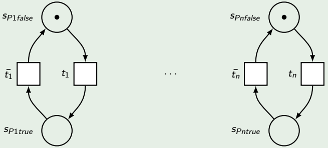


Figure: Petri net

- Size is linear in n

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;              # t̄_i
    wait(random)
end while
```

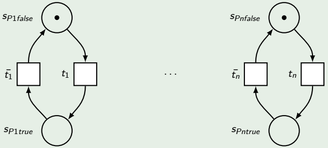


Figure: Petri net

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

Example

Given n processes all doing the same following task:

```
bool x = false;
while true do
    x = true;            # t_i
    wait(random)
    x = false;           # t̄_i
    wait(random)
end while
```

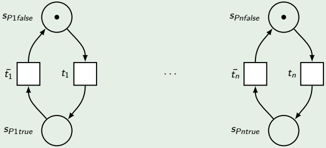


Figure: Petri net

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

Example

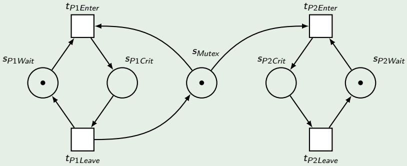


Figure: Petri net

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

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 Occurence Net with a labelling function

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 Occurence Net with a labelling function
- An Occurence 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 Occurence Net with a labelling function
- An Occurence Net is a net with a simpler structure
- The labelling functions assigns each node in the occurence net a label of the original net

Definition of Occurence Nets (1)

- Intuitively: Occurence Nets can be seen as 1-safe Nets.

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)

Definition of Occurence Nets (1)

- 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)

Definition of Occurence Nets (1)

- 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)

Example (Property 1)

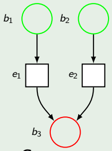


Figure: Counterexample

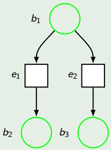


Figure: Example

- 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)

Example (Property 1)

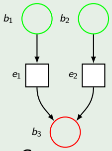


Figure: Counterexample

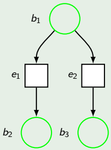


Figure: Example

- 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)

Example (Property 1)

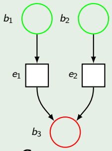


Figure: Counterexample

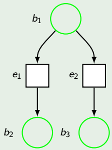


Figure: Example

- 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)

Example (Property 1)

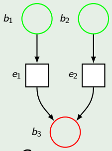


Figure: Counterexample

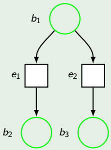


Figure: Example

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)

Example (Property 2)

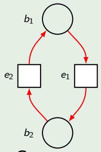


Figure: Counterexample

Definition of Occurence Nets (3)

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)

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)

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

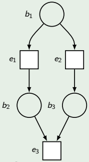


Figure: Counterexample

Configurations

- Configurations describe the possible events that can be fired in an Occurence Net

Configurations

- Configurations describe the possible events that can be fired in an Occurence 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. $\forall e, e' \in C : \neg(e \# e')$

- Configurations describe the possible events that can be fired in an Occurence Net

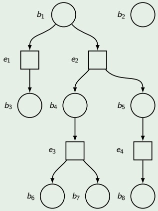
Definition

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

- $C_0 = \emptyset$



- Configurations describe the possible events that can be fired in an Occurence Net

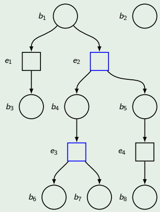
Definition

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 Occurence Net

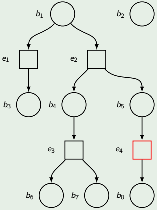
Definition

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 Occurence Net

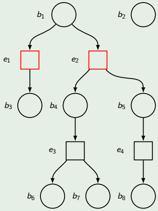
Definition

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)

Definition

Let $\mathcal{N} = (P, T, W, M_0)$ be a Petri net. A Branching Process is a labelled occurence 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 β) and ${}^\bullet p(e)$ (in $\mathcal{N}$)
3. The restriction of p to $Min(O) := \{b \in B | 0 = |{}^\bullet b|\}$ is a bijection between $Min(O)$ 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)

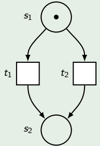


Figure: $\mathcal{N} = (P, T, W, M_0)$

- $p(b_1) = s_1 \in P$
- $p(b_2) = s_2 \in P$
- $p(b_3) = s_2 \in P$
- $p(e_1) = t_1 \in T$
- $p(e_2) = t_2 \in T$

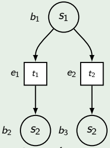


Figure: $\beta = (B, E, F, p)$

Definition: Branching Process (2)

Definition

Let $\mathcal{N} = (P, T, W, M_0)$ be a Petri net. A Branching Process is a labelled occurence 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 β) and ${}^\bullet p(e)$ (in $\mathcal{N}$)
3. The restriction of p to $Min(O) := \{b \in B | 0 = |{}^\bullet b|\}$ is a bijection between $Min(O)$ 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)

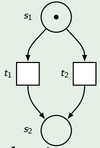


Figure: $\mathcal{N} = (P, T, W, M_0)$

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

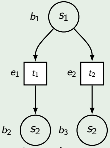


Figure: $\beta = (B, E, F, p)$

Definition

Let $\mathcal{N} = (P, T, W, M_0)$ be a Petri net. A Branching Process is a labelled occurence 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 β) and ${}^\bullet p(e)$ (in $\mathcal{N}$)
3. The restriction of p to $Min(O) := \{b \in B | 0 = |{}^\bullet b|\}$ is a bijection between $Min(O)$ 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 3: β starts at M_0)

- $M_0 = \{s_1, s_1, s_2\}$
- β 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

Let $\mathcal{N} = (P, T, W, M_0)$ be a Petri net. A Branching Process is a labelled occurence 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 β) and ${}^\bullet p(e)$ (in $\mathcal{N}$)
3. The restriction of p to $Min(O) := \{b \in B | 0 = |{}^\bullet b|\}$ is a bijection between $Min(O)$ 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 4: β does not duplicate transitions)

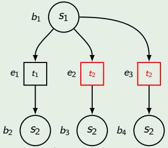


Figure: Counterexample

- 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. $\forall e, e' \in C : \neg(e \# e')$

- $Mark(C)$ denotes the marking reached by fireing all events in C.

- What is the marking reached by a configuration?

Definition

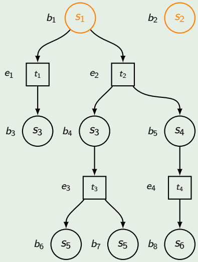
A set of events is a configuration C iff the following properties are satisfied:

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

- $Mark(C)$ denotes the marking reached by fireing all events in C.

Example

- $Mark(\emptyset) = p(Min(O)) = p(\{b_1, b_2\}) = \{s_1, s_2\}$



- What is the marking reached by a configuration?

Definition

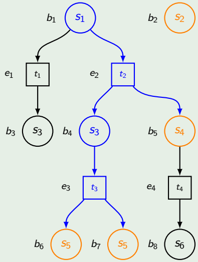
A set of events is a configuration C iff the following properties are satisfied:

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

- $Mark(C)$ denotes the marking reached by fireing all events in C.

Example

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



Example

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

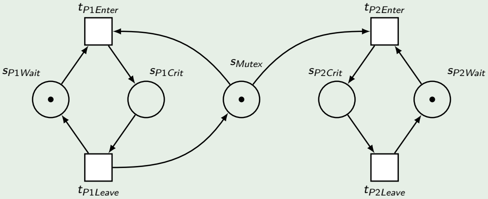


Figure: Petri Net $\mathcal{N}$

Example

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

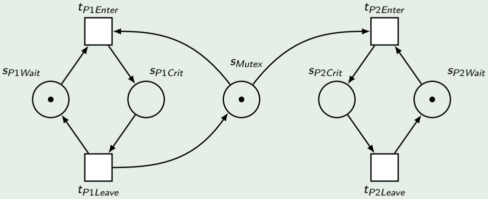


Figure: Petri Net $\mathcal{N}$

- Goal: Compute Unfolding of $\mathcal{N}$

Example

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

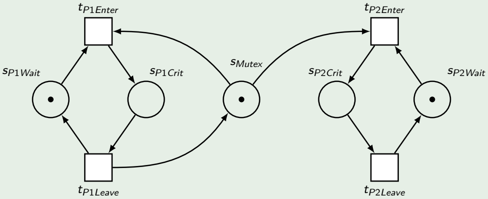


Figure: Petri Net $\mathcal{N}$

- 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

Example: Unfolding of Mutual Exclusion Model

Definition (Occurence 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

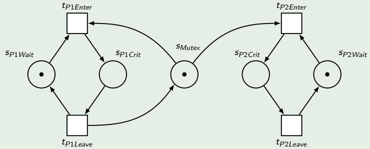


Figure: Petri Net $\mathcal{N}$

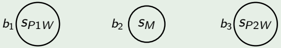


Figure: Current State while unfolding

Example: Unfolding of Mutual Exclusion Model

Definition (Occurence 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

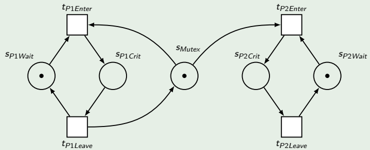


Figure: Petri Net $\mathcal{N}$

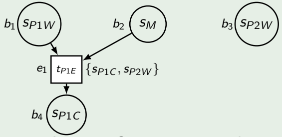


Figure: Current State while unfolding

Definition (Occurence 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

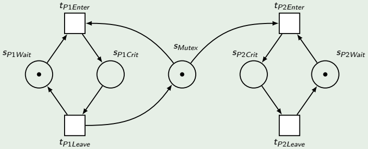


Figure: Petri Net $\mathcal{N}$

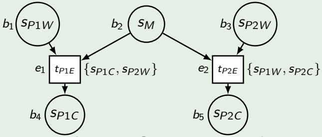


Figure: Current State while unfolding

Definition (Occurence 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

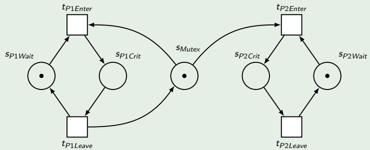


Figure: Petri Net $\mathcal{N}$

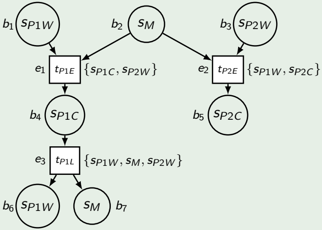


Figure: Current State while unfolding

Definition (Occurence 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

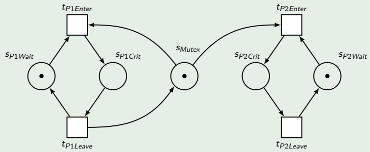


Figure: Petri Net $\mathcal{N}$

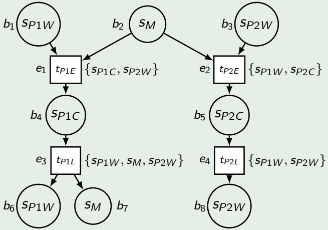


Figure: Current State while unfolding

Definition (Occurence 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

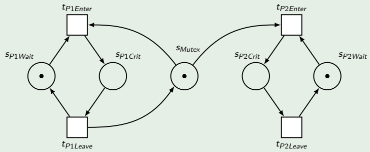
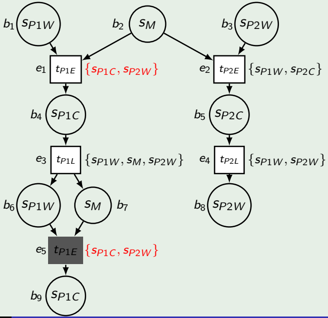


Figure: Petri Net $\mathcal{N}$



Definition (Occurence 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

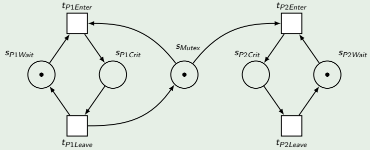
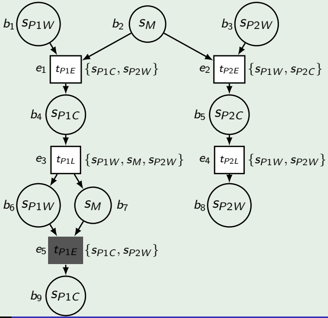


Figure: Petri Net $\mathcal{N}$



Deadlock checking in Unfoldings

Definition (Deadlock: Petri Net vs Unfolding)

Let $\mathcal{N}$ be a Petri Net and β its Unfolding.
$\mathcal{N}$ has a deadlock if there exists a reachable marking M which is no terminal marking and no transition is enabled.
$\Leftrightarrow$ There exists a configuration C in β 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 $Mark(C) = \{s_{P1W}, s_{P2W}\}$ is a deadlock!
- Search techniques:
  - Backtracking search
- Exponential time complexity for search

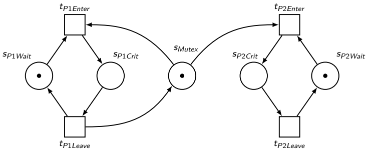


Figure: Petri Net $\mathcal{N}$ with deadlock

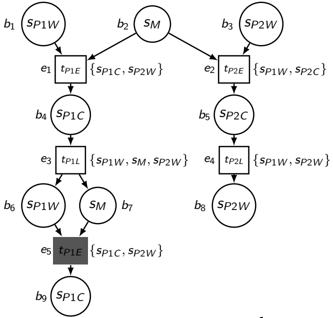


Figure: Unfolding of $\mathcal{N}$

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

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

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

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

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

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

References

- Using Unfoldings to avoid the State Explosion Problem - K.L. McMillan
- An improvment of McMillans Net Unfolding - J. Esparza, S. Römer, W. Vögler