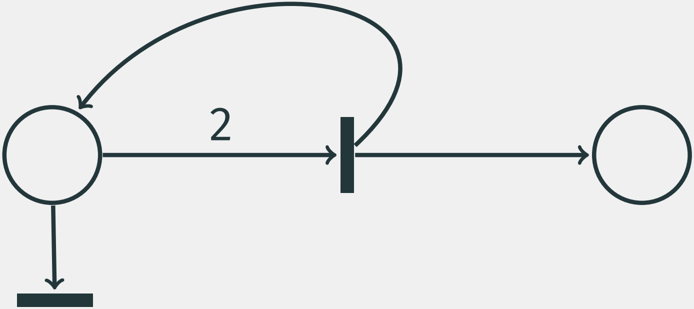
Approaching the Coverability Problem Continuously

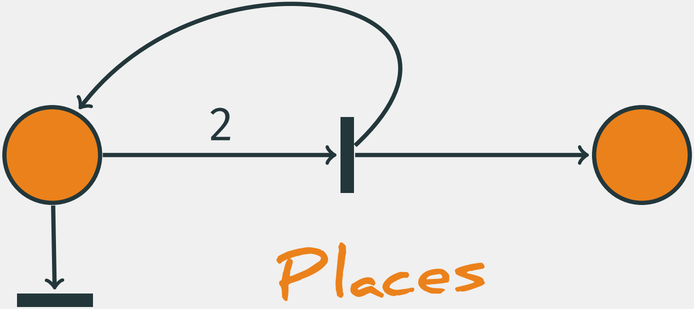
Michael Blondin, Alain Finkel, Christoph Haase, Serge Haddad

université PARIS-SACLAY
ENS CACHAN
CNRS
Inria INVENTORS FOR THE DIGITAL WORLD
Université de Montréal

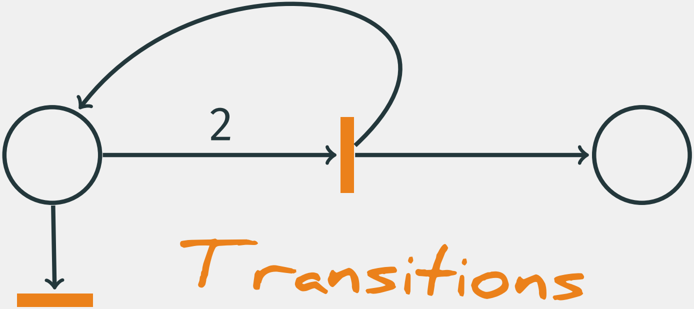
(Discrete) Petri nets



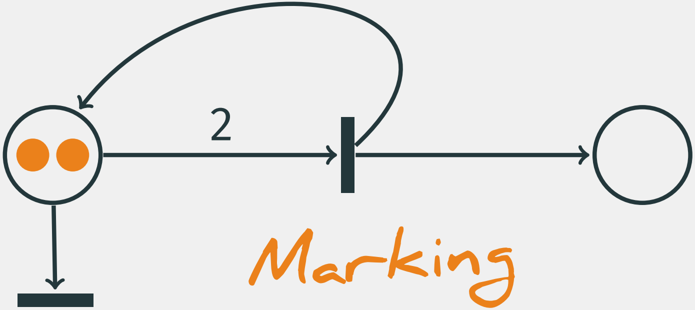
(Discrete) Petri nets



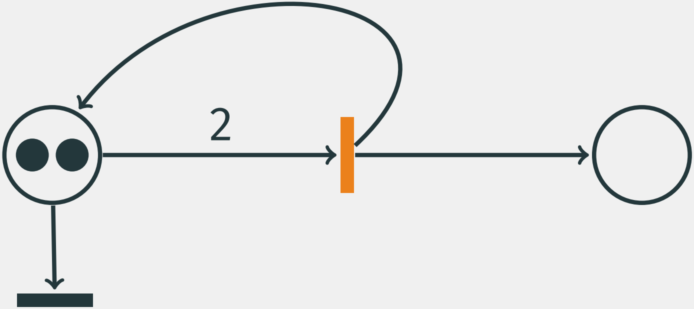
(Discrete) Petri nets



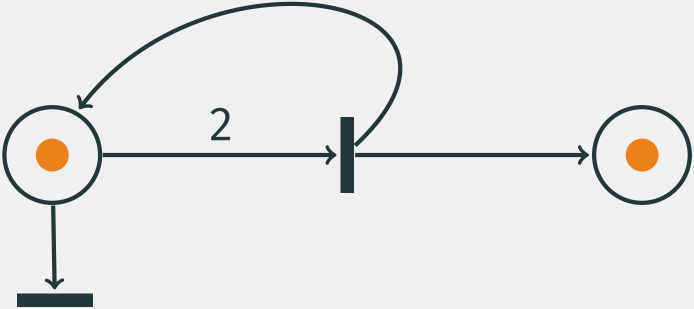
(Discrete) Petri nets



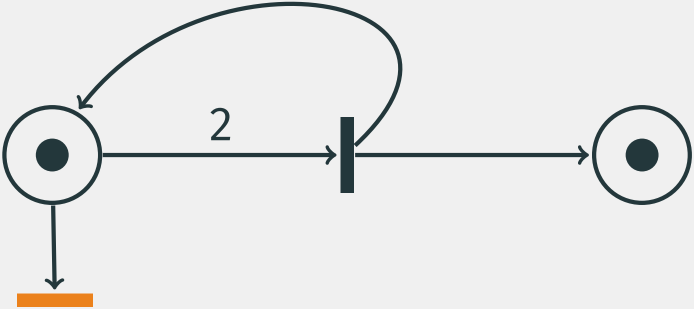
(Discrete) Petri nets



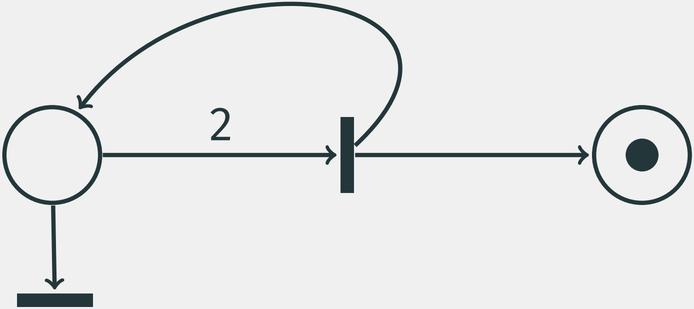
(Discrete) Petri nets



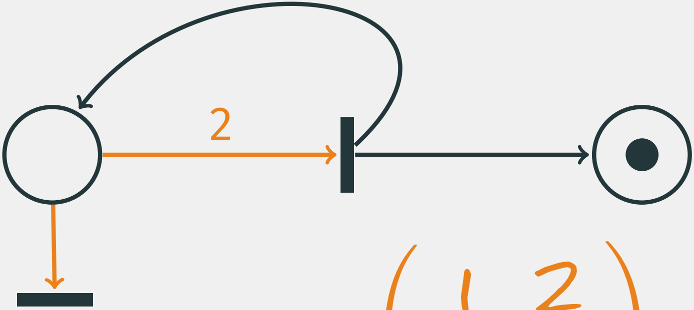
(Discrete) Petri nets



(Discrete) Petri nets

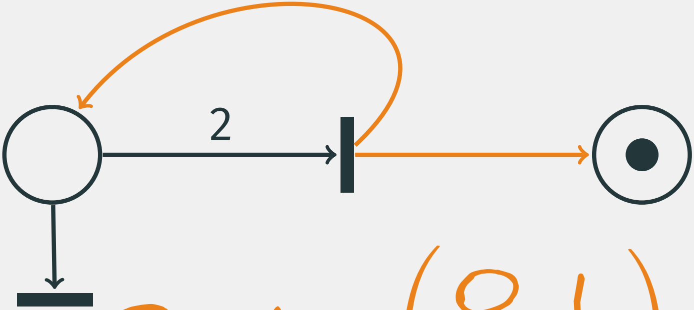


(Discrete) Petri nets



$$Pre = \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix}$$

(Discrete) Petri nets



$$Post = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Lamport mutual exclusion "1-bit algorithm"

Verifying safety with Petri nets

Process 1

Process 2

Lamport mutual exclusion "1-bit algorithm"

Lamport mutual exclusion "1-bit algorithm"

Verifying safety with Petri nets

```
while True:
  x = True
  while y: pass
  # critical section
  x = False
```

```
while True:
★ y = True
  if x then:
    y = False
    while x: pass
    goto ★
  # critical section
  y = False
```

Verifying safety with Petri nets

```
while True:
  x = True
  while y: pass
  # critical section
  x = False
```

```
while True:
★ y = True
  if x then:
    y = False
    while x: pass
    goto ★
  # critical section
  y = False
```

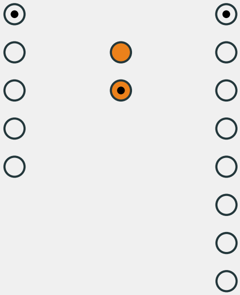
Verifying safety with Petri nets

```
while True:
  x = True
  while y: pass
  # critical section
  x = False
```

```
  while True:
★   y = True
    if x then:
      y = False
      while x: pass
      goto ★
    # critical section
    y = False
```

Verifying safety with Petri nets

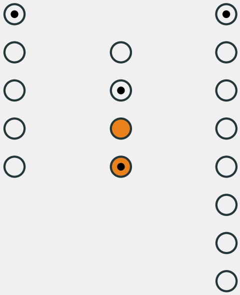
```
while True:
  x = True
  while y: pass
  # critical section
  x = False
```



```
  while True:
★   y = True
    if x then:
      y = False
      while x: pass
      goto ★
    # critical section
    y = False
```

Verifying safety with Petri nets

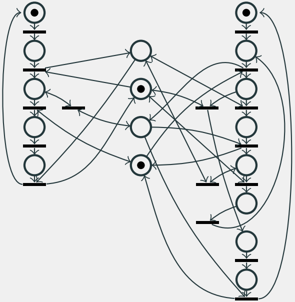
```
while True:
  x = True
  while y: pass
  # critical section
  x = False
```



```
while True:
★ y = True
   if x then:
     y = False
     while x: pass
     goto ★
   # critical section
   y = False
```

Verifying safety with Petri nets

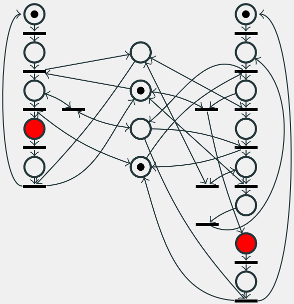
```
while True:
  x = True
  while y: pass
  # critical section
  x = False
```



```
while True:
★ y = True
  if x then:
    y = False
    while x: pass
    goto ★
  # critical section
  y = False
```

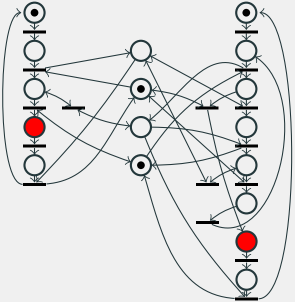
Verifying safety with Petri nets

```
while True:
  x = True
  while y: pass
  # critical section
  x = False
```



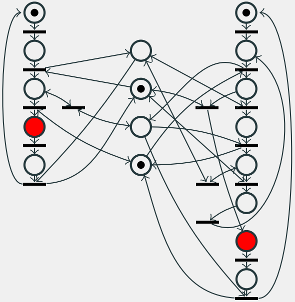
```
while True:
★   y = True
    if x then:
      y = False
      while x: pass
      goto ★
    # critical section
    y = False
```

Verifying safety with Petri nets



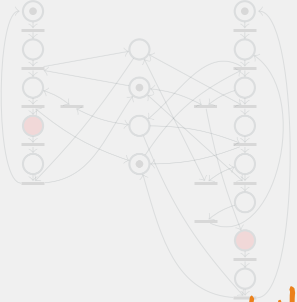
Processes at both critical sections $\iff$ each  ≥ 1

Verifying safety with Petri nets



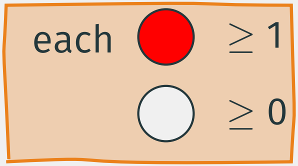
Processes at both critical sections $\iff$ each  ≥ 1

 ≥ 0



Coverability problem

Processes at both critical sections $\iff$



Coverability problem

Problem

Input:	Petri net $\mathcal{N}$, initial marking $\boldsymbol{m}_0$, target marking $\boldsymbol{m}$
Question:	Is some $\boldsymbol{m}' \geq \boldsymbol{m}$ reachable from $\boldsymbol{m}_0$ in $\mathcal{N}$?

Coverability problem

Problem

Input:	Petri net $\mathcal{N}$, initial marking $\boldsymbol{m}_0$, target marking $\boldsymbol{m}$
Question:	Is some $\boldsymbol{m}' \geq \boldsymbol{m}$ reachable from $\boldsymbol{m}_0$ in $\mathcal{N}$?

How to solve it?

- Forward: build reachability tree from initial marking
- Backward: find predecessors of markings covering target
- EXPSPACE-complete

Coverability problem

Problem

Input:	Petri net $\mathcal{N}$, initial marking $\boldsymbol{m}_0$, target marking $\boldsymbol{m}$
Question:	Is some $\boldsymbol{m}' \geq \boldsymbol{m}$ reachable from $\boldsymbol{m}_0$ in $\mathcal{N}$?

How to solve it? **Karp & Miller '69**

- Forward: build reachability tree from initial marking
- Backward: find predecessors of markings covering target
- EXPSPACE-complete

Coverability problem

Problem

Input: Petri net $\mathcal{N}$, initial marking $\boldsymbol{m}_0$, target marking $\boldsymbol{m}$

Question: Is some $\boldsymbol{m}' \geq \boldsymbol{m}$ reachable from $\boldsymbol{m}_0$ in $\mathcal{N}$?

How to solve it? **Arnold & Latteux '78, Abdulla *et al.* '96**

- Forward: build reachability tree from initial marking
- Backward: find predecessors of markings covering target
- EXPSPACE-complete

Coverability problem

Problem

Input: Petri net $\mathcal{N}$, initial marking $\boldsymbol{m}_0$, target marking $\boldsymbol{m}$

Question: Is some $\boldsymbol{m}' \geq \boldsymbol{m}$ reachable from $\boldsymbol{m}_0$ in $\mathcal{N}$?

How to solve it? **Lipton '76, Rackoff '78**

- Forward: build reachability tree from initial marking
- Backward: find predecessors of markings covering target
- EXPSPACE-complete

Coverability problem

Problem

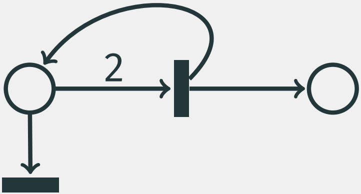
Input: Petri net $\mathcal{N}$, initial marking $\boldsymbol{m}_0$, target marking $\boldsymbol{m}$

Question: Is some $\boldsymbol{m}' \geq \boldsymbol{m}$ reachable from $\boldsymbol{m}_0$ in $\mathcal{N}$?

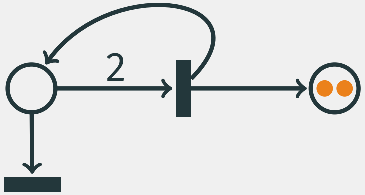
How to solve it?

- Forward: build reachability tree from initial marking
- Backward: find predecessors of markings covering target
- EXPSPACE-complete

Backward algorithm

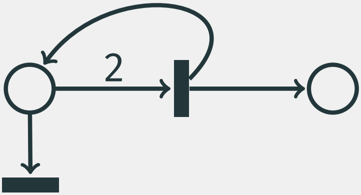


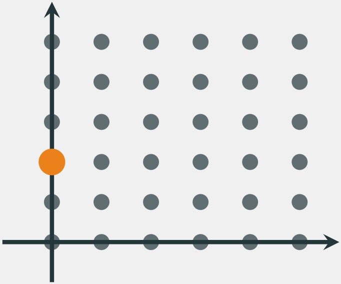
Backward algorithm



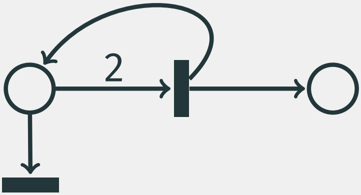
What initial markings may cover $(0, 2)$?

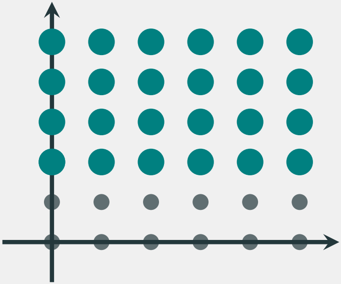
Backward algorithm



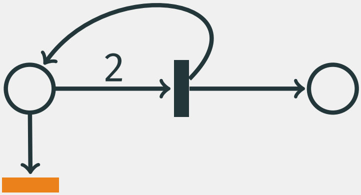


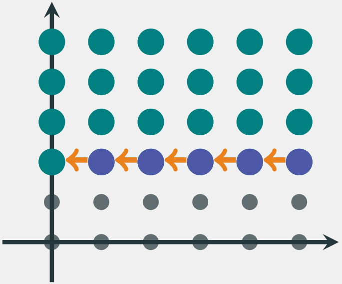
Backward algorithm



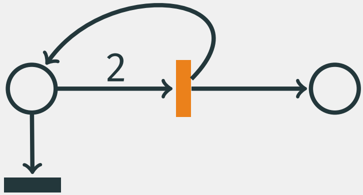


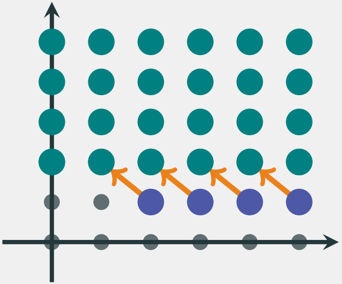
Backward algorithm



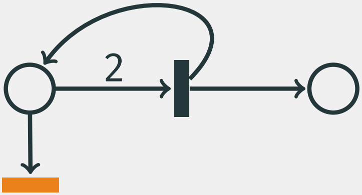


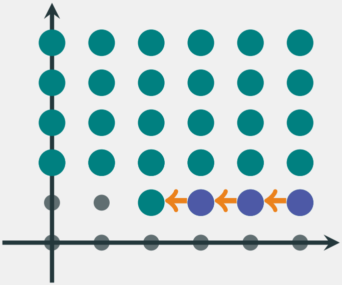
Backward algorithm

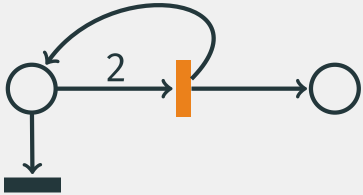


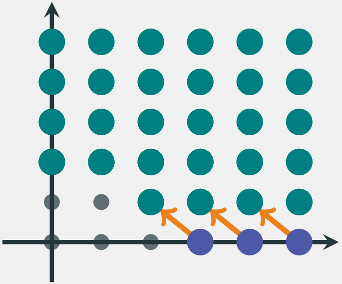


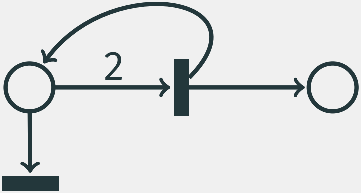
Backward algorithm

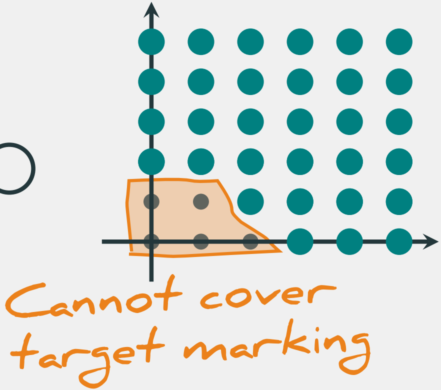


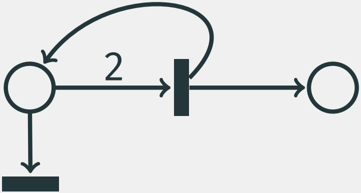


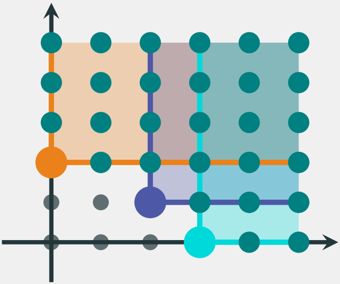

2




2


Cannot cover
target marking

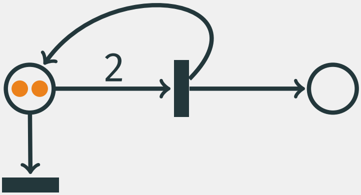


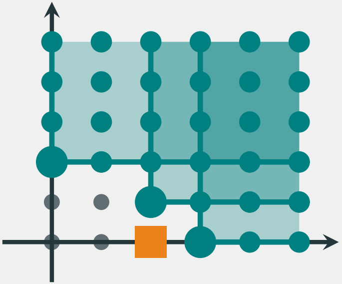


Basis size may become doubly exponential

(Bozzelli & Ganty '11)

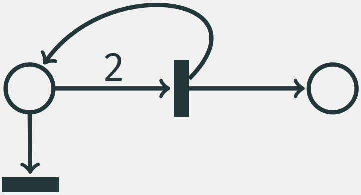
Backward algorithm

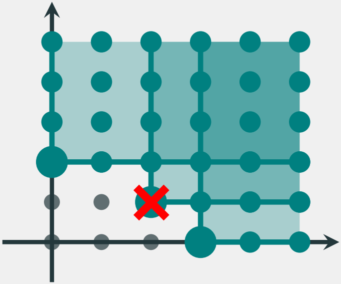




We only care about some initial marking...

Backward algorithm

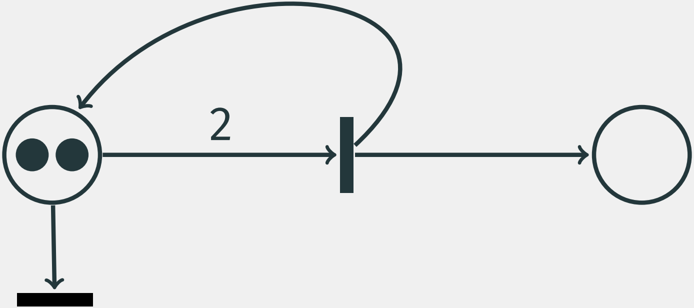




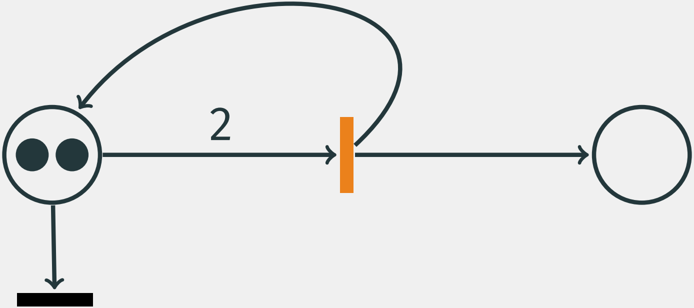
We only care about some initial marking...

Speedup by pruning basis!

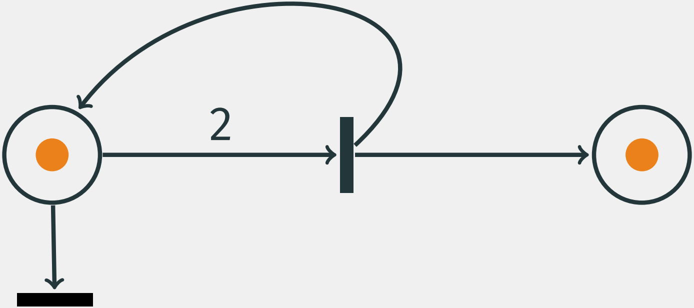
(Discrete) Petri nets

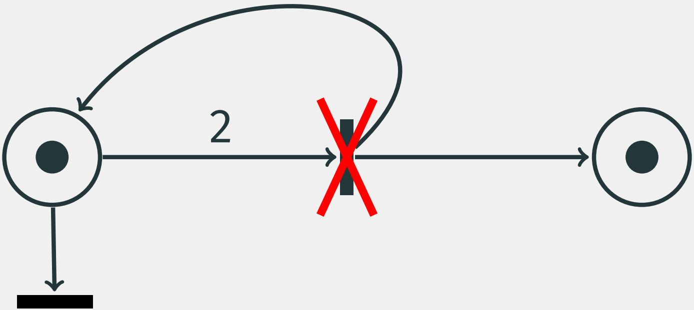


(Discrete) Petri nets

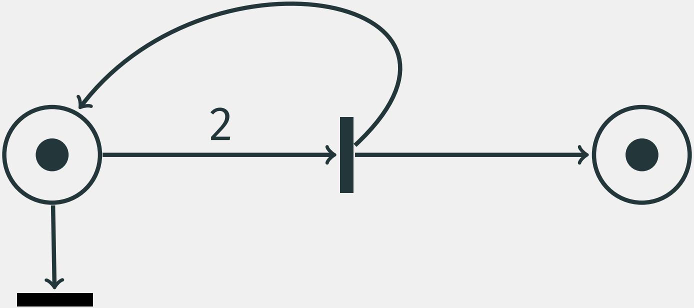


(Discrete) Petri nets

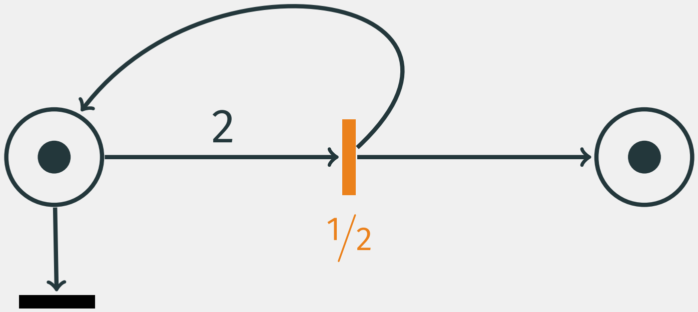



2

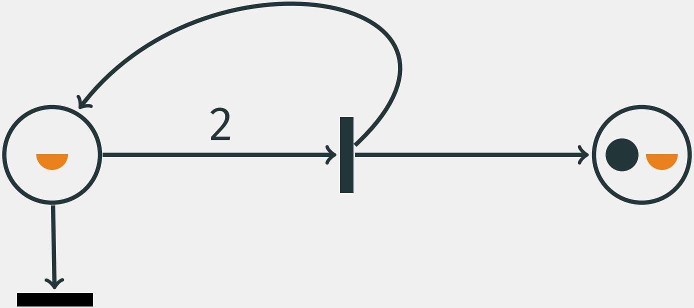
~~(Discrete)~~ Continuous Petri nets



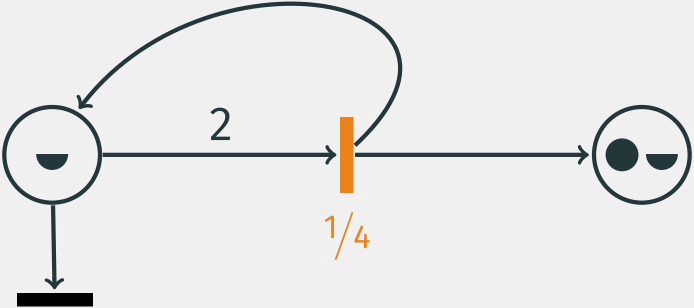
~~(Discrete)~~ Continuous Petri nets



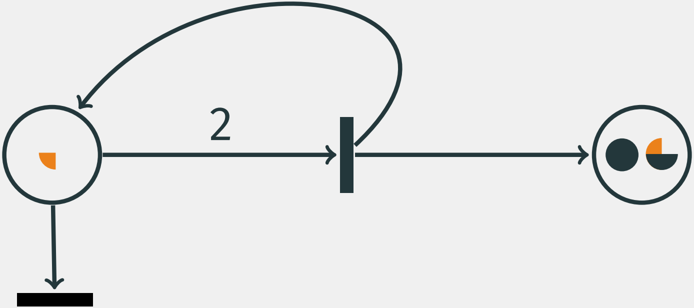
~~(Discrete)~~ Continuous Petri nets



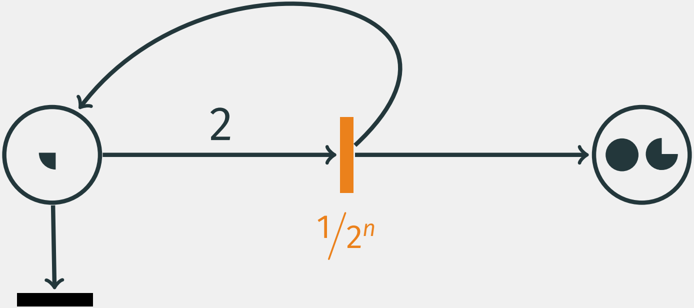
~~(Discrete)~~ Continuous Petri nets



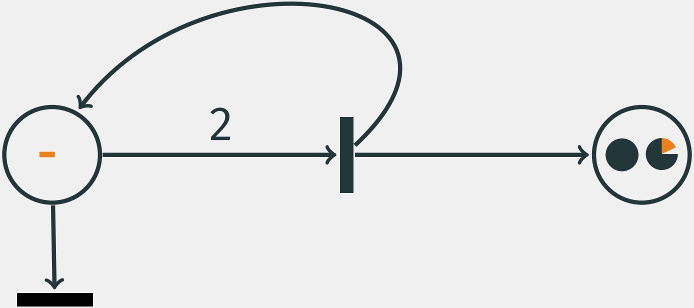
~~(Discrete)~~ Continuous Petri nets



~~(Discrete)~~ Continuous Petri nets



~~(Discrete)~~ Continuous Petri nets



$\boldsymbol{m}$ is coverable from $\boldsymbol{m}_0$

$$\Downarrow$$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$

Continuity to over-approximate coverability

$\boldsymbol{m}$ is coverable from $\boldsymbol{m}_0$

EXPSPACE

$\Downarrow$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$

$\Downarrow \quad \not\Uparrow$

PTIME

$\boldsymbol{m}_0$ and $\boldsymbol{m}$ satisfy conditions of

Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic '14

PTIME / NP / coNP

$\boldsymbol{m}$ is not coverable from $\boldsymbol{m}_0$

Safety

$\Uparrow$

$\boldsymbol{m}$ is not $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$

Coverability in continuous Petri nets

Fix some continuous Petri net $(P, T, \mathbf{Pre}, \mathbf{Post})$

m is $\mathbb{Q}$-coverable from m_0 iff... **Fraca & Haddad '13**

Coverability in continuous Petri nets

Fix some continuous Petri net $(P, T, \mathbf{Pre}, \mathbf{Post})$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v} \in \mathbb{Q}^{T}_{\geq 0}$ such that

- $\boldsymbol{m}' = \boldsymbol{m_0} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$

Coverability in continuous Petri nets

Fix some continuous Petri net $(P, T, \mathbf{Pre}, \mathbf{Post})$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}' = \boldsymbol{m_0} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets

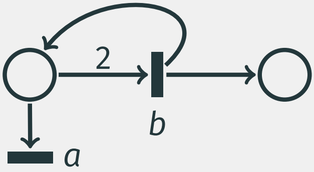
Fix some continuous Petri net $(P, T, \mathbf{Pre}, \mathbf{Post})$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}' = \boldsymbol{m}_0 + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets



$\boldsymbol{m}_0 = (2, 0)$

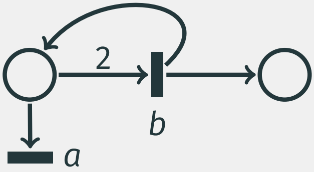
$\boldsymbol{m} \;\; = (0, 2)$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\boldsymbol{m}' = \boldsymbol{m}_0 + (\textbf{Post} - \textbf{Pre}) \cdot \boldsymbol{v}$
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets



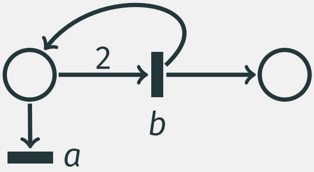
$$\boldsymbol{m}_0 = (2, 0)$$
$$\boldsymbol{m} = (0, 2)$$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2$
  $2 \leq \boldsymbol{v}_b$
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets



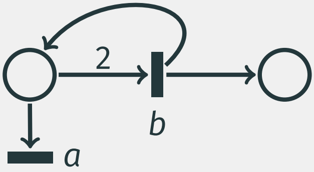
$$\boldsymbol{m}_0 = (2, 0)$$
$$\boldsymbol{m} = (0, 2)$$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2$, $2 \leq \boldsymbol{v}_b$ $\implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2,\ \boldsymbol{m}' = \boldsymbol{m}$
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets



$\boldsymbol{m}_0 = (2, 0)$

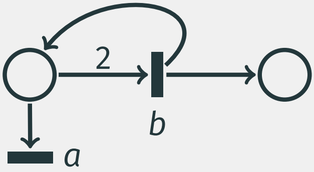
$\boldsymbol{m} \;\; = (0, 2)$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{array}{l} 0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2 \\ 2 \leq \boldsymbol{v}_b \end{array} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets



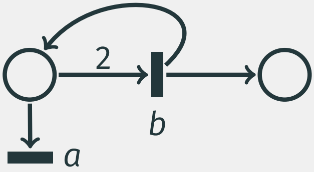
$$\boldsymbol{m}_0 = (2, 0)$$
$$\boldsymbol{m} \;= (0, 2)$$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{array}{l} 0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2 \\ 2 \leq \boldsymbol{v}_b \end{array} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets



$\boldsymbol{m}_0 = (2, 0)$

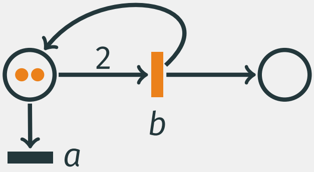
$\boldsymbol{m} \;\; = (0, 2)$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{array}{l} 0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2 \\ 2 \leq \boldsymbol{v}_b \end{array} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{b\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{b\}$

Coverability in continuous Petri nets



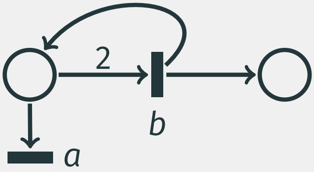
$\boldsymbol{m}_0 = (2, 0)$
$\boldsymbol{m} \;\; = (0, 2)$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{matrix} 0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2 \\ 2 \leq \boldsymbol{v}_b \end{matrix} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{b\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{b\}$

Coverability in continuous Petri nets



$\boldsymbol{m}_0 = (2, 0)$

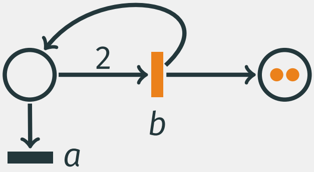
$\boldsymbol{m} = (0, 2)$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{matrix} 0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2 \\ 2 \leq \boldsymbol{v}_b \end{matrix} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{b\}$ ✓
- some execution to $\boldsymbol{m}'$ fires exactly $\{b\}$

Coverability in continuous Petri nets



$\boldsymbol{m}_0 = (2, 0)$

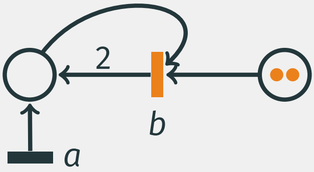
$\boldsymbol{m} \;\; = (0, 2)$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{matrix} 0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2 \\ 2 \leq \boldsymbol{v}_b \end{matrix} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{b\}$ ✓
- some execution to $\boldsymbol{m}'$ fires exactly $\{b\}$

Coverability in continuous Petri nets



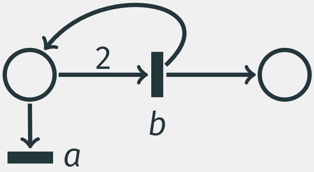
$$\boldsymbol{m}_0 = (2, 0)$$
$$\boldsymbol{m} = (0, 2)$$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2$, $2 \leq \boldsymbol{v}_b$ $\implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{b\}$ ✓
- some execution to $\boldsymbol{m}'$ fires exactly $\{b\}$

Coverability in continuous Petri nets



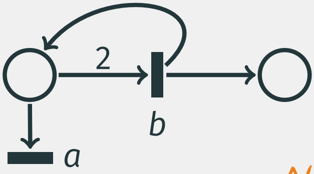
$$\boldsymbol{m}_0 = (2, 0)$$
$$\boldsymbol{m} = (0, 2)$$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{array}{l} 0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2 \\ 2 \leq \boldsymbol{v}_b \end{array} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{b\}$ ✓
- some execution to $\boldsymbol{m}'$ fires exactly $\{b\}$ ✗

Coverability in continuous Petri nets



$\boldsymbol{m}_0 = (2, 0)$

$\boldsymbol{m} = (0, 2)$

Not $\mathbb{Q}$-coverable from

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{array}{l} 0 \leq \boldsymbol{v}_b + \boldsymbol{v}_a \leq 2 \\ 2 \leq \boldsymbol{v}_b \end{array} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{b\}$ ✓
- some execution to $\boldsymbol{m}'$ fires exactly $\{b\}$ ✗

Coverability in continuous Petri nets

Polynomial time!

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}' = \boldsymbol{m}_0 + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets

Logical characterization **Contribution**

$\mathbb{Q}$-coverability can be encoded in a linear size formula of

existential $\mathsf{FO}(\mathbb{Q}_{\geq 0}, +, <)$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}' = \boldsymbol{m_0} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets

Logical characterization **Contribution**

$\mathbb{Q}$-coverability can be encoded in a linear size formula of

$$\text{existential FO}(\mathbb{N}, +, <)$$

Even better approximation

***m* is $\mathbb{Q}$-coverable from m_0 iff...** **Fraca & Haddad '13**

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v} \in \mathbb{Q}^T_{\geq 0}$ such that

- $\boldsymbol{m}' = \boldsymbol{m_0} + (\textbf{Post} - \textbf{Pre}) \cdot \boldsymbol{v}$
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets

Logical characterization

Contribution

$\mathbb{Q}$-coverability can be encoded in a linear size formula of

$$\text{existential FO}(\mathbb{Q}_{\geq 0}, +, <)$$

$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...

Fraca & Haddad '13

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}' = \boldsymbol{m}_0 + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$ ← Straightforward
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Coverability in continuous Petri nets

Logical characterization — Contribution

$\mathbb{Q}$-coverability can be encoded in a linear size formula of

$$\text{existential FO}(\mathbb{Q}_{\geq 0}, +, <)$$

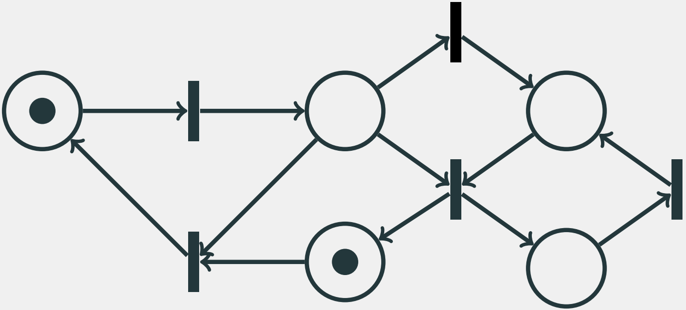
$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff... — Fraca & Haddad '13

there exist $\boldsymbol{m}' \geq \boldsymbol{m}$ and $\boldsymbol{v} \in \mathbb{Q}^T_{\geq 0}$ such that

- $\boldsymbol{m}' = \boldsymbol{m_0} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- some execution from $\boldsymbol{m}_0$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

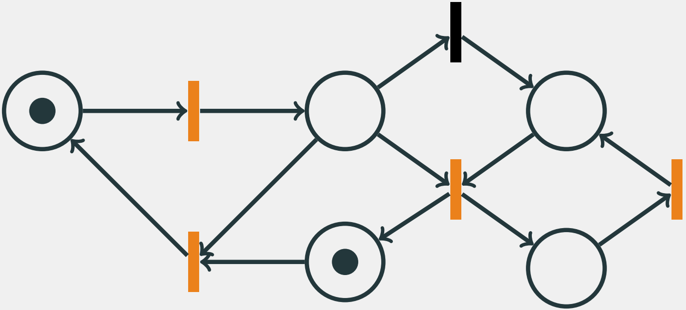
More subtle

Encoding the firing set conditions

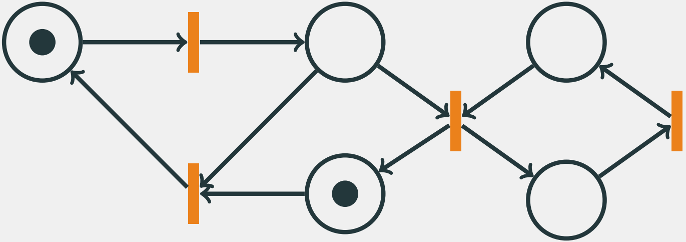


Testing whether some transitions can be fired from initial marking

Encoding the firing set conditions

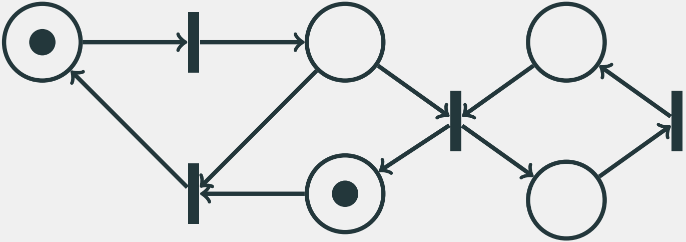


Testing whether some transitions can be fired from initial marking



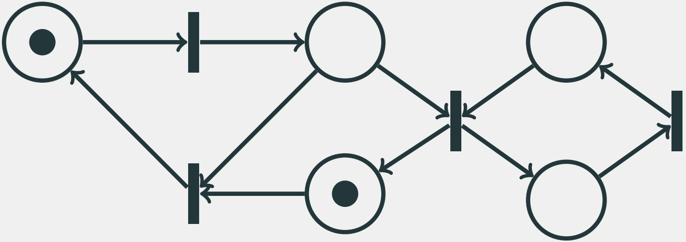
Testing whether some transitions can be fired from initial marking

Encoding the firing set conditions



Simulate a "breadth-first" transitions firing

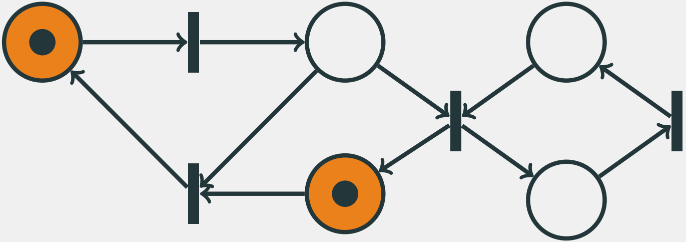
Encoding the firing set conditions



Simulate a "breadth-first" transitions firing

by numbering places/transitions

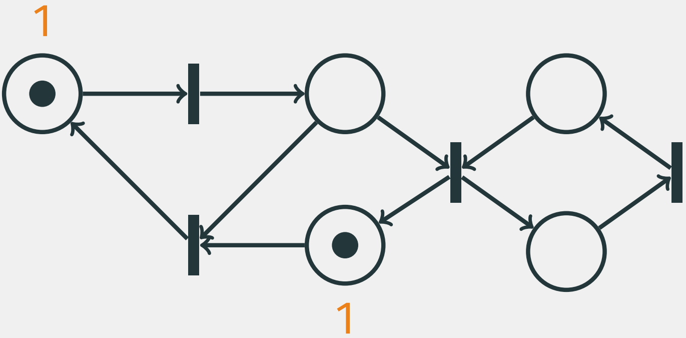
(Verma, Seidl & Schwentick '05)



Simulate a "breadth-first" transitions firing
by numbering places/transitions

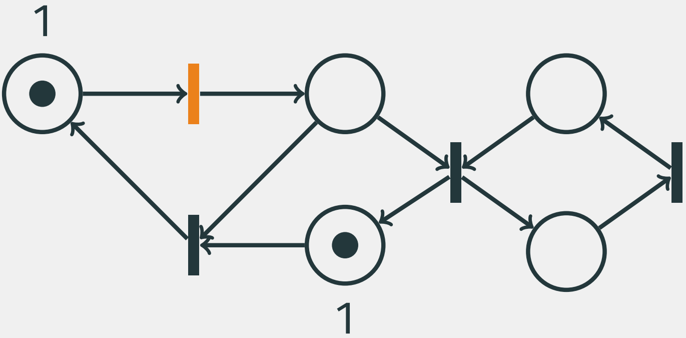
(Verma, Seidl & Schwentick '05)

Encoding the firing set conditions



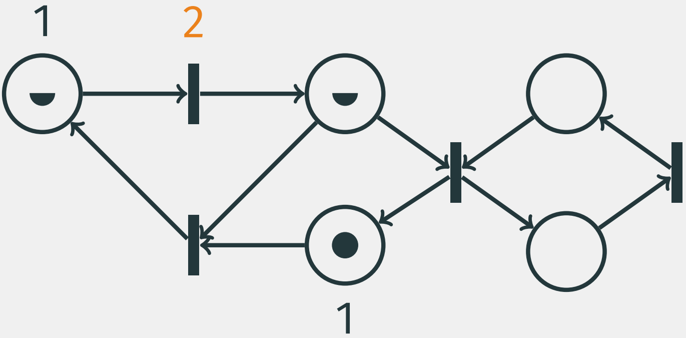
Simulate a "breadth-first" transitions firing
by numbering places/transitions
(Verma, Seidl & Schwentick '05)

Encoding the firing set conditions



Simulate a "breadth-first" transitions firing
by numbering places/transitions
(Verma, Seidl & Schwentick '05)

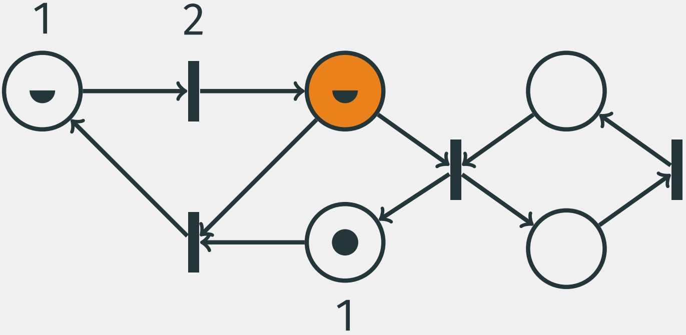
Encoding the firing set conditions



Simulate a "breadth-first" transitions firing
by numbering places/transitions

(Verma, Seidl & Schwentick '05)

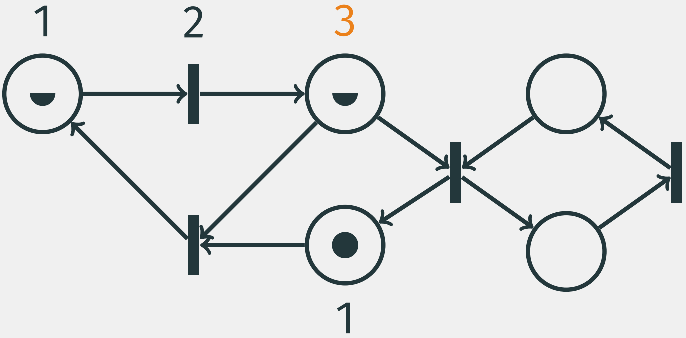
Encoding the firing set conditions



Simulate a "breadth-first" transitions firing by numbering places/transitions

(Verma, Seidl & Schwentick '05)

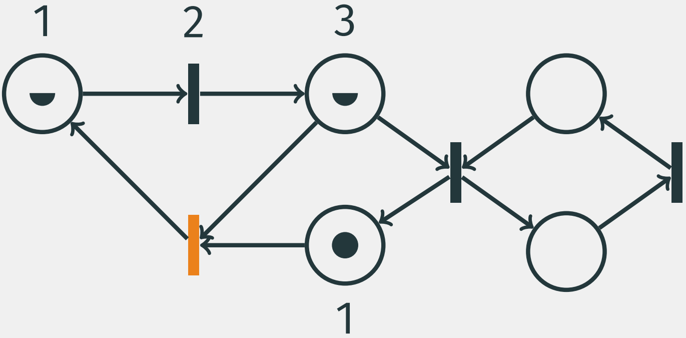
Encoding the firing set conditions



Simulate a "breadth-first" transitions firing by numbering places/transitions

(Verma, Seidl & Schwentick '05)

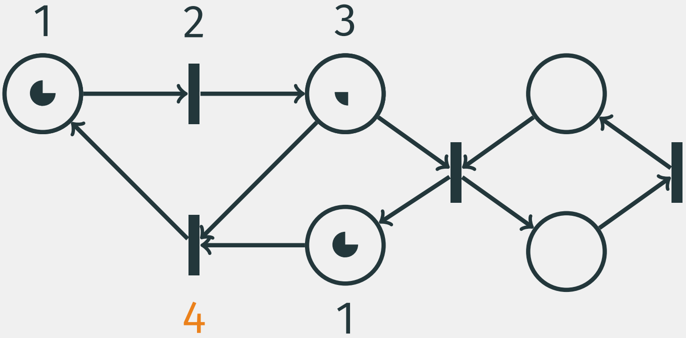
Encoding the firing set conditions



Simulate a "breadth-first" transitions firing by numbering places/transitions

(Verma, Seidl & Schwentick '05)

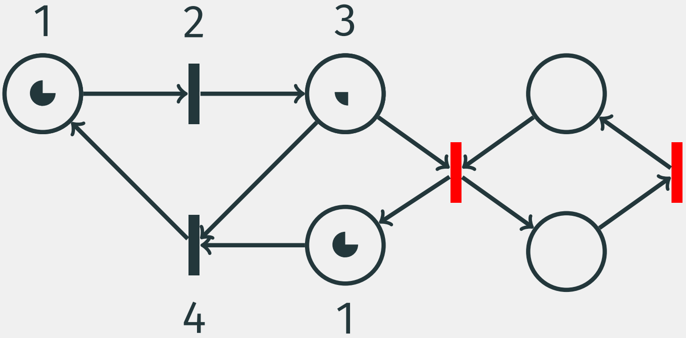
Encoding the firing set conditions



Simulate a "breadth-first" transitions firing by numbering places/transitions

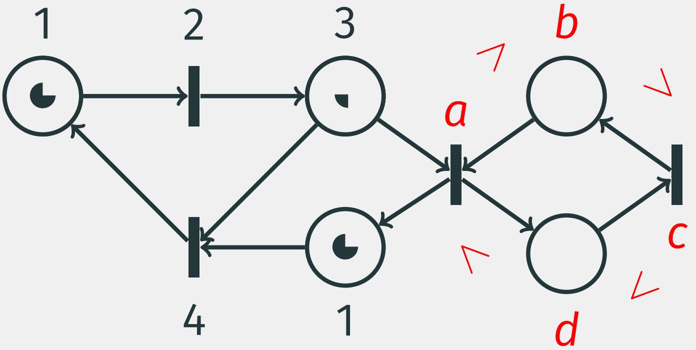
(Verma, Seidl & Schwentick '05)

Encoding the firing set conditions



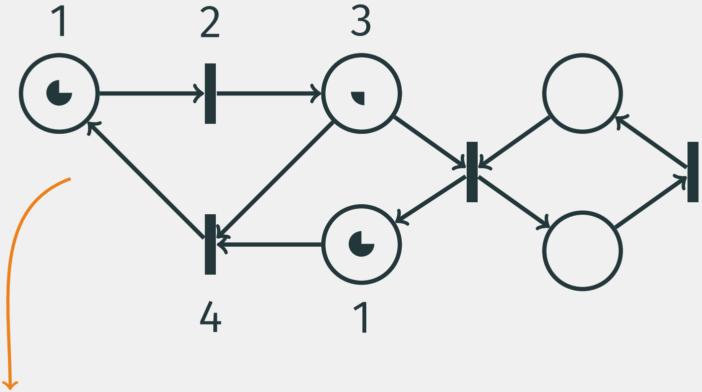
Simulate a "breadth-first" transitions firing
by numbering places/transitions

(Verma, Seidl & Schwentick '05)



Simulate a "breadth-first" transitions firing
by numbering places/transitions
(Verma, Seidl & Schwentick '05)

Encoding the firing set conditions



$$\varphi(\boldsymbol{x}) = \exists \boldsymbol{y} : \bigwedge_{p \in P} \boldsymbol{y}(p) > 0 \rightarrow \bigwedge_{t \in {}^\bullet p} \boldsymbol{y}(t) < \boldsymbol{y}(p) \cdots$$

Backward coverability modulo $\mathbb{Q}$-coverability

```
if target marking m is not ℚ-coverable:
  return False
```

Polynomial time

Backward coverability modulo $\mathbb{Q}$-coverability

```
if target marking m is not Q-coverable:
  return False
X = {target marking m}
while (initial marking m_0 not covered by X):
  B = markings obtained from X one step backward
  B = B \ {b ∈ B : ¬φ(b)}
  if B = ∅: return False
  φ(x) = φ(x) ∧ ⋀_{pruned b} x ≱ b
  X = X ∪ B
return True
```

Backward coverability modulo $\mathbb{Q}$-coverability

```
if target marking m is not Q-coverable:
  return False
X = {target marking m}
while (initial marking m_0 not covered by X):
  B = markings obtained from X one step backward
  B = B \ {b ∈ B : ¬φ(b)}
  if B = ∅: return False
  φ(x) = φ(x) ∧ ⋀_{pruned b} x ≱ b
  X = X ∪ B
return True
```

Backward coverability modulo $\mathbb{Q}$-coverability

```
if target marking m is not ℚ-coverable:
  return False
X = {target marking m}
while (initial marking m_0 not covered by X):
  B = markings obtained from X one step backward
  B = B \ {b ∈ B : ¬φ(b)}
  if B = ∅: return False
  φ(x) = φ(x) ∧ ⋀_{pruned b} x ≱ b
  X = X ∪ B
return True
```

Backward coverability modulo $\mathbb{Q}$-coverability

```
if target marking m is not Q-coverable:
    return False
X = {target marking m}
while (initial marking m_0 not covered by X):
    B = markings obtained from X one step backward
    B = B \ {b ∈ B : ¬φ(b)}
    if B = ∅: return False
    φ(x) = φ(x) ∧ ⋀_{pruned b} x ≱ b
    X = X ∪ B
return True
```

Backward coverability modulo $\mathbb{Q}$-coverability

```
if target marking m is not ℚ-coverable:
  return False
```

$X = \{\text{target marking } \boldsymbol{m}\}$

while (initial marking $\boldsymbol{m}_0$ not covered by X):

  B = markings obtained from X one step backward

  $B = B \setminus \{\boldsymbol{b} \in B : \neg\varphi(\boldsymbol{b})\}$ ← $\mathbb{Q}$-coverability pruning (better than poly. time)

  if $B = \emptyset$: **return** False

  $\varphi(\boldsymbol{x}) = \varphi(\boldsymbol{x}) \ \wedge \ \bigwedge_{\text{pruned } \boldsymbol{b}} \boldsymbol{x} \not\geq \boldsymbol{b}$

  $X = X \cup B$

return True

Backward coverability modulo $\mathbb{Q}$-coverability

```
if target marking m is not Q-coverable:
  return False
X = {target marking m}
while (initial marking m_0 not covered by X):
  B = markings obtained from X one step backward
  B = B \ {b ∈ B : ¬φ(b)}
  if B = ∅: return False
  φ(x) = φ(x) ∧ ⋀_{pruned b} x ≱ b
  X = X ∪ B
return True
```

Backward coverability modulo $\mathbb{Q}$-coverability

```
if target marking m is not ℚ-coverable:
  return False
X = {target marking m}
while (initial marking m₀ not covered by X):
  B = markings obtained from X one step backward
  B = B \ {b ∈ B : ¬φ(b)}
  if B = ∅: return False
  φ(x) = φ(x) ∧ ⋀_{pruned b} x ≱ b
  X = X ∪ B
return True
```

SMT solver guidance

Backward coverability modulo $\mathbb{Q}$-coverability

```
if target marking m is not Q-coverable:
   return False
X = {target marking m}
while (initial marking m_0 not covered by X):
   B = markings obtained from X one step backward
   B = B \ {b ∈ B : ¬φ(b)}
   if B = ∅: return False
   φ(x) = φ(x) ∧ ⋀_{pruned b} x ≱ b
   X = X ∪ B
return True
```

Backward coverability modulo $\mathbb{Q}$-coverability

```
if target marking m is not ℚ-coverable:
    return False
X = {target marking m}
while (initial marking m₀ not covered by X):
    B = markings obtained from X one step backward
    B = B \ {b ∈ B : ¬φ(b)}
    if B = ∅: return False
    φ(x) = φ(x) ∧ ⋀_{pruned b} x ≱ b
    X = X ∪ B
return True
```

An implementation: QCOVER

python™

- 760 lines of code
- uses the MIST `.spec` format for counter machines
- supports dense/sparse matrices through NUMPY/SCIPY
- experimental parallelism support

An implementation: QCover

python™

- 760 lines of code
- uses the MIST `.spec` format for counter machines
- supports dense/sparse matrices through NumPy/SciPy
- experimental parallelism support

An implementation: QCOVER

python™

- 760 lines of code
- uses the MIST `.spec` format for counter machines
- supports dense/sparse matrices through NUMPY/SCIPY
- experimental parallelism support

An implementation: QCOVER

- 760 lines of code
- uses the MIST `.spec` format for counter machines
- supports dense/sparse matrices through NUMPY/SCIPY
- experimental parallelism support

An implementation: QCOVER

python™

- 760 lines of code
- uses the MIST `.spec` format for counter machines
- supports dense/sparse matrices through NUMPY/SCIPY
- experimental parallelism support

An implementation: QCOVER

python™

- 760 lines of code
- uses the MIST `.spec` format for counter machines
- supports dense/sparse matrices through NUMPY/SCIPY
- experimental parallelism support

SMT solver: Z3 (Microsoft research)

- FO($\mathbb{Q}_{\geq 0}, +, <$) formula satisfiability
- Fraca & Haddad "polynomial time" algorithm
  (rational linear programming with <)

An implementation: QCOVER

python™

- 760 lines of code
- uses the MIST `.spec` format for counter machines
- supports dense/sparse matrices through NUMPY/SCIPY
- experimental parallelism support

SMT solver: Z3 (Microsoft research)

- FO($\mathbb{Q}_{\geq 0}, +, <$) formula satisfiability
- Fraca & Haddad "polynomial time" algorithm
  (rational linear programming with <)

An implementation: QCOVER

python™

- 760 lines of code
- uses the MIST `.spec` format for counter machines
- supports dense/sparse matrices through NUMPY/SCIPY
- experimental parallelism support

SMT solver: Z3 (Microsoft research)

- FO($\mathbb{Q}_{\geq 0}, +, <$) formula satisfiability
- Fraca & Haddad "polynomial time" algorithm
  (rational linear programming with <)

Benchmarks

QCover tested against

- MIST: Ganty, Meuter, Delzanno, Kalyon, Raskin & Van Begin '07
- BFC: Kaiser, Kroening & Wahl '14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic '14

Benchmarks

QCover tested against

- MIST: Ganty, Meuter, Delzanno, Kalyon, Raskin & Van Begin '07
- BFC: Kaiser, Kroening & Wahl '14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic '14

Benchmarks

QCover tested against

- MIST: Ganty, Meuter, Delzanno, Kalyon, Raskin & Van Begin '07
- BFC: Kaiser, Kroening & Wahl '14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic '14

Benchmarks

QCover tested against

- MIST: Ganty, Meuter, Delzanno, Kalyon, Raskin & Van Begin '07
- BFC: Kaiser, Kroening & Wahl '14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic '14

Benchmarks

- 176 Petri nets: average of 1054 places & 8458 transitions
- Drawn from 5 existing suites

Benchmarks

QCover tested against

- MIST: Ganty, Meuter, Delzanno, Kalyon, Raskin & Van Begin '07
- BFC: Kaiser, Kroening & Wahl '14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic '14

Benchmarks

- 176 Petri nets: average of 1054 places & 8458 transitions
- Drawn from 5 existing suites including
  - Multithreaded C programs with shared memory (BFC)
  - Mutual exclusion, communication protocols, etc. (MIST)
  - Erlang concurrent programs (SOTER, D'Osualdo, Kochems & Ong '13)
  - Message analysis of a medical and a bug tracking system (Petrinizer)

Benchmarks

QCover tested against

- MIST: Ganty, Meuter, Delzanno, Kalyon, Raskin & Van Begin '07
- BFC: Kaiser, Kroening & Wahl '14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic '14

Benchmarks

- 176 Petri nets: average of 1054 places & 8458 transitions
- Drawn from 5 existing suites including
  - Multithreaded C programs with shared memory (BFC)
  - Mutual exclusion, communication protocols, etc. (MIST)
  - Erlang concurrent programs (SOTER, D'Osualdo, Kochems & Ong '13)
  - Message analysis of a medical and a bug tracking system (Petrinizer)

Benchmarks

QCover tested against

- MIST: Ganty, Meuter, Delzanno, Kalyon, Raskin & Van Begin '07
- BFC: Kaiser, Kroening & Wahl '14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic '14

Benchmarks

- 176 Petri nets: average of 1054 places & 8458 transitions
- Drawn from 5 existing suites including
  - Multithreaded C programs with shared memory (BFC)
  - Mutual exclusion, communication protocols, etc. (MIST)
  - Erlang concurrent programs (SOTER, D'Osualdo, Kochems & Ong '13)
  - Message analysis of a medical and a bug tracking system (Petrinizer)

Benchmarks

QCover tested against

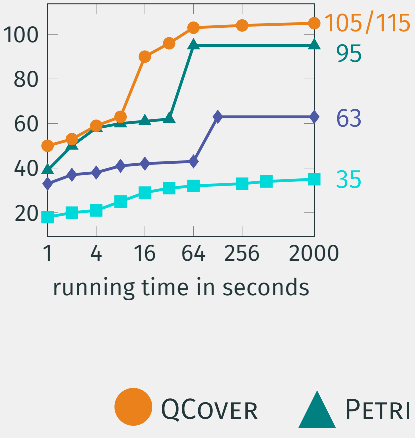
- MIST: Ganty, Meuter, Delzanno, Kalyon, Raskin & Van Begin '07
- BFC: Kaiser, Kroening & Wahl '14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic '14

Benchmarks

- 176 Petri nets: average of 1054 places & 8458 transitions
- Drawn from 5 existing suites including
  - Multithreaded C programs with shared memory (BFC)
  - Mutual exclusion, communication protocols, etc. (MIST)
  - Erlang concurrent programs (SOTER, D'Osualdo, Kochems & Ong '13)
  - Message analysis of a medical and a bug tracking system (Petrinizer)

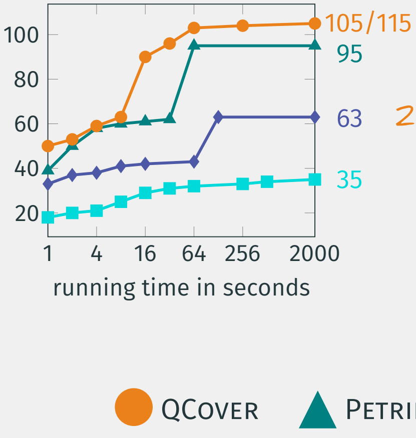
Benchmarks

Instances proven safe



Benchmarks

Instances proven safe



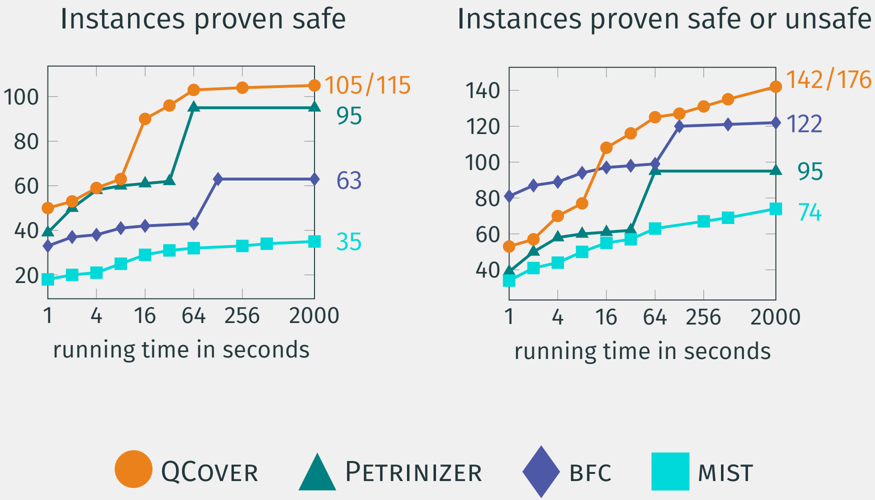
Largest nets proved safe:

21143 places
7150 trans.
42 secs.

6690 places
11934 trans.
21 secs.

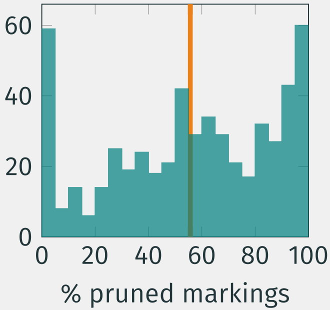
754 places
27370 trans.
3 secs.

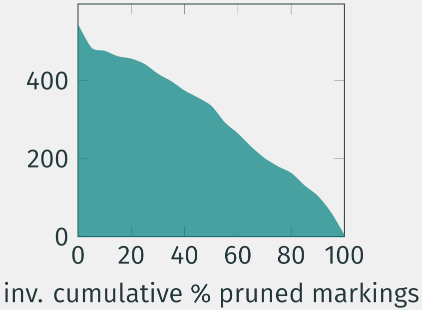
Benchmarks


Instances proven safe
100
80
60
40
20
1
4
16
64
256
2000
running time in seconds
105/115
95
63
35
Instances proven safe or unsafe
140
120
100
80
60
40
1
4
16
64
256
2000
running time in seconds
142/176
122
95
74
QCover
Petrinizer
BFC
MIST

Benchmarks

Markings pruning efficiency across all iterations



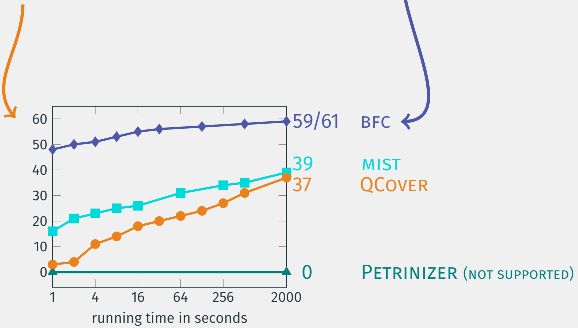


Future work

- Combine our approach with a forward algorithm to better handle unsafe instances

Future work

- Combine our approach with a forward algorithm to better handle unsafe instances



Future work

- Combine our approach with a forward algorithm to better handle unsafe instances
- Use more efficient data structures, *e.g.* sharing trees

(Delzanno, Raskin & Van Begin '04)

Future work

- Combine our approach with a forward algorithm to better handle unsafe instances
- Use more efficient data structures, *e.g.* sharing trees

  (Delzanno, Raskin & Van Begin '04)
- Extend to Petri nets with transfer/reset arcs

Thank you! Dank u!