# Approaching the Coverability Problem Continuously

Michael Blondin

Joint work with Alain Finkel, Christoph Haase, Serge Haddad

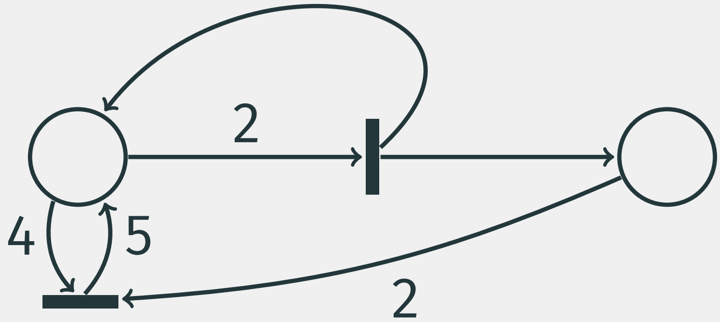
université PARIS-SACLAY

ENS CACHAN

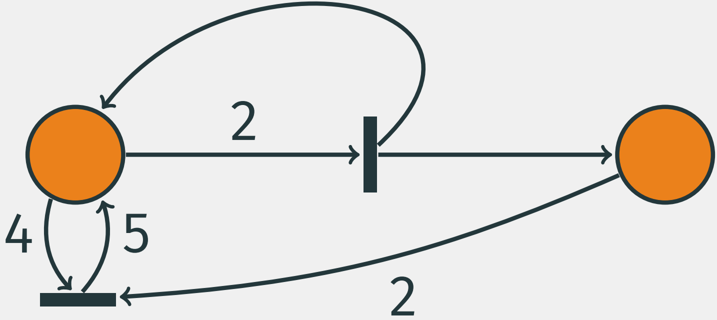
CNRS

Université de Montréal

# (Discrete) Petri nets

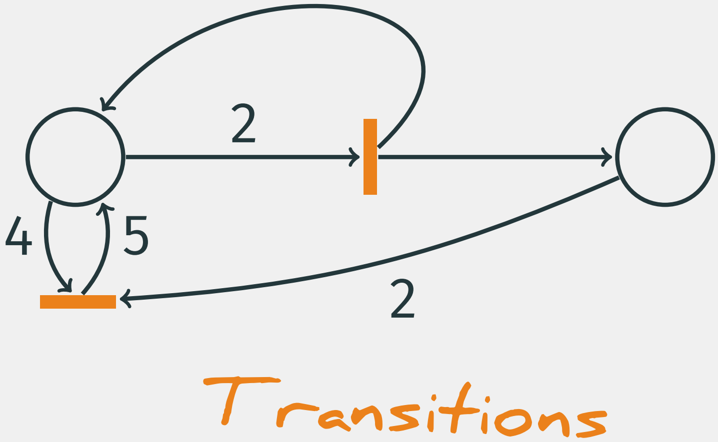


# (Discrete) Petri nets

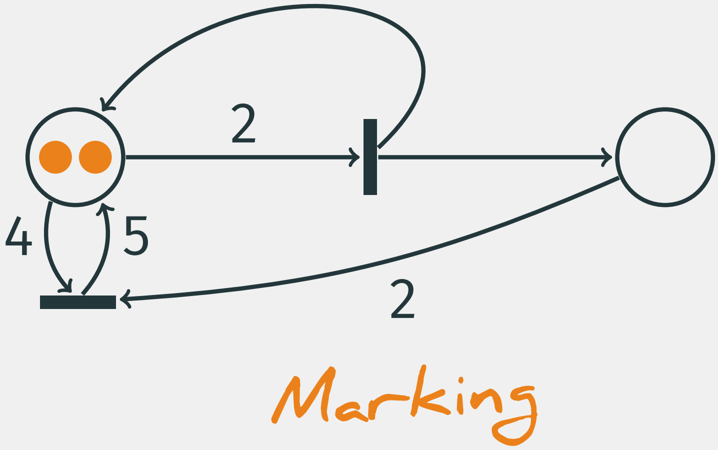



Places

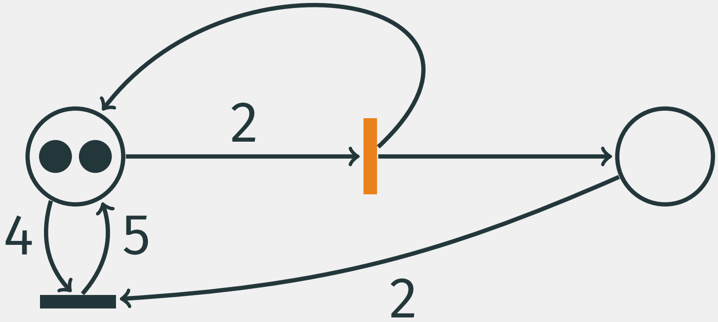
# (Discrete) Petri nets



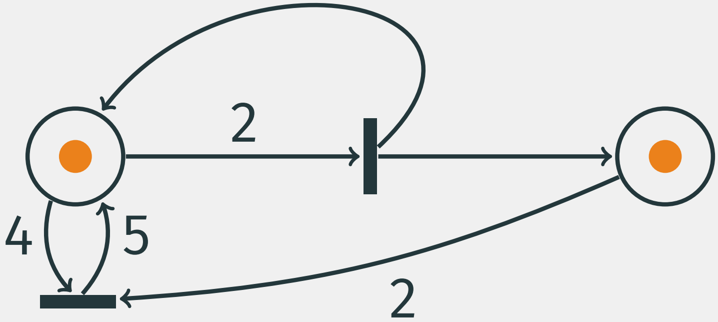
# (Discrete) Petri nets



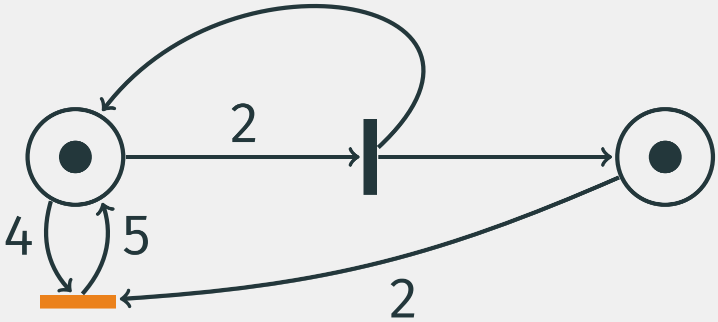
# (Discrete) Petri nets



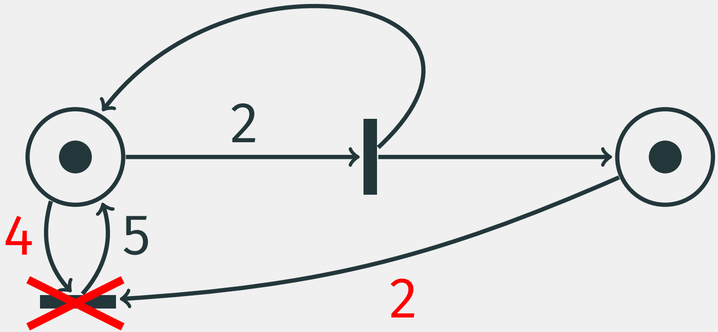
# (Discrete) Petri nets



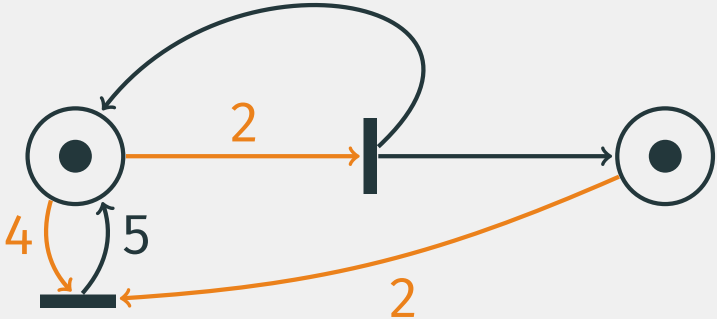
# (Discrete) Petri nets



# (Discrete) Petri nets

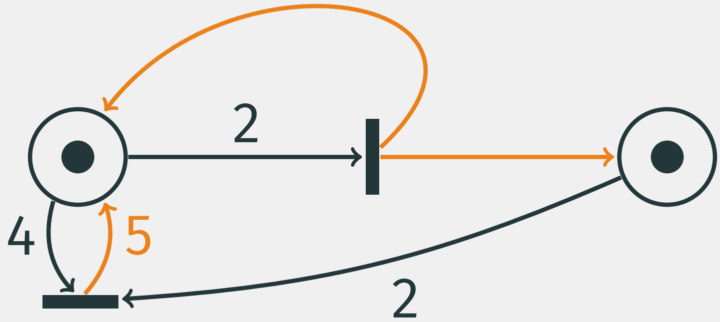


# (Discrete) Petri nets




$$Pre = \begin{pmatrix} 4 & 2 \\ 2 & 0 \end{pmatrix}$$

# (Discrete) Petri nets




$$Post = \begin{pmatrix} 5 & 1 \\ 0 & 1 \end{pmatrix}$$

Lamport mutual exclusion "1-bit algorithm"

Process 1

Process 2

Lamport mutual exclusion "1-bit algorithm"

# Verifying safety with Petri nets


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
```

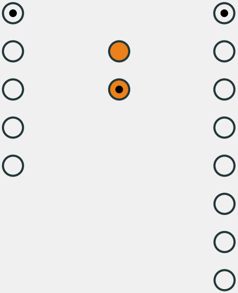
# Verifying safety with Petri nets

```
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

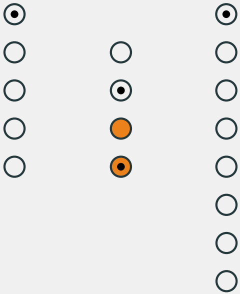
```
while True:
  x = True
  while y: pass
  # critical section
  x = False
```



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

# Verifying safety with Petri nets

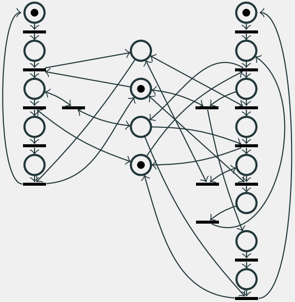
```
while True:
  x = True
  while y: pass
  # critical section
  x = False
```



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

# Verifying safety with Petri nets

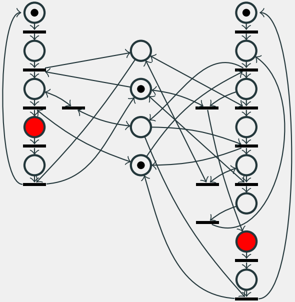
```
while True:
  x = True
  while y: pass
  # critical section
  x = False
```



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

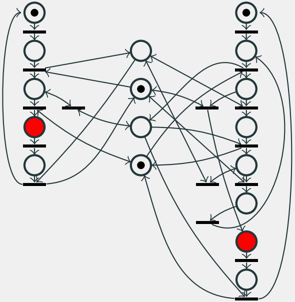
# Verifying safety with Petri nets

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



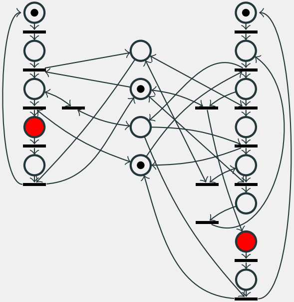
```
while True:
★ y = True
  if x then:
    y = False
    while x: pass
    goto ★
  # critical section
  y = False
```

# Verifying safety with Petri nets



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

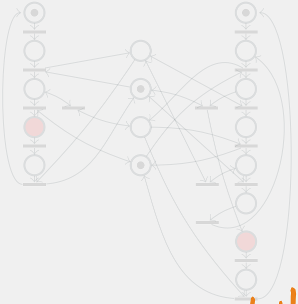
# Verifying safety with Petri nets



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

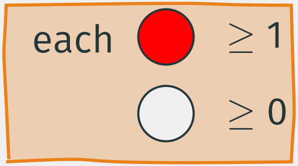
 $\geq 0$

# Verifying safety with Petri nets



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 $\mathcal{N}$? |

EXPSPACE-complete

Lipton STOC'76, Rackoff TCS'78

**Problem**

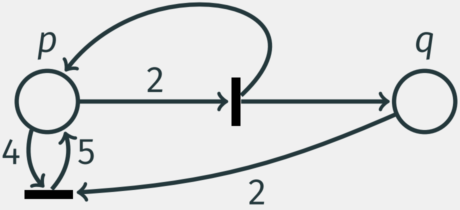
| | |
|---|---|
| Input: | Petri net $\mathcal{N}$, initial marking $\boldsymbol{m}_0$, target marking $\boldsymbol{m}$ |
| Question: | Is some $\boldsymbol{m}' \geq \boldsymbol{m}$ reachable from $\boldsymbol{m}_0$ in $\mathcal{N}$? |

# EXPSPACE-complete

Lipton STOC'76, Rackoff TCS'78

How can we be more efficient?

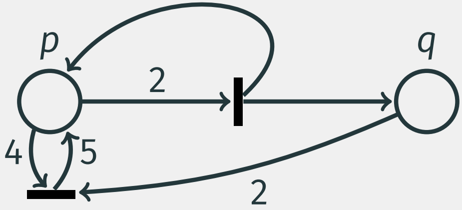
# Over-approximating coverability: state equation




$\boldsymbol{m}_0$ can cover $\boldsymbol{m} \implies$

$$\exists \boldsymbol{v} \geq \mathbf{0} \text{ s.t. } \boldsymbol{m}_0 + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v} \geq \boldsymbol{m}$$

# Over-approximating coverability: state equation

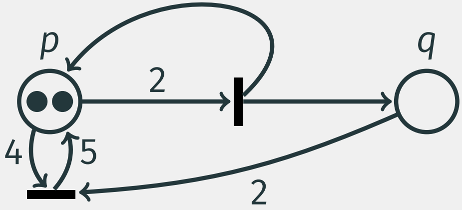



$$\boldsymbol{m}_0 \text{ can cover } \quad \boldsymbol{m} \quad \Longrightarrow$$

$$\exists \boldsymbol{v} \geq \boldsymbol{0} \text{ s.t. } \boldsymbol{m}_0 + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v} \geq \boldsymbol{m}$$

State equation

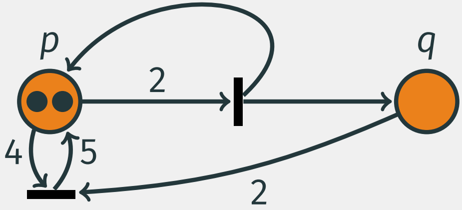
# Over-approximating coverability: state equation




$(2\ \ 0)$ can cover $\boldsymbol{m} \implies$

$$2 + x - y \geq \boldsymbol{m}(p)$$
$$0 - 2x + y \geq \boldsymbol{m}(q)$$

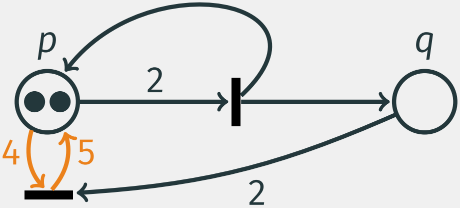
# Over-approximating coverability: state equation




$(2 \;\; 0)$ can cover $\boldsymbol{m} \implies$

$$\begin{aligned} 2 + x - y &\geq \boldsymbol{m}(p) \\ 0 - 2x + y &\geq \boldsymbol{m}(q) \end{aligned}$$

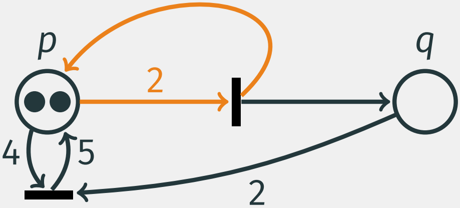
# Over-approximating coverability: state equation




$(2 \;\; 0)$ can cover $\boldsymbol{m}$ $\implies$

$$2 + x - y \geq \boldsymbol{m}(p)$$
$$0 - 2x + y \geq \boldsymbol{m}(q)$$

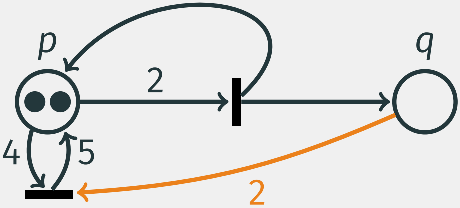
# Over-approximating coverability: state equation




$(2 \;\; 0)$ can cover $\boldsymbol{m} \implies$

$$2 + x - y \geq \boldsymbol{m}(p)$$
$$0 - 2x + y \geq \boldsymbol{m}(q)$$

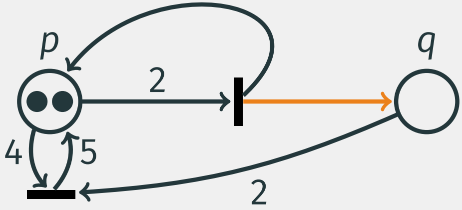
# Over-approximating coverability: state equation




$(2 \;\; 0)$ can cover $\boldsymbol{m} \implies$

$$2 + x - y \geq \boldsymbol{m}(p)$$
$$0 - 2x + y \geq \boldsymbol{m}(q)$$

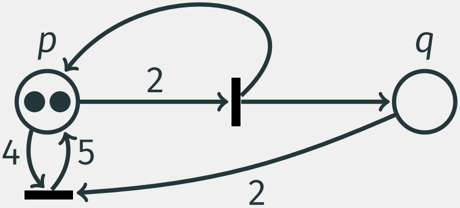
# Over-approximating coverability: state equation




$(2\ \ 0)$ can cover $\boldsymbol{m}$ $\Longrightarrow$

$$2 + x - y \geq \boldsymbol{m}(p)$$
$$0 - 2x + y \geq \boldsymbol{m}(q)$$

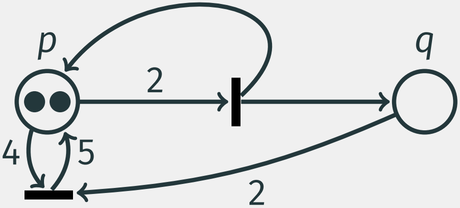
# Over-approximating coverability: state equation




$(2 \;\; 0)$ can cover $(0 \;\; 3) \Longrightarrow$

$$\begin{aligned} 2 + x - y &\geq 0 \\ 0 - 2x + y &\geq 3 \end{aligned}$$

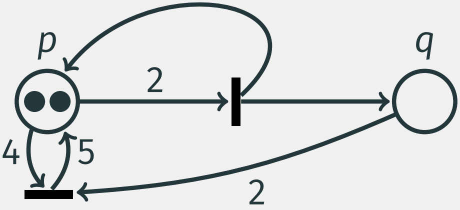
# Over-approximating coverability: state equation




$(2\ \ 0)$ can cover $(0\ \ 3) \Longrightarrow$

$$\begin{aligned} 2 + x - y &\geq 0 \\ 0 - 2x + y &\geq 3 \end{aligned} \quad ✗$$

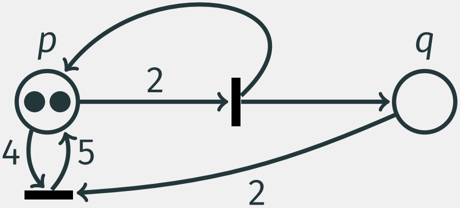
# Over-approximating coverability: state equation




$(2 \;\; 0)$ can cover $(0 \;\; 2) \implies$

$$
\begin{aligned}
2 + x - y &\geq 0 \\
0 - 2x + y &\geq 2
\end{aligned}
$$

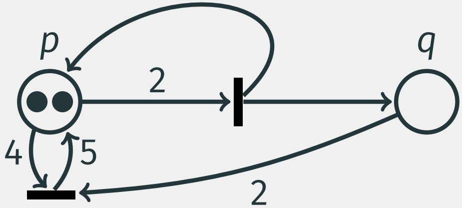
# Over-approximating coverability: state equation




$(2 \ \ 0)$ can cover $(0 \ \ 2) \Longrightarrow$

$$\begin{aligned} 2 + 0 - 2 &\geq 0 \\ 0 - 0 + 2 &\geq 2 \end{aligned} \quad \checkmark$$

# Over-approximating coverability: state equation



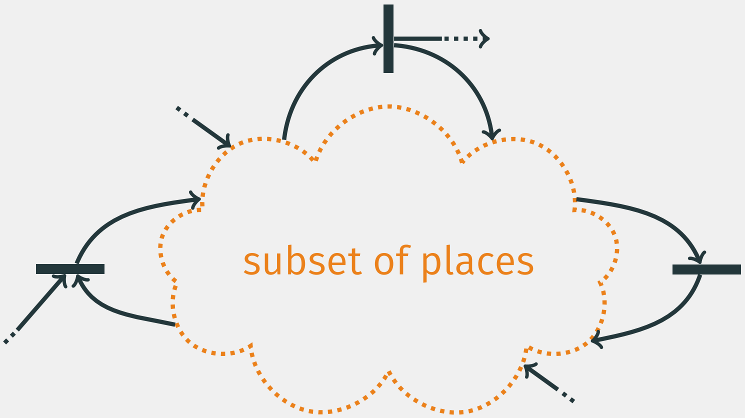

$(2 \;\; 0)$ can cover $(0 \;\; 2) \Longrightarrow$

Yet, not coverable...

$$2 + 0 - 2 \geq 0$$
$$0 - 0 + 2 \geq 2$$

✓

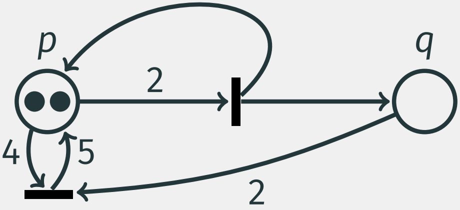
# Traps



# Traps

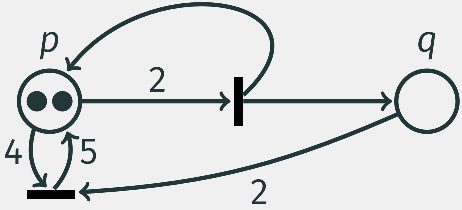
Once trap has tokens, it will always have tokens

# Over-approximating coverability: trap constraints




$(2 \;\; 0)$ can cover $(0 \;\; 2)$?

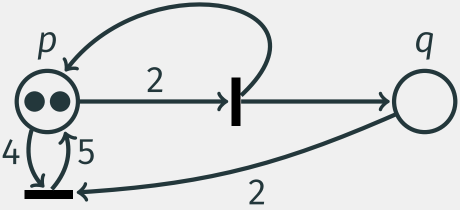
# Over-approximating coverability: trap constraints




$(2 \;\; 0)$ can cover $(0 \;\; 2)$?

$$2 + \;\; x - y \geq 0$$
$$0 - 2x + y \geq 2$$

# Over-approximating coverability: trap constraints

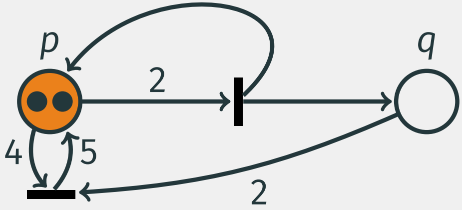



$(2 \;\; 0)$ can cover $(0 \;\; 2)$?

$$2 + \quad x - y = 0$$

$$0 - \; 2x + y = 2$$

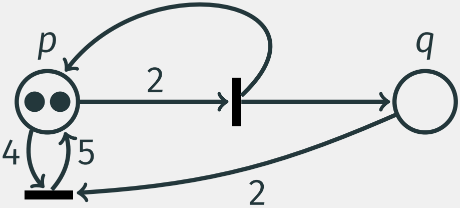
# Over-approximating coverability: trap constraints




$(2 \;\; 0)$ can cover $(0 \;\; 2)$?

$$2 + \quad x - y = 0$$
$$0 - 2x + y = 2$$

# Over-approximating coverability: trap constraints

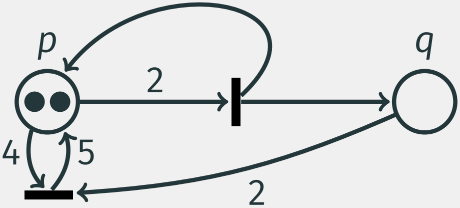



$(2 \;\; 0)$ can cover $(0 \;\; 2)$?

State equation ✓

Trap constraints ✗

## Over-approximating coverability: trap constraints

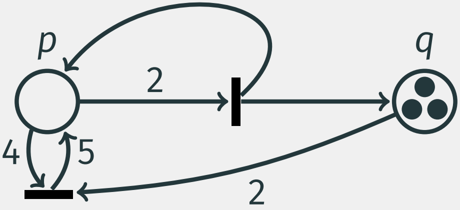



$(2 \;\; 0)$ can cover $(0 \;\; 2)$? *No!*

State equation ✓

Trap constraints ✗

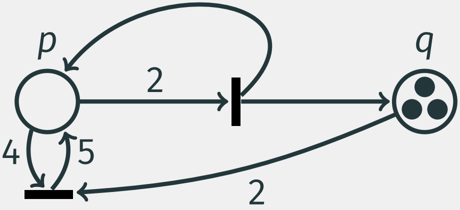
# Over-approximating coverability: trap constraints




$\begin{pmatrix}0 & 3\end{pmatrix}$ can cover $\begin{pmatrix}1 & 1\end{pmatrix}$?

$$0 + \quad x - y \geq 1$$
$$3 - 2x + y \geq 1$$

# Over-approximating coverability: trap constraints

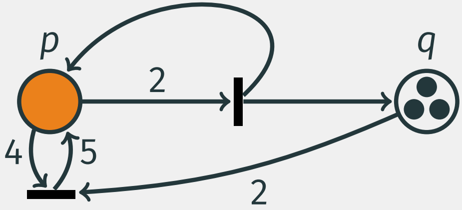



$(0 \;\; 3)$ can cover $(1 \;\; 1)$?

$$0 + \quad 1 - 0 \geq 1$$

$$3 - 2 \cdot 1 + 0 \geq 1$$

# Over-approximating coverability: trap constraints

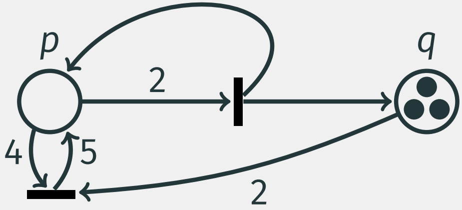



$(0 \;\; 3)$ can cover $(1 \;\; 1)$?

$$0 + \quad 1 - 0 \geq 1$$

$$3 - 2 \cdot 1 + 0 \geq 1$$

# Over-approximating coverability: trap constraints

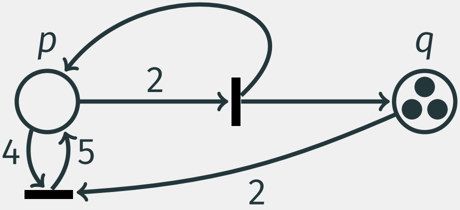



$(0 \;\; 3)$ can cover $(1 \;\; 1)$?

State equation ✓

Trap constraints ✓

# Over-approximating coverability: trap constraints




(0  3) can cover (1  1)?

Yet, not coverable...

State equation ✓

Trap constraints ✓

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

$$\Downarrow$$

$\boldsymbol{m}_0, \boldsymbol{m}$ satisfy

state equation and trap constraints

# Deciding safety with over-approximations

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

Safety

$\Uparrow$

$\boldsymbol{m}_0, \boldsymbol{m}$ do not satisfy

state equation or trap constraints

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

$\Uparrow$

$\boldsymbol{m}_0, \boldsymbol{m}$ do not satisfy

state equation or trap constraints

Efficient in practice!

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

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

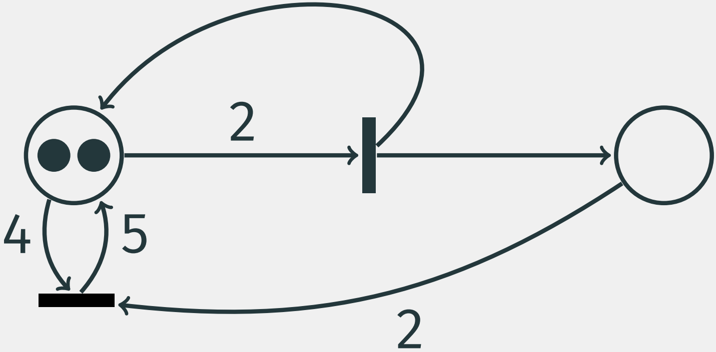
$$\Uparrow$$

$\boldsymbol{m}_0, \boldsymbol{m}$ do not satisfy

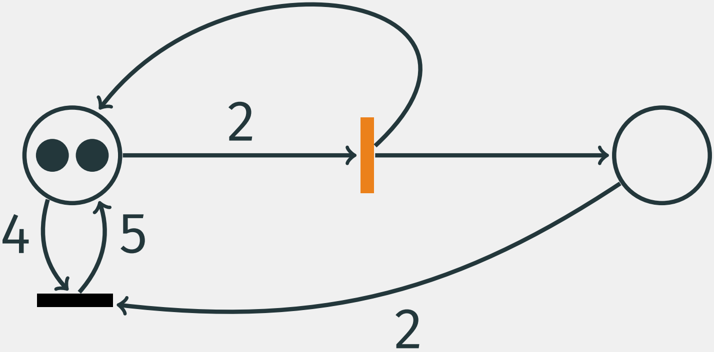
state equation or trap constraints

Any finer approximation, yet efficient?

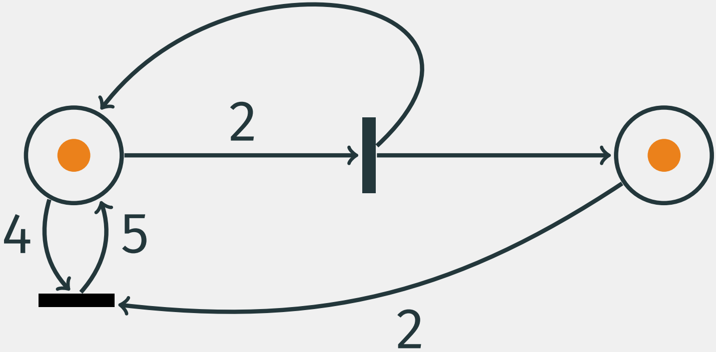
# (Discrete) Petri nets



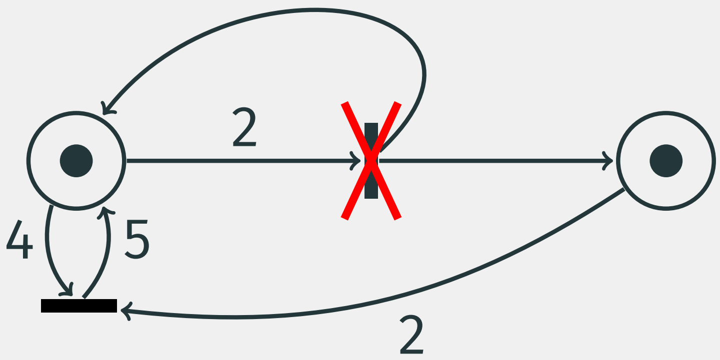
# (Discrete) Petri nets



# (Discrete) Petri nets

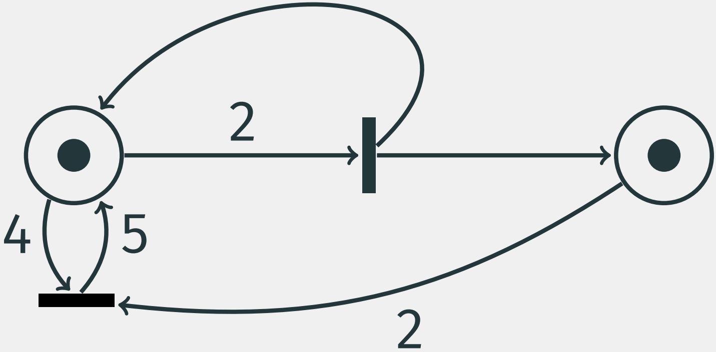


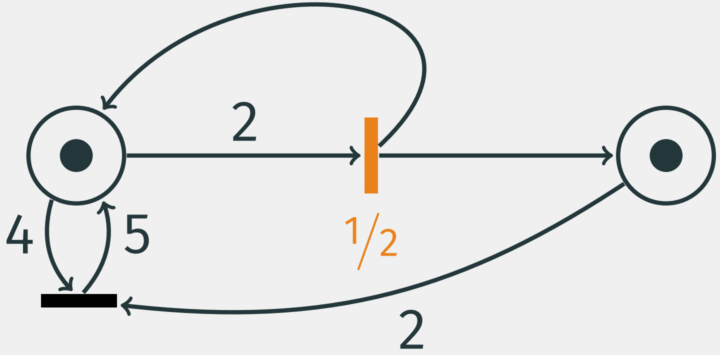
# (Discrete) Petri nets

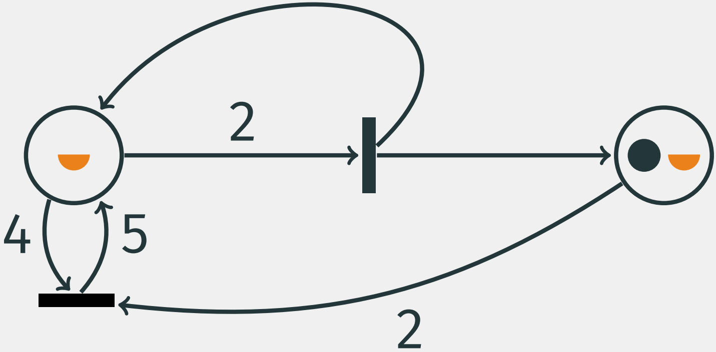


# ~~(Discrete)~~ Continuous Petri nets



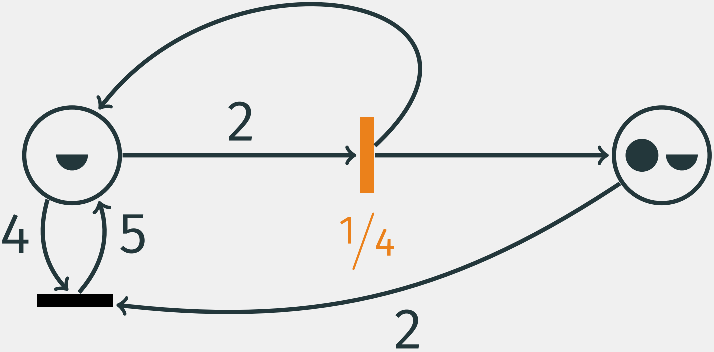


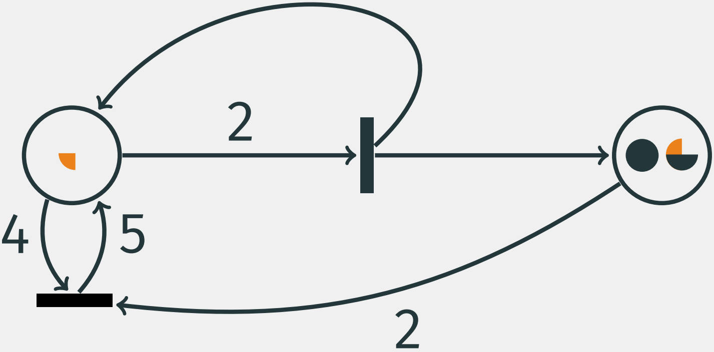

2
4
5
1/2
2

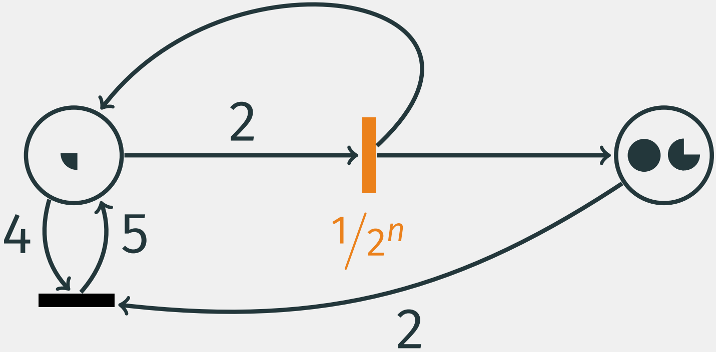

2
4
5
2

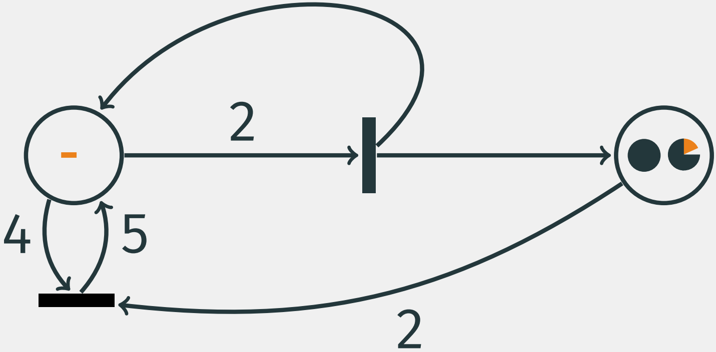
# ~~(Discrete)~~ Continuous Petri nets

David & Alla PN'87




2
4
5
2


2
4
5
1/2^n
2


2
4
5
2

# Continuity to over-approximate coverability

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

$$\Downarrow$$

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

# Continuity to over-approximate coverability

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

EXPSPACE

$$\Downarrow$$

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

PTIME

$$\Downarrow \quad \not\Uparrow$$

$\boldsymbol{m}_0$ and $\boldsymbol{m}$ satisfy
state equation & trap constraints

Esparza & Melzer FMSD'00
Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic CAV'14

PTIME / NP / coNP

# Continuity to over-approximate coverability

$\boldsymbol{m}$ 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** Fundam. Inf.'15

## 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** Fundam. Inf.'15

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

- $\boldsymbol{m}' = \boldsymbol{m}_0 + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$

# 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** **Fundam. Inf.'15**

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

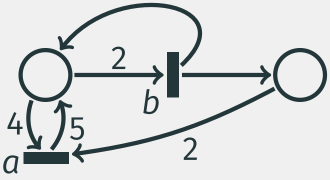
Fix some continuous Petri net $(P, T, \mathbf{Pre}, \mathbf{Post})$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** — **Fraca & Haddad** Fundam. Inf.'15

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 $\{t \in T : \boldsymbol{v}_t > 0\}$

# Coverability in continuous Petri nets



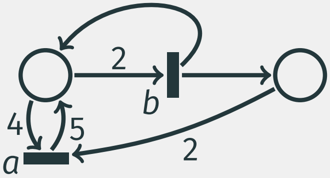

$$\boldsymbol{m}_0 = (0, 3)$$
$$\boldsymbol{m} = (1, 1)$$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** Fraca & Haddad Fundam. Inf.'15

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

- $\boldsymbol{m}' = \boldsymbol{m}_0 + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- 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



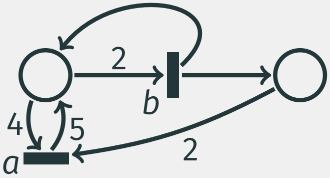

$$\boldsymbol{m}_0 = (0, 3)$$
$$\boldsymbol{m} = (1, 1)$$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** **Fraca & Haddad** Fundam. Inf.'15

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

- $0 + \boldsymbol{v}_a - \boldsymbol{v}_b \geq 1$
  $3 - 2\boldsymbol{v}_a + \boldsymbol{v}_b \geq 1$
- 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



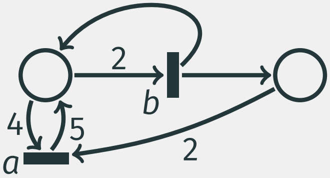

$$\boldsymbol{m}_0 = (0, 3)$$
$$\boldsymbol{m} = (1, 1)$$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** **Fraca & Haddad** **Fundam. Inf.'15**

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

- $\begin{array}{l} 0 + \boldsymbol{v}_a - \boldsymbol{v}_b \geq 1 \\ 3 - 2\boldsymbol{v}_a + \boldsymbol{v}_b \geq 1 \end{array} \implies \boldsymbol{v}_a = 1,\ \boldsymbol{v}_b = 0,\ \boldsymbol{m}' = \boldsymbol{m}$
- 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



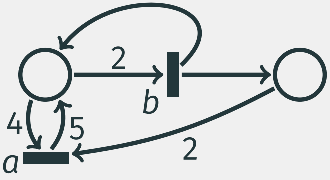

$$\boldsymbol{m}_0 = (0, 3)$$
$$\boldsymbol{m} = (1, 1)$$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** **Fraca & Haddad** Fundam. Inf.'15

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

- $\begin{array}{l} 0 + \boldsymbol{v}_a - \boldsymbol{v}_b \geq 1 \\ 3 - 2\boldsymbol{v}_a + \boldsymbol{v}_b \geq 1 \end{array} \implies \boldsymbol{v}_a = 1,\ \boldsymbol{v}_b = 0,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- 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



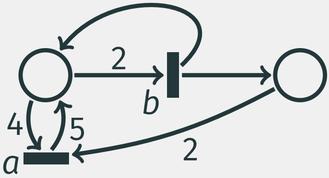

$$\mathbf{m}_0 = (0, 3)$$
$$\mathbf{m} = (1, 1)$$

**$\mathbf{m}$ is $\mathbb{Q}$-coverable from $\mathbf{m}_0$ iff...** Fraca & Haddad Fundam. Inf.'15

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

- $\begin{matrix} 0 + \mathbf{v}_a - \mathbf{v}_b \geq 1 \\ 3 - 2\mathbf{v}_a + \mathbf{v}_b \geq 1 \end{matrix} \implies \mathbf{v}_a = 1,\ \mathbf{v}_b = 0,\ \mathbf{m}' = \mathbf{m}$ ✓
- some execution from $\mathbf{m}_0$ fires exactly $\{t \in \{a, b\} : \mathbf{v}_t > 0\}$
- some execution to $\mathbf{m}'$ fires exactly $\{t \in \{a, b\} : \mathbf{v}_t > 0\}$

# Coverability in continuous Petri nets




$\boldsymbol{m}_0 = (0, 3)$

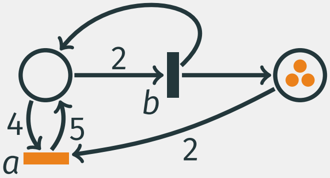
$\boldsymbol{m} = (1, 1)$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** **Fraca & Haddad** Fundam. Inf.'15

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

- $\begin{aligned} 0 + \boldsymbol{v}_a - \boldsymbol{v}_b &\geq 1 \\ 3 - 2\boldsymbol{v}_a + \boldsymbol{v}_b &\geq 1 \end{aligned} \implies \boldsymbol{v}_a = 1,\ \boldsymbol{v}_b = 0,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{a\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{a\}$

# Coverability in continuous Petri nets



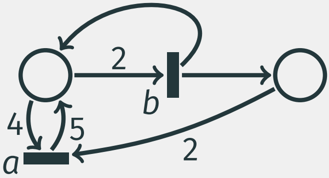

$\boldsymbol{m}_0 = (0, 3)$
$\boldsymbol{m} = (1, 1)$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** **Fraca & Haddad** Fundam. Inf.'15

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

- $\begin{array}{l} 0 + \boldsymbol{v}_a - \boldsymbol{v}_b \geq 1 \\ 3 - 2\boldsymbol{v}_a + \boldsymbol{v}_b \geq 1 \end{array} \implies \boldsymbol{v}_a = 1,\ \boldsymbol{v}_b = 0,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{a\}$
- some execution to $\boldsymbol{m}'$ fires exactly $\{a\}$

# Coverability in continuous Petri nets



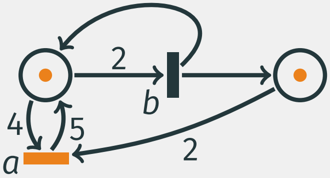

$$\boldsymbol{m}_0 = (0, 3)$$
$$\boldsymbol{m} = (1, 1)$$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** Fraca & Haddad Fundam. Inf.'15

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

- $\begin{array}{l} 0 + \boldsymbol{v}_a - \boldsymbol{v}_b \geq 1 \\ 3 - 2\boldsymbol{v}_a + \boldsymbol{v}_b \geq 1 \end{array} \implies \boldsymbol{v}_a = 1,\ \boldsymbol{v}_b = 0,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{a\}$ ✗
- some execution to $\boldsymbol{m}'$ fires exactly $\{a\}$

# Coverability in continuous Petri nets



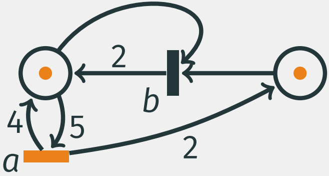

$$\boldsymbol{m}_0 = (0, 3)$$
$$\boldsymbol{m} = (1, 1)$$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** Fraca & Haddad Fundam. Inf.'15

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

- $\begin{array}{l} 0 + \boldsymbol{v}_a - \boldsymbol{v}_b \geq 1 \\ 3 - 2\boldsymbol{v}_a + \boldsymbol{v}_b \geq 1 \end{array} \implies \boldsymbol{v}_a = 1,\ \boldsymbol{v}_b = 0,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{a\}$ ✗
- some execution to $\boldsymbol{m}'$ fires exactly $\{a\}$

# Coverability in continuous Petri nets




$\boldsymbol{m}_0 = (0, 3)$

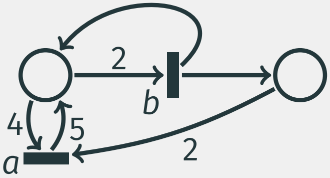
$\boldsymbol{m} = (1, 1)$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** **Fraca & Haddad Fundam. Inf.'15**

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

- $\begin{matrix} 0 + \boldsymbol{v}_a - \boldsymbol{v}_b \geq 1 \\ 3 - 2\boldsymbol{v}_a + \boldsymbol{v}_b \geq 1 \end{matrix} \implies \boldsymbol{v}_a = 1,\ \boldsymbol{v}_b = 0,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{a\}$ ✗
- some execution to $\boldsymbol{m}'$ fires exactly $\{a\}$

## Coverability in continuous Petri nets



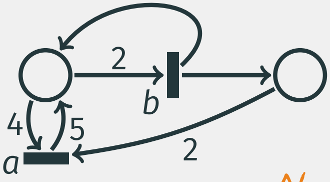

$\boldsymbol{m}_0 = (0, 3)$

$\boldsymbol{m} \; = (1, 1)$

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** **Fraca & Haddad** Fundam. Inf.'15

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

- $\begin{array}{l} 0 + \boldsymbol{v}_a - \boldsymbol{v}_b \geq 1 \\ 3 - 2\boldsymbol{v}_a + \boldsymbol{v}_b \geq 1 \end{array} \implies \boldsymbol{v}_a = 1,\ \boldsymbol{v}_b = 0,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{a\}$ ✗
- some execution to $\boldsymbol{m}'$ fires exactly $\{a\}$ ✓




$\boldsymbol{m}_0 = (0, 3)$

$\boldsymbol{m} = (1, 1)$

Not ℚ-coverable from

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** — Fraca & Haddad Fundam. Inf.'15

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

- $\begin{aligned} 0 + \boldsymbol{v}_a - \boldsymbol{v}_b &\geq 1 \\ 3 - 2\boldsymbol{v}_a + \boldsymbol{v}_b &\geq 1 \end{aligned} \implies \boldsymbol{v}_a = 1,\ \boldsymbol{v}_b = 0,\ \boldsymbol{m}' = \boldsymbol{m}$ ✓
- some execution from $\boldsymbol{m}_0$ fires exactly $\{a\}$ ✗
- some execution to $\boldsymbol{m}'$ fires exactly $\{a\}$ ✓

# Coverability in continuous Petri nets

Polynomial time!

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** **Fraca & Haddad** Fundam. Inf.'15

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** — B., Finkel, Haase & Haddad TACAS'16

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

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

***m* is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** — Fraca & Haddad Fundam. Inf.'15

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 $\{t \in T : \boldsymbol{v}_t > 0\}$

**Logical characterization** — B., Finkel, Haase & Haddad TACAS'16

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

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

↖ Even better approximation

**$\boldsymbol{m}$ is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** — Fraca & Haddad Fundam. Inf.'15

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 $\{t \in T : \boldsymbol{v}_t > 0\}$

# Coverability in continuous Petri nets

**Logical characterization** — B., Finkel, Haase & Haddad TACAS'16

$\mathbb{Q}$-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 Fundam. Inf.'15

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

- $\boldsymbol{m}' = \boldsymbol{m}_0 + (\textbf{Post} - \textbf{Pre}) \cdot \boldsymbol{v}$ ← 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** — B., Finkel, Haase & Haddad TACAS'16

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

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

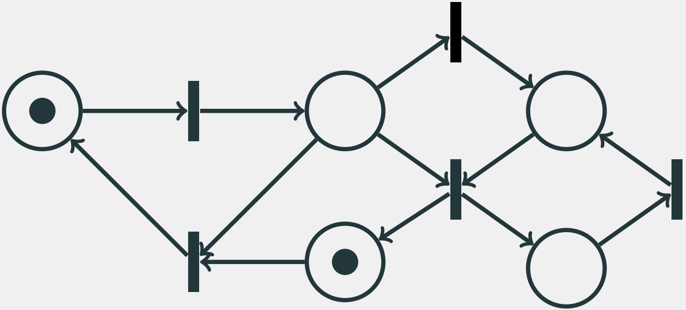
***m*** **is $\mathbb{Q}$-coverable from $\boldsymbol{m}_0$ iff...** — Fraca & Haddad Fundam. Inf.'15

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 $\{t \in T : \boldsymbol{v}_t > 0\}$

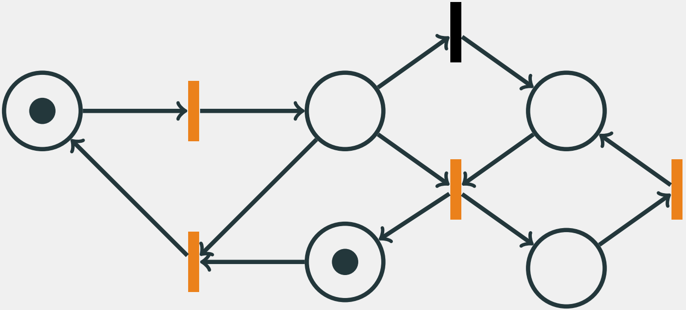
More subtle

# Encoding the firing set conditions



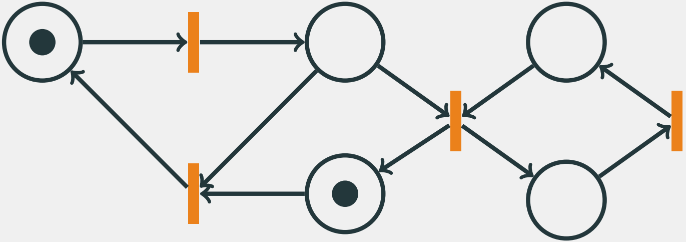
Testing whether some transitions can be fired from initial marking

# Encoding the firing set conditions



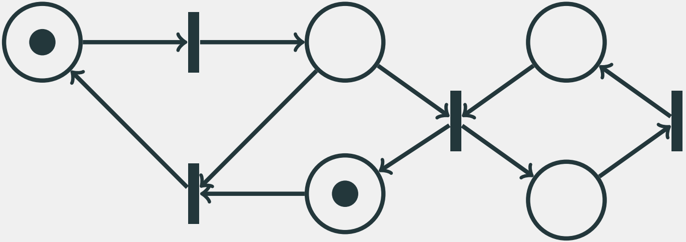
Testing whether some transitions can be fired from initial marking

# Encoding the firing set conditions



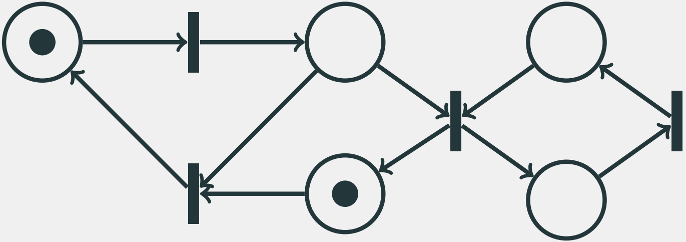
Testing whether some transitions can be fired from initial marking

# Encoding the firing set conditions



Simulate a "breadth-first" transitions firing

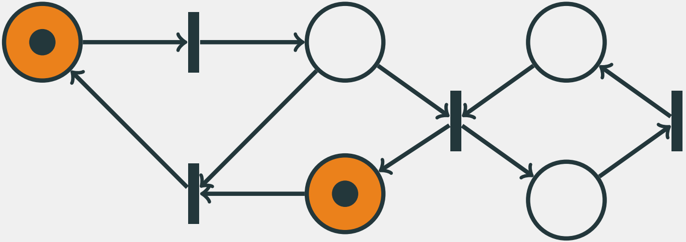
# Encoding the firing set conditions



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

Verma, Seidl & Schwentick CADE'05

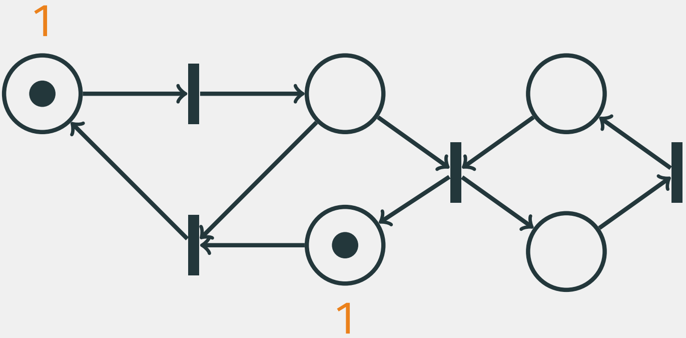
# Encoding the firing set conditions



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

Verma, Seidl & Schwentick CADE'05

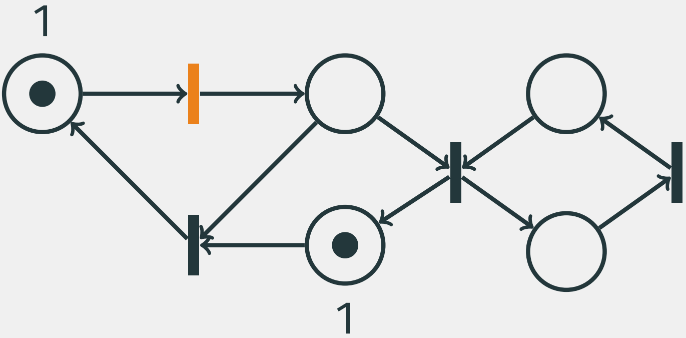
# Encoding the firing set conditions




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

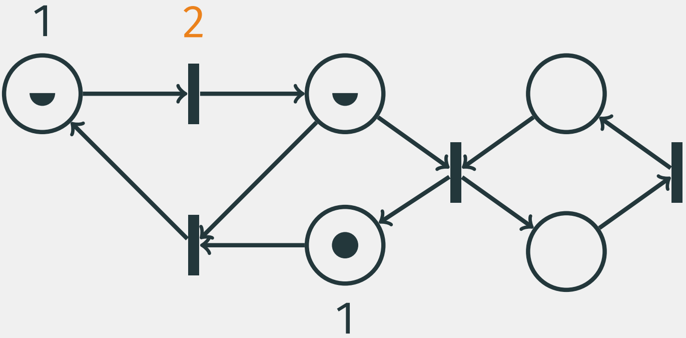
Verma, Seidl & Schwentick CADE'05

# Encoding the firing set conditions




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

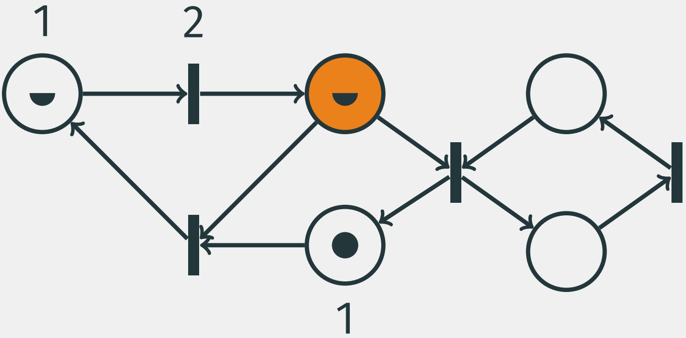
Verma, Seidl & Schwentick CADE'05




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

Verma, Seidl & Schwentick CADE'05

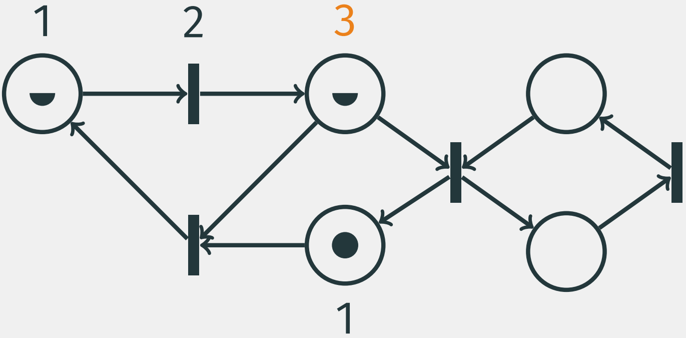
# Encoding the firing set conditions




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

Verma, Seidl & Schwentick CADE'05

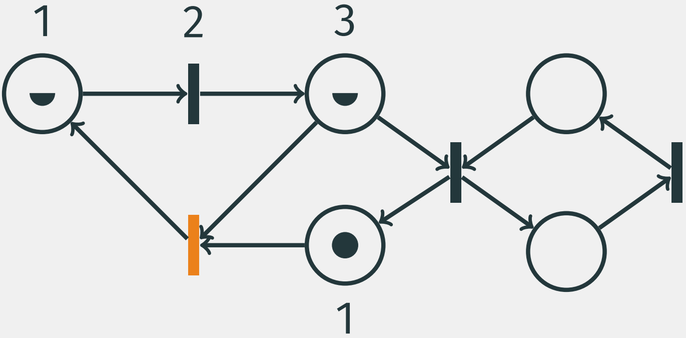
# Encoding the firing set conditions




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

Verma, Seidl & Schwentick CADE'05

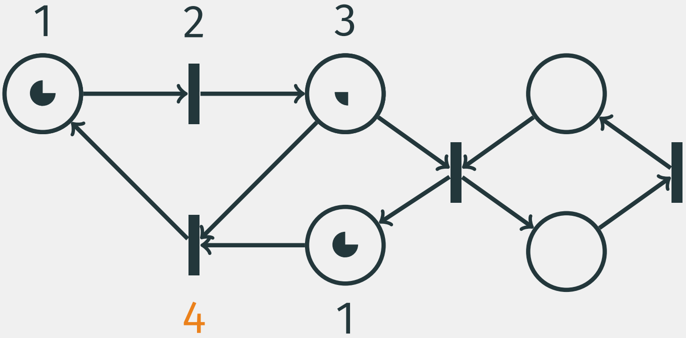
# Encoding the firing set conditions




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

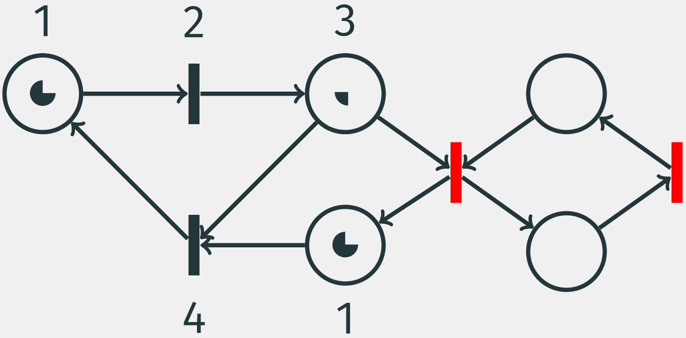
Verma, Seidl & Schwentick CADE'05

# Encoding the firing set conditions




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

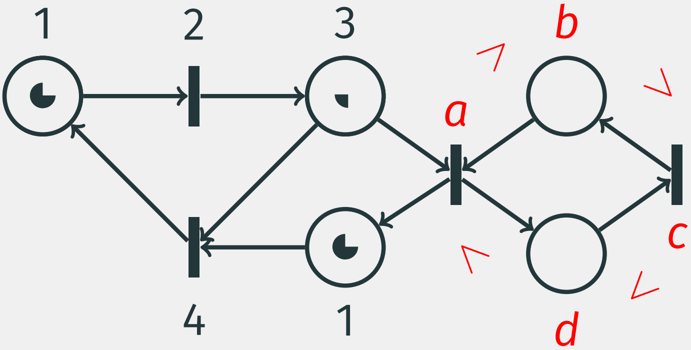
Verma, Seidl & Schwentick CADE'05




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

Verma, Seidl & Schwentick CADE'05

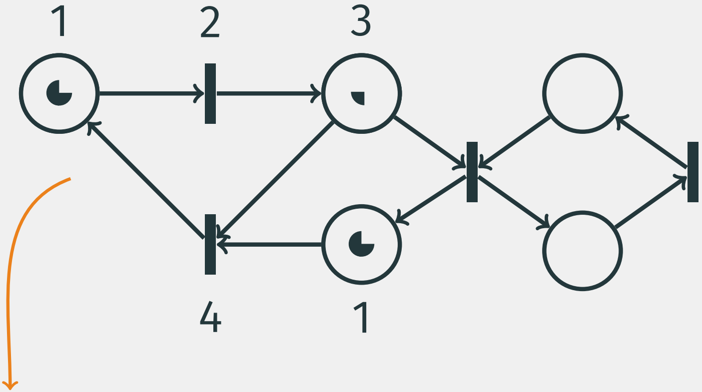
# Encoding the firing set conditions




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

Verma, Seidl & Schwentick CADE'05

# Encoding the firing set conditions




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

$\mathbb{Q}$-coverability: efficient but incomplete...

Combine approaches!

# From $\mathbb{Q}$-coverability to a complete algorithm

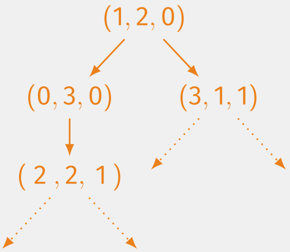
**Forward algorithm** **Karp & Miller** JCSS'69

- Build reachability tree from initial marking
- "Accelerate" loops

# From $\mathbb{Q}$-coverability to a complete algorithm

**Forward algorithm** **Karp & Miller JCSS'69**

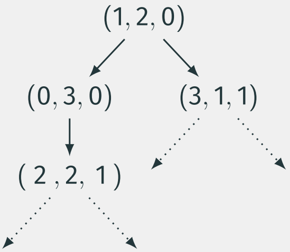
- Build reachability tree from initial marking
- "Accelerate" loops



# From $\mathbb{Q}$-coverability to a complete algorithm

**Forward algorithm** — **Karp & Miller** JCSS'69

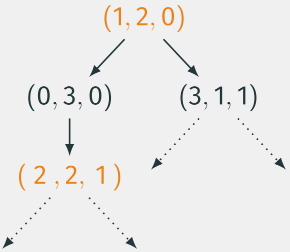
- Build reachability tree from initial marking
- "Accelerate" loops



# From $\mathbb{Q}$-coverability to a complete algorithm

**Forward algorithm** **Karp & Miller** JCSS'69

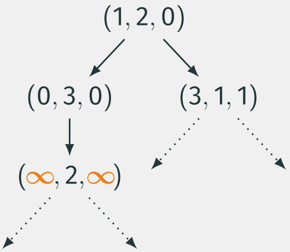
- Build reachability tree from initial marking
- "Accelerate" loops



# From $\mathbb{Q}$-coverability to a complete algorithm

**Forward algorithm** **Karp & Miller** JCSS'69

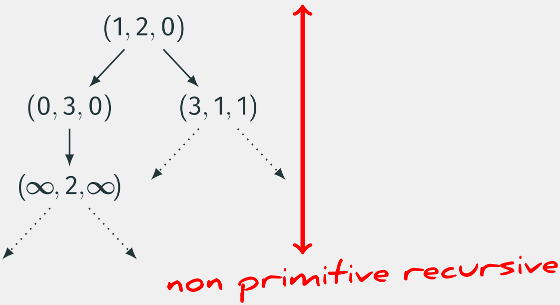
- Build reachability tree from initial marking
- "Accelerate" loops



# From $\mathbb{Q}$-coverability to a complete algorithm

**Forward algorithm** — **Karp & Miller** JCSS'69

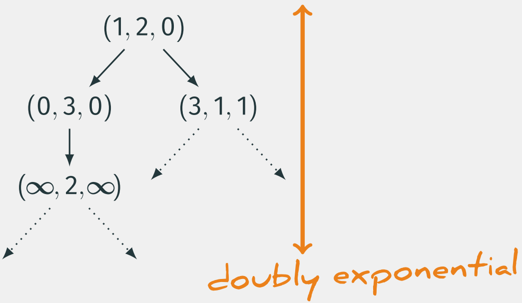
- Build reachability tree from initial marking
- "Accelerate" loops



# From $\mathbb{Q}$-coverability to a complete algorithm

**Forward algorithm** **Karp & Miller** JCSS'69

- Build reachability tree from initial marking
- "Accelerate" loops



# From $\mathbb{Q}$-coverability to a complete algorithm

**Forward algorithm** — **Karp & Miller** JCSS'69

- Build reachability tree from initial marking
- "Accelerate" loops

**Backward algorithm** — **Arnold & Latteux** Calcolo'78, **Abdulla, Cerans, Jonsson & Tsay** LICS'96

- Start from upward closure of target marking
- Compute predecessors of current markings

# From $\mathbb{Q}$-coverability to a complete algorithm

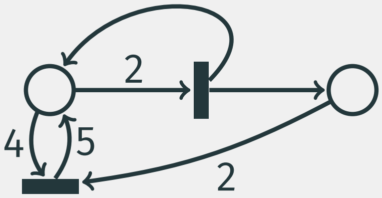
**Forward algorithm** — **Karp & Miller JCSS'69**

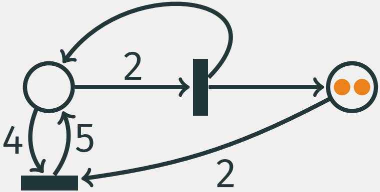
- Build reachability tree from initial marking
- "Accelerate" loops

**Backward algorithm** — **Arnold & Latteux Calcolo'78, Abdulla, Cerans, Jonsson & Tsay LICS'96**

- Start from upward closure of target marking
- Compute predecessors of current markings

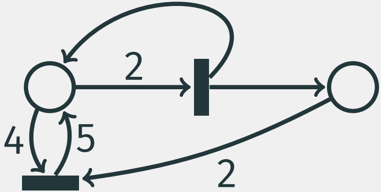
# Backward algorithm

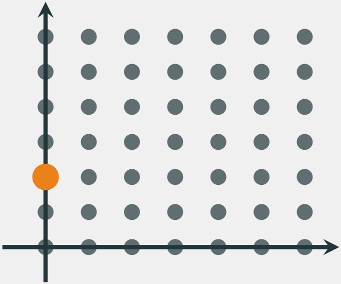




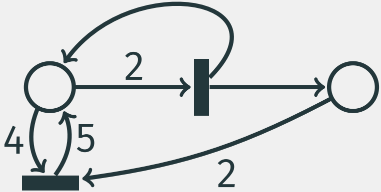

What initial markings may cover $(0, 2)$?

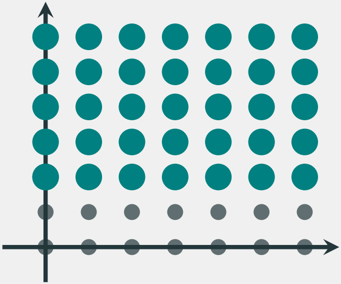
# Backward algorithm



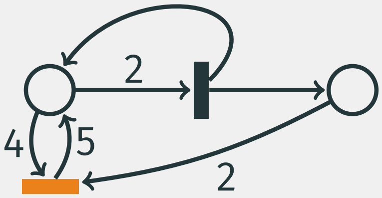



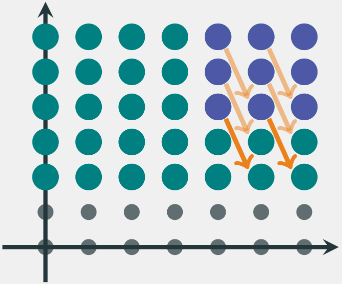
# Backward algorithm



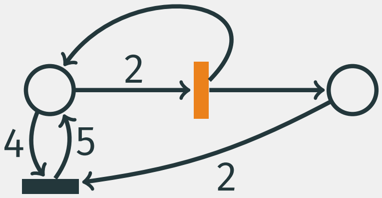



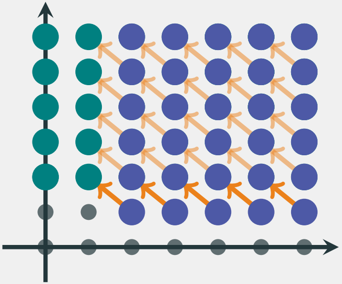
# Backward algorithm



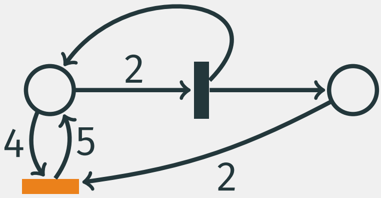



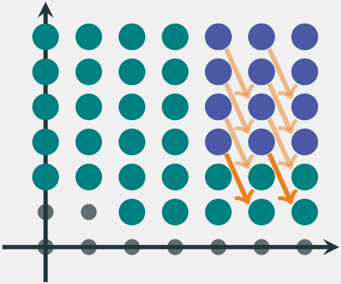
# Backward algorithm



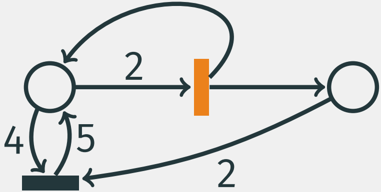



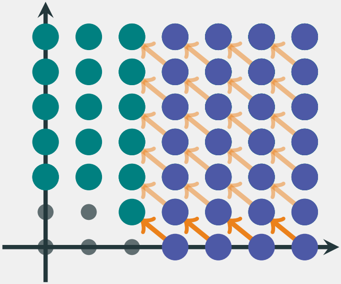
# Backward algorithm



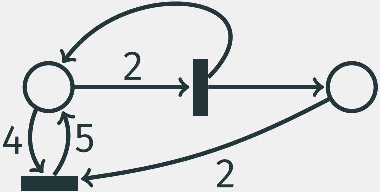



# Backward algorithm

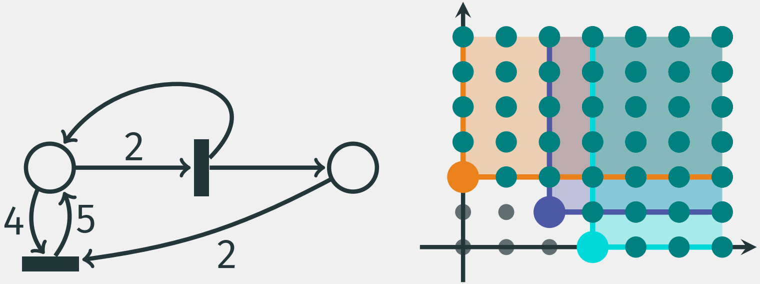


# Backward algorithm

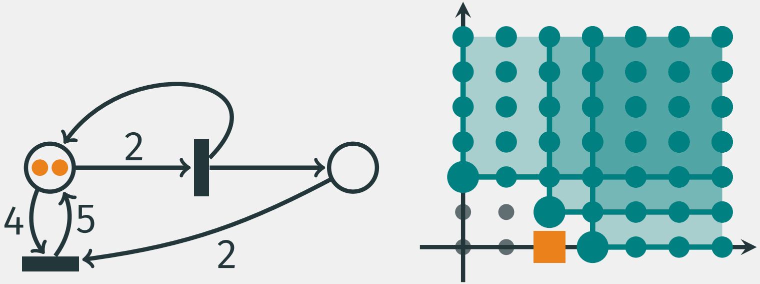



# Backward algorithm



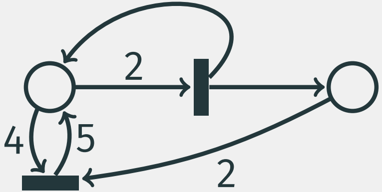

Basis size may become doubly exponential

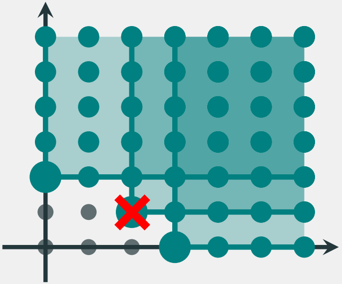
Bozzelli & Ganty RP'11




We only care about some initial marking...

# Backward algorithm



We only care about some initial marking...

Prune basis with $\mathbb{Q}$-coverability!

# 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
```

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

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

  $B =$ markings obtained from $X$ one step backward

  $B = B \setminus \{\boldsymbol{b} \in B : \neg\varphi(\boldsymbol{b})\}$

  `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
```

# 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)}     ← ℚ-coverability pruning (better than poly. time)
  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 $\boldsymbol{m}$ is not $\mathbb{Q}$-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

  $B = B \setminus \{\boldsymbol{b} \in B : \neg\varphi(\boldsymbol{b})\}$

  **if** $B = \emptyset$: **return** False   SMT solver guidance

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

  $X = X \cup B$

**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 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
```

# 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

python™

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

- 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 ACM TOPLAS'14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic CAV'14

# Benchmarks

QCover tested against

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

# Benchmarks

QCover tested against

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

# Benchmarks

QCover tested against

- MIST: Ganty, Meuter, Delzanno, Kalyon, Raskin & Van Begin '07
- BFC: Kaiser, Kroening & Wahl ACM TOPLAS'14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic CAV'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 ACM TOPLAS'14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic CAV'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 SAS'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 ACM TOPLAS'14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic CAV'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 SAS'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 ACM TOPLAS'14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic CAV'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 SAS'13)
  - Message analysis of a medical and a bug tracking system (Petrinizer)

# Benchmarks

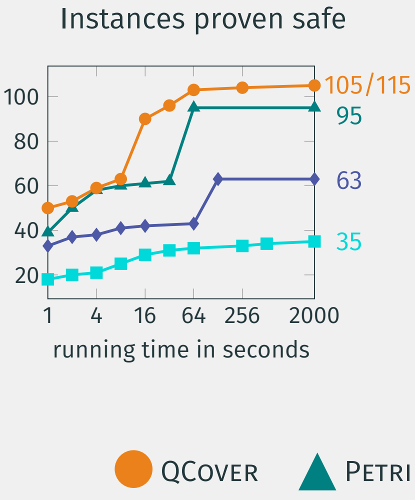
QCover tested against

- MIST: Ganty, Meuter, Delzanno, Kalyon, Raskin & Van Begin '07
- BFC: Kaiser, Kroening & Wahl ACM TOPLAS'14
- Petrinizer: Esparza, Ledesma-Garza, Majumdar, Meyer & Niksic CAV'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 SAS'13)
  - Message analysis of a medical and a bug tracking system (Petrinizer)

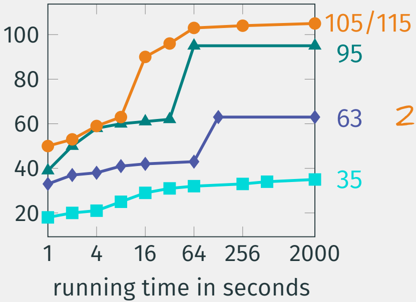
# Benchmarks



Instances proven safe
100
80
60
40
20
1
4
16
64
256
2000
running time in seconds
105/115
95
63
35
QCover
Petrinizer
BFC
MIST

# Benchmarks

Instances proven safe




Largest nets proved safe:

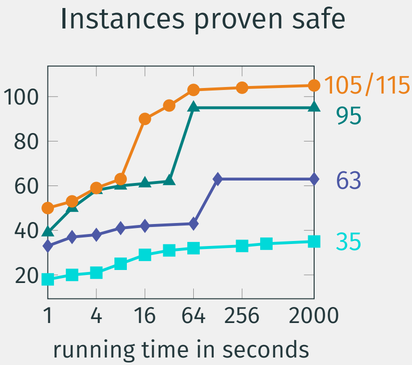
21143 places
7150 trans.
42 secs.

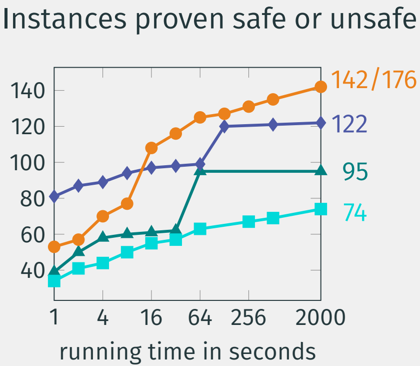
6690 places
11934 trans.
21 secs.

754 places
27370 trans.
3 secs.

QCover PETRINIZER BFC MIST

# Benchmarks

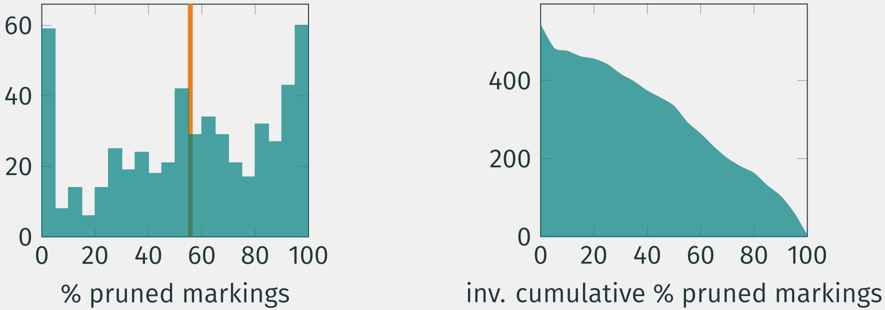

Instances proven safe
100
80
60
40
20
1
4
16
64
256
2000
running time in seconds
105/115
95
63
35


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

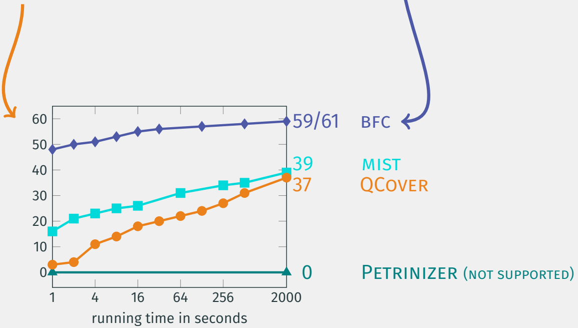


# Future work

- 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 STTT'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 STTT'04)
- Extend to Petri nets with transfer/reset arcs

# Thank you!
# Vielen Dank!