The Logical View on Continuous Petri Nets

Michael Blondin

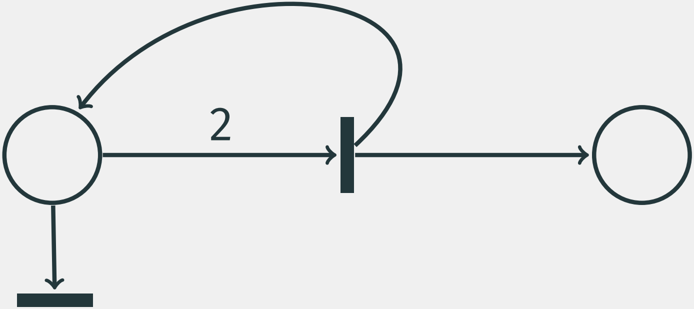
Joint work with Alain Finkel, Christoph Haase, Serge Haddad

The Logical View on Continuous Petri Nets

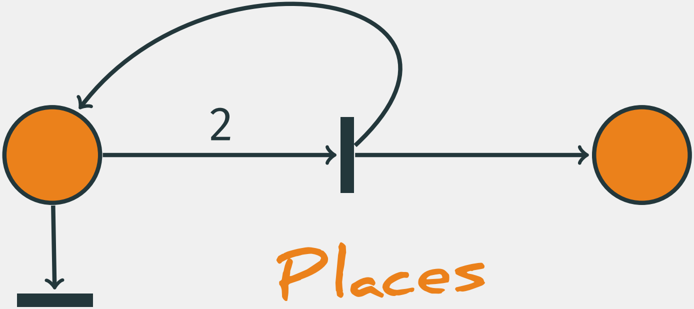
Michael Blondin

Joint work with Alain Finkel, Christoph Haase, Serge Haddad

Petri nets

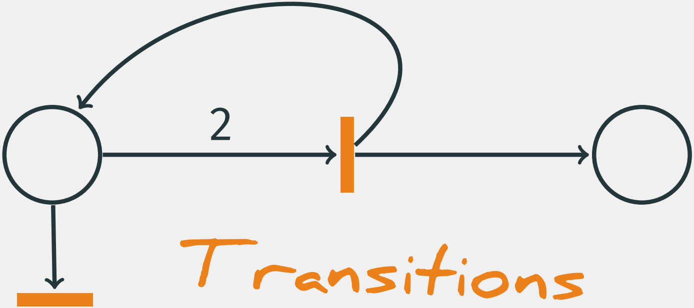


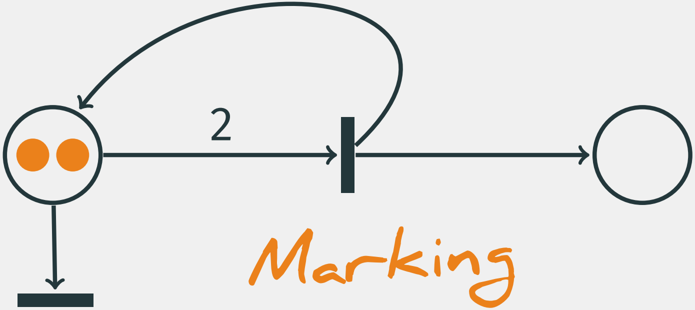
Petri nets



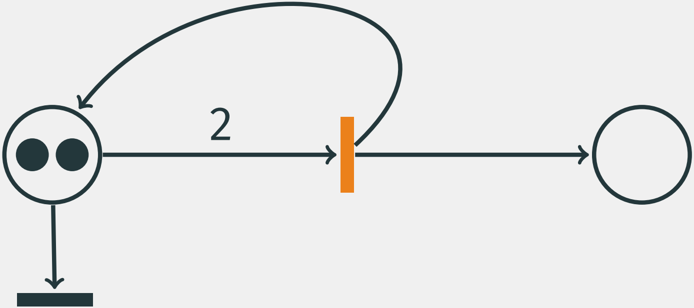
Places

Petri nets

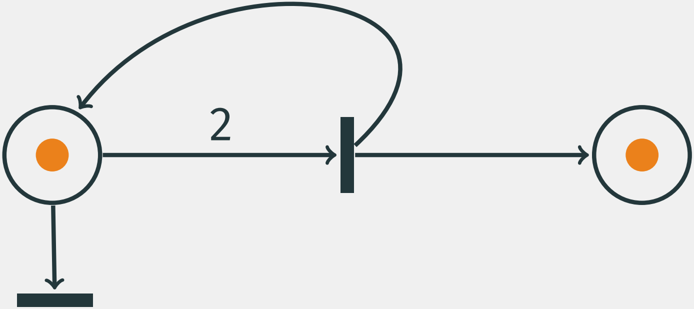



2
Marking

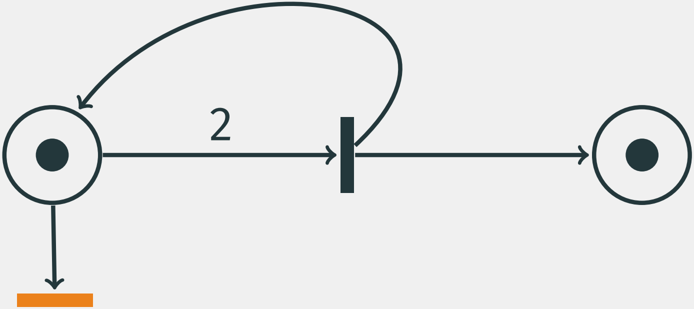
Petri nets



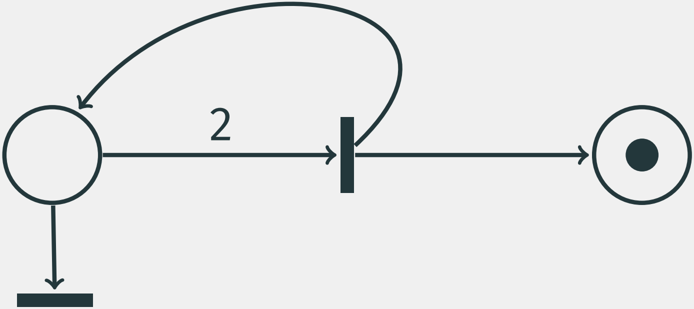
Petri nets



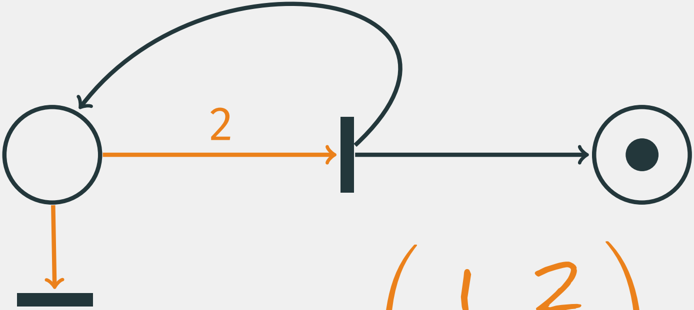
Petri nets



Petri nets

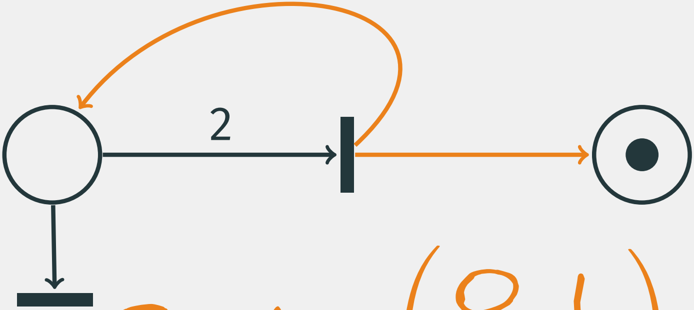


Petri nets



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

Petri nets



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

Safety verification with Petri nets

Process 1

Process 2

Safety verification with Petri nets

Process 1

critical section

Process 2

critical section

Safety verification 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
```

Lamport mutual exclusion algorithm

Safety verification 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
```

shared variables: x, y

Safety verification with Petri nets

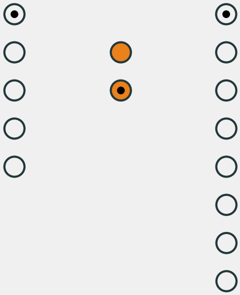
```
while true:
  x = true
  while y: pass
  # critical section
  x = false
```

```
while true:
★ y = true
  if x then:
    y = false
    while x: pass
    goto ★
  # critical section
  y = false
```

```
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
```

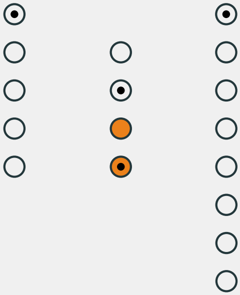
```
while true:
  x = true
  while y: pass
  # critical section
  x = false
```



```
while true:
★ y = true
  if x then:
    y = false
    while x: pass
    goto ★
  # critical section
  y = false
```

Safety verification with Petri nets

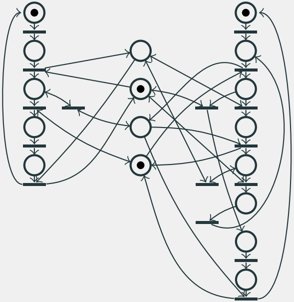
```
while true:
  x = true
  while y: pass
  # critical section
  x = false
```



```
   while true:
★   y = true
    if x then:
      y = false
      while x: pass
      goto ★
    # critical section
    y = false
```

Safety verification with Petri nets

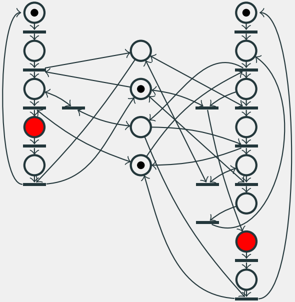
```
while true:
  x = true
  while y: pass
  # critical section
  x = false
```



```
while true:
★  y = true
   if x then:
     y = false
     while x: pass
     goto ★
   # critical section
   y = false
```

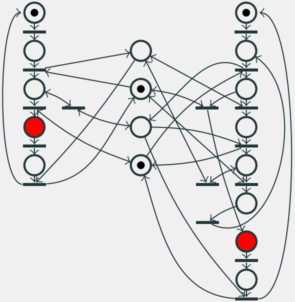
Safety verification with Petri nets

```
while true:
  x = true
  while y: pass
  # critical section
  x = false
```



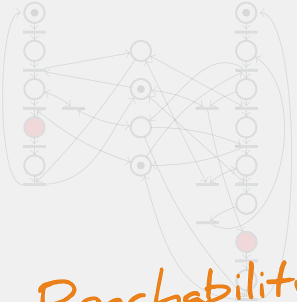
```
while true:
★  y = true
   if x then:
     y = false
     while x: pass
     goto ★
   # critical section
   y = false
```

Safety verification with Petri nets



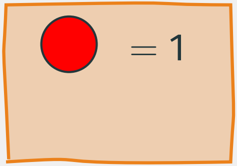
Processes at both critical sections $\iff$  $= 1$

Safety verification with Petri nets

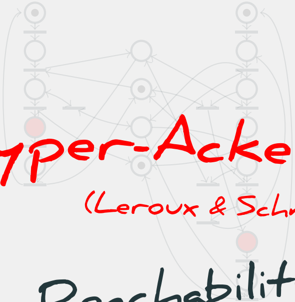


Reachability problem

Processes at both critical sections $\Longleftrightarrow$



Safety verification with Petri nets

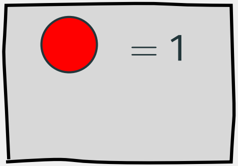


Hyper-Ackermannian!

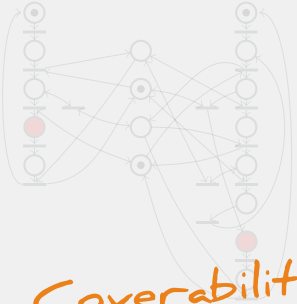
(Leroux & Schmitz '15)

Reachability problem

Processes at both critical sections $\iff$

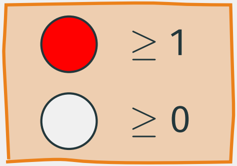


Safety verification with Petri nets



Coverability problem

Processes at both critical sections $\iff$

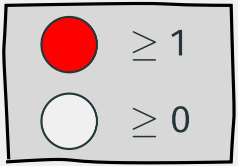


EXPSPACE-complete!

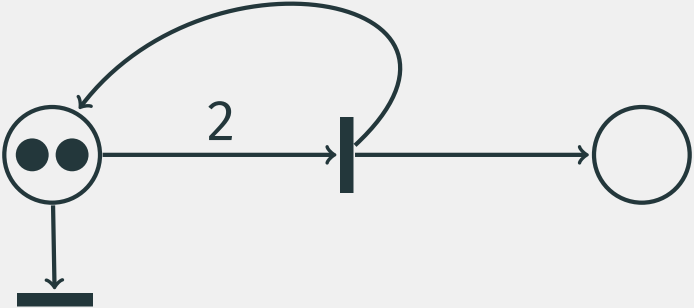
(Lipton '76, Rackoff '78)

Coverability problem

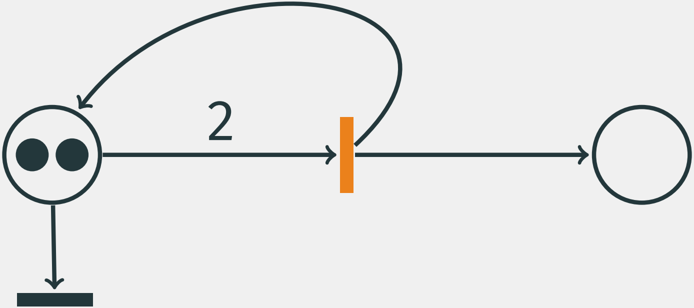
Processes at both critical sections $\iff$



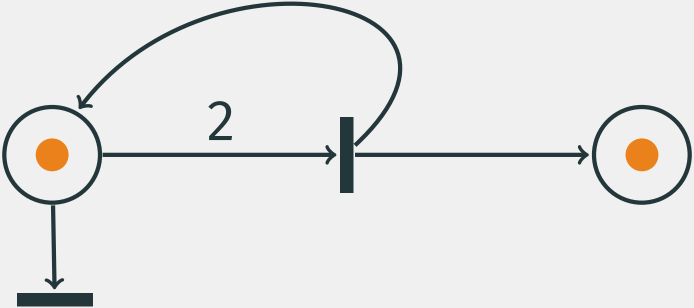
(Discrete) Petri nets



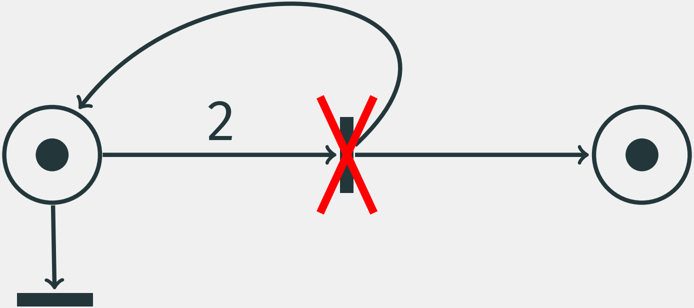
(Discrete) Petri nets



(Discrete) Petri nets

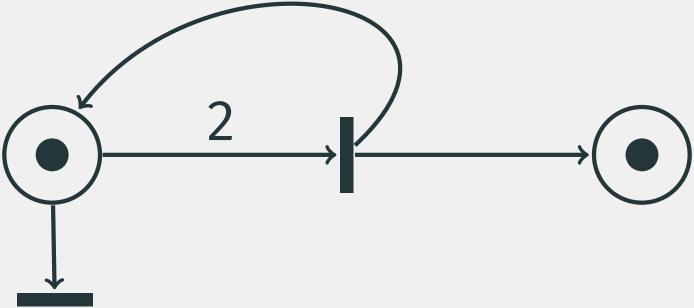


(Discrete) Petri nets



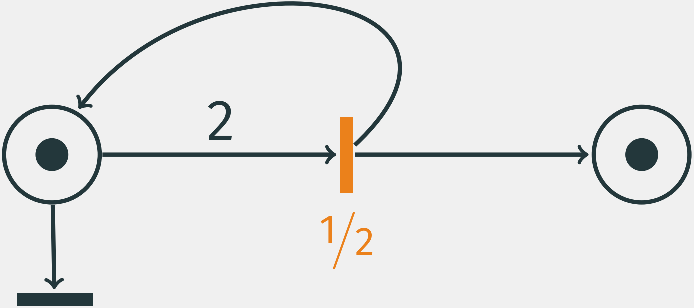
~~(Discrete)~~ Continuous Petri nets

David & Alla '87



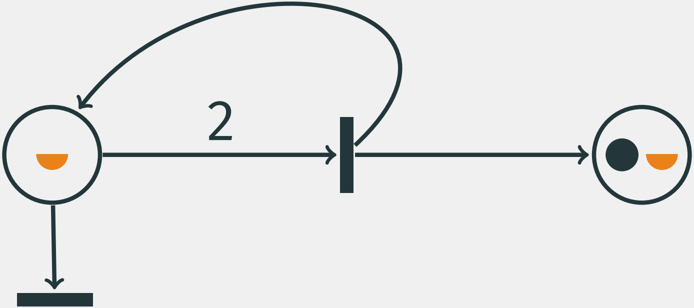
~~(Discrete)~~ Continuous Petri nets

David & Alla '87



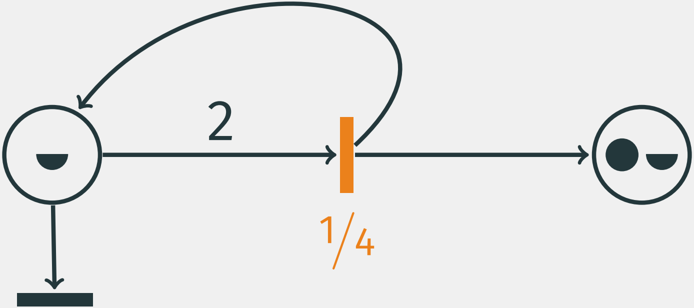
~~(Discrete)~~ Continuous Petri nets

David & Alla '87

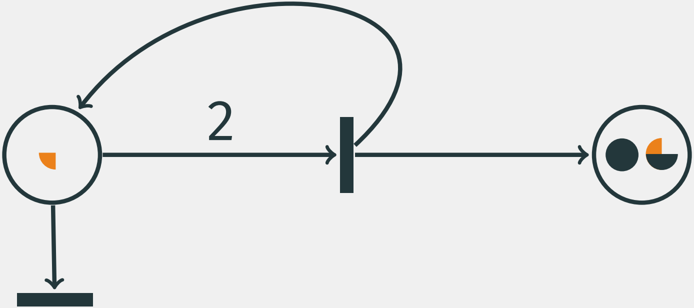


~~(Discrete)~~ Continuous Petri nets

David & Alla '87

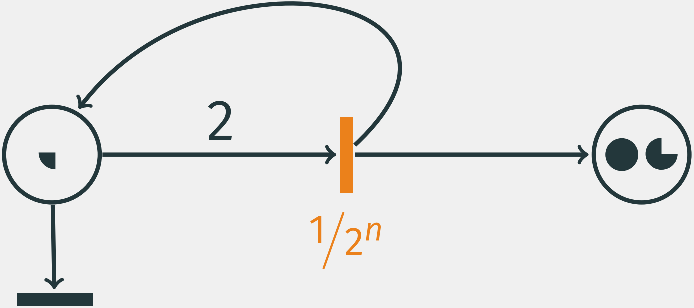


~~(Discrete)~~ Continuous Petri nets



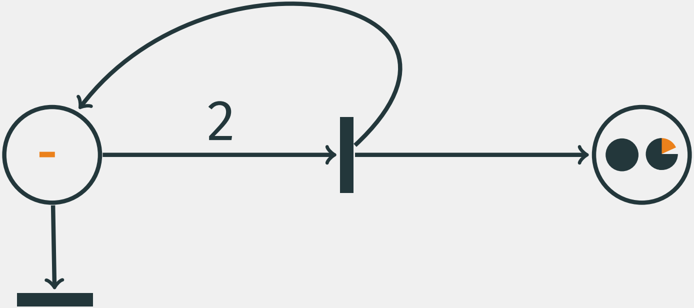
~~(Discrete)~~ Continuous Petri nets

David & Alla '87



~~(Discrete)~~ Continuous Petri nets

David & Alla '87



$$\boldsymbol{m}_{\text{target}} \text{ reachable from } \boldsymbol{m}_{\text{init}}$$

$$\Downarrow$$

$$\boldsymbol{m}_{\text{target}}\ \mathbb{Q}\text{-reachable from } \boldsymbol{m}_{\text{init}}$$

$\boldsymbol{m}_{\text{target}}$ not reachable from $\boldsymbol{m}_{\text{init}}$

Safety

$\Uparrow$

$\boldsymbol{m}_{\text{target}}$ not $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{init}}$

$$\boldsymbol{m}_{\text{target}} \text{ not reachable from } \boldsymbol{m}_{\text{init}}$$

Safety

$$\Uparrow$$

$$\boldsymbol{m}_{\text{target}} \text{ not } \mathbb{Q}\text{-reachable from } \boldsymbol{m}_{\text{init}}$$

Q-reachability $\in$ PTIME!

(Fraca & Haddad '13)

Logical characterization of $\mathbb{Q}$-reachability

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

m_{init} is $\mathbb{Q}$-reachable from m_{target} iff... **Fraca & Haddad '13**

Logical characterization of $\mathbb{Q}$-reachability

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

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}_{\text{target}} = \boldsymbol{m}_{\text{init}} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$

Logical characterization of $\mathbb{Q}$-reachability

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

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v} \in \mathbb{Q}^{T}_{\geq 0}$ such that

- $\boldsymbol{m}_{\text{target}} = \boldsymbol{m}_{\text{init}} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability

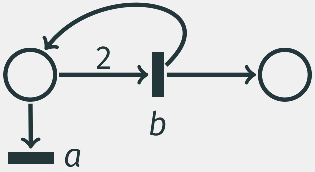
Fix some continuous Petri net $(P, T, \mathbf{Pre}, \mathbf{Post})$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}_{\text{target}} = \boldsymbol{m}_{\text{init}} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability



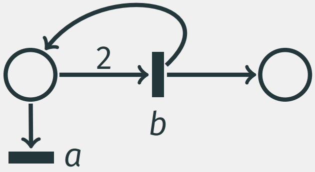
$$\boldsymbol{m}_{\text{init}} = (2, 0)$$
$$\boldsymbol{m}_{\text{target}} = (0, 2)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\boldsymbol{m}_{\text{target}} = \boldsymbol{m}_{\text{init}} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability



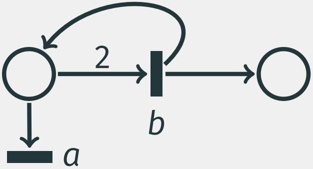
$$\boldsymbol{m}_{\text{init}} = (2, 0)$$
$$\boldsymbol{m}_{\text{target}} = (0, 2)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $$\begin{aligned} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b &= 0 \\ \boldsymbol{v}_b &= 2 \end{aligned}$$
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability



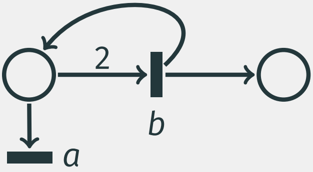
$$\boldsymbol{m}_{\text{init}} = (2, 0)$$
$$\boldsymbol{m}_{\text{target}} = (0, 2)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{aligned} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b &= 0 \\ \boldsymbol{v}_b &= 2 \end{aligned} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2$
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability



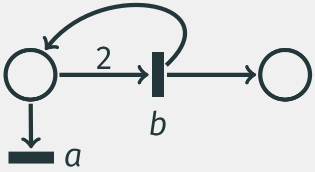
$$\boldsymbol{m}_{\text{init}} = (2, 0)$$
$$\boldsymbol{m}_{\text{target}} = (0, 2)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{aligned} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b &= 0 \\ \boldsymbol{v}_b &= 2 \end{aligned} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2$ ✓
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability



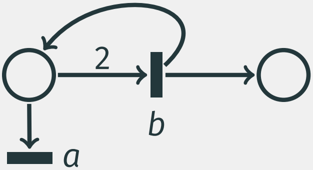
$$\boldsymbol{m}_{\text{init}} = (2, 0)$$
$$\boldsymbol{m}_{\text{target}} = (0, 2)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{aligned} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b &= 0 \\ \boldsymbol{v}_b &= 2 \end{aligned} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2$ ✓
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in \{a, b\} : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability



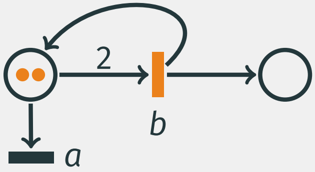
$$\boldsymbol{m}_{\text{init}} = (2, 0)$$
$$\boldsymbol{m}_{\text{target}} = (0, 2)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{array}{r} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b = 0 \\ \boldsymbol{v}_b = 2 \end{array} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2$ ✓
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{b\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{b\}$

Logical characterization of $\mathbb{Q}$-reachability



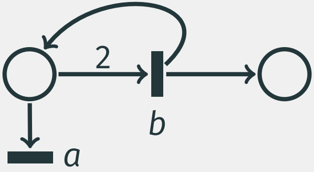
$$\boldsymbol{m}_{\text{init}} = (2, 0)$$
$$\boldsymbol{m}_{\text{target}} = (0, 2)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{aligned} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b &= 0 \\ \boldsymbol{v}_b &= 2 \end{aligned} \implies \boldsymbol{v}_a = 0, \ \boldsymbol{v}_b = 2$ ✓
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{b\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{b\}$

Logical characterization of $\mathbb{Q}$-reachability



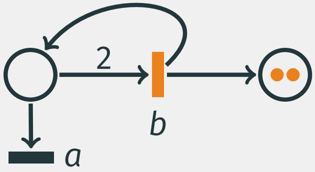
$$\begin{aligned} \boldsymbol{m}_{\text{init}} &= (2, 0) \\ \boldsymbol{m}_{\text{target}} &= (0, 2) \end{aligned}$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{aligned} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b &= 0 \\ \boldsymbol{v}_b &= 2 \end{aligned} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2$ ✓
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{b\}$ ✓
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{b\}$

Logical characterization of $\mathbb{Q}$-reachability

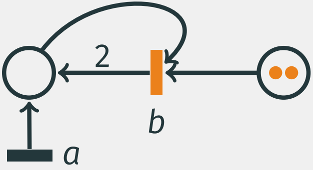


$$\boldsymbol{m}_{\text{init}} = (2, 0)$$
$$\boldsymbol{m}_{\text{target}} = (0, 2)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{aligned} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b &= 0 \\ \boldsymbol{v}_b &= 2 \end{aligned} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2$ ✓
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{b\}$ ✓
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{b\}$



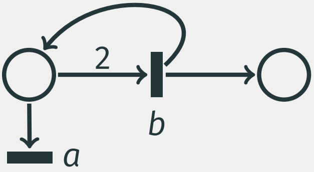
$$\boldsymbol{m}_{\text{init}} = (2, 0)$$
$$\boldsymbol{m}_{\text{target}} = (0, 2)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{aligned} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b &= 0 \\ \boldsymbol{v}_b &= 2 \end{aligned} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2$ ✓
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{b\}$ ✓
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{b\}$

Logical characterization of $\mathbb{Q}$-reachability



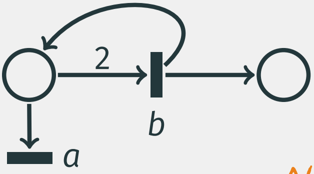
$$\boldsymbol{m}_{\text{init}} = (2, 0)$$
$$\boldsymbol{m}_{\text{target}} = (0, 2)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{aligned} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b &= 0 \\ \boldsymbol{v}_b &= 2 \end{aligned} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2$ ✓
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{b\}$ ✓
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{b\}$ ✗

Logical characterization of $\mathbb{Q}$-reachability



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

$\boxed{\boldsymbol{m}_{\text{target}} = (0, 2)}$

Not Q-reachable from

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v}_a, \boldsymbol{v}_b \in \mathbb{Q}_{\geq 0}$ such that

- $\begin{aligned} 2 - \boldsymbol{v}_a - \boldsymbol{v}_b &= 0 \\ \boldsymbol{v}_b &= 2 \end{aligned} \implies \boldsymbol{v}_a = 0,\ \boldsymbol{v}_b = 2$ ✓
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{b\}$ ✓
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{b\}$ ✗

Logical characterization of $\mathbb{Q}$-reachability

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

$\mathbb{Q}$-reachability definable by linear size formula of

$$\exists\, \mathsf{FO}(\mathbb{Q}, +, <)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... — **Fraca & Haddad '13**

there exists $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}_{\text{target}} = \boldsymbol{m}_{\text{init}} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability

Theorem — B., Finkel, Haase & Haddad '16

$\mathbb{Q}$-reachability definable by linear size formula of

$$\exists\, \mathsf{FO}(\mathbb{N}, +, <)$$

Better approximation!

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... — Fraca & Haddad '13

there exists $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}_{\text{target}} = \boldsymbol{m}_{\text{init}} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability

Theorem — B., Finkel, Haase & Haddad '16

$\mathbb{Q}$-reachability definable by linear size formula of

$$\exists\, \mathrm{FO}(\mathbb{Q}, +, <)$$

Testing validity $\in$ NP

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... — Fraca & Haddad '13

there exists $\boldsymbol{v} \in \mathbb{Q}^{T}_{\geq 0}$ such that

- $\boldsymbol{m}_{\text{target}} = \boldsymbol{m}_{\text{init}} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability

Theorem **B., Finkel, Haase & Haddad '16**

$\mathbb{Q}$-reachability definable by linear size formula of

$$\exists\, \mathsf{FO}(\mathbb{Q}, +, <)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... **Fraca & Haddad '13**

there exists $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}_{\text{target}} = \boldsymbol{m}_{\text{init}} + (\textbf{Post} - \textbf{Pre}) \cdot \boldsymbol{v}$ Straightforward
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Logical characterization of $\mathbb{Q}$-reachability

Theorem — B., Finkel, Haase & Haddad '16

$\mathbb{Q}$-reachability definable by linear size formula of

$$\exists\, \mathsf{FO}(\mathbb{Q}, +, <)$$

$\boldsymbol{m}_{\text{init}}$ is $\mathbb{Q}$-reachable from $\boldsymbol{m}_{\text{target}}$ iff... — Fraca & Haddad '13

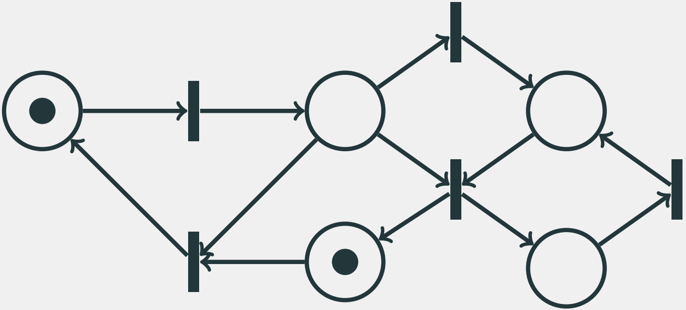
there exists $\boldsymbol{v} \in \mathbb{Q}_{\geq 0}^{T}$ such that

- $\boldsymbol{m}_{\text{target}} = \boldsymbol{m}_{\text{init}} + (\mathbf{Post} - \mathbf{Pre}) \cdot \boldsymbol{v}$

More subtle

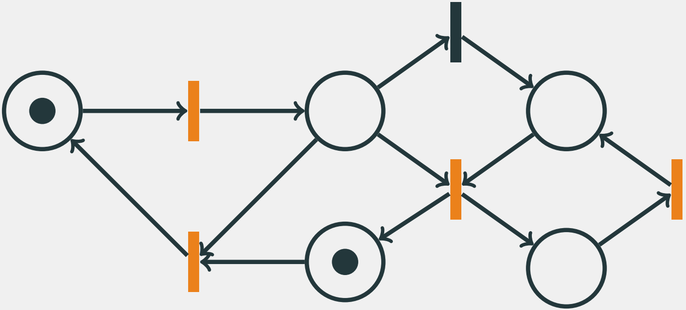
- an execution from $\boldsymbol{m}_{\text{init}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$
- an execution to $\boldsymbol{m}_{\text{target}}$ fires exactly $\{t \in T : \boldsymbol{v}_t > 0\}$

Encoding the firing set conditions



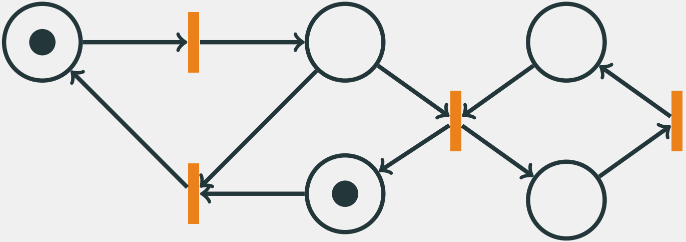
Testing whether some transitions can be fired from initial marking

Encoding the firing set conditions



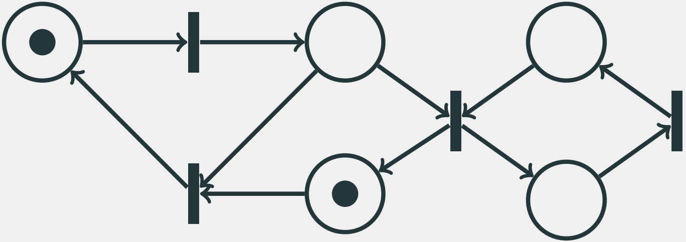
Testing whether some transitions can be fired from initial marking

Encoding the firing set conditions

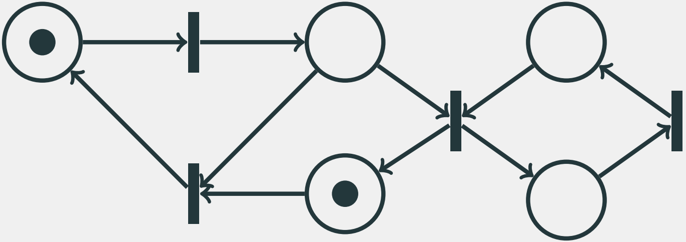


Testing whether some transitions can be fired from initial marking

Encoding the firing set conditions



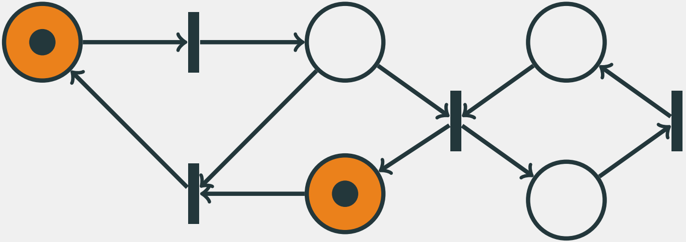
Simulate a "breadth-first" transitions firing



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

Verma, Seidl & Schwentick '05

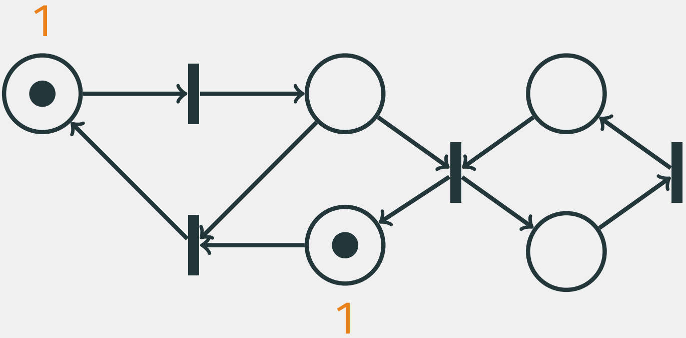
Encoding the firing set conditions



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

Verma, Seidl & Schwentick '05

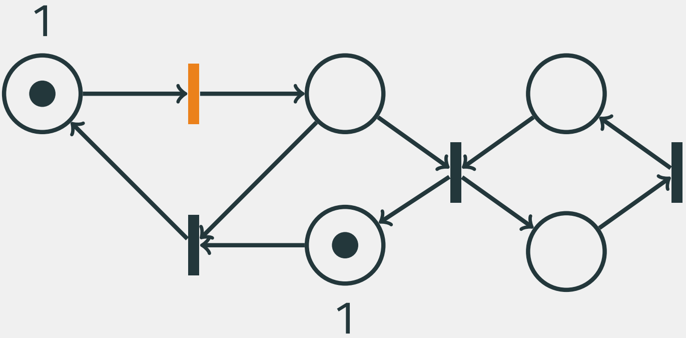
Encoding the firing set conditions



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

Verma, Seidl & Schwentick '05

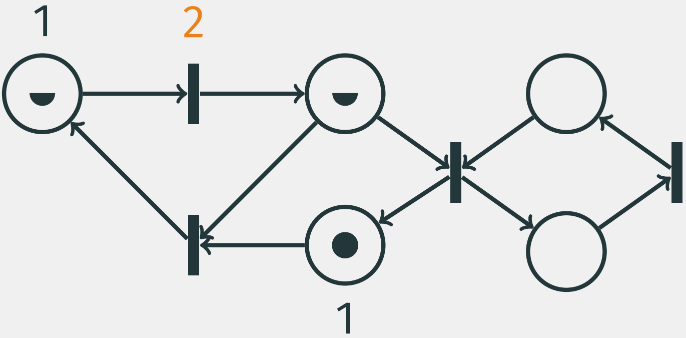
Encoding the firing set conditions



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

Verma, Seidl & Schwentick '05

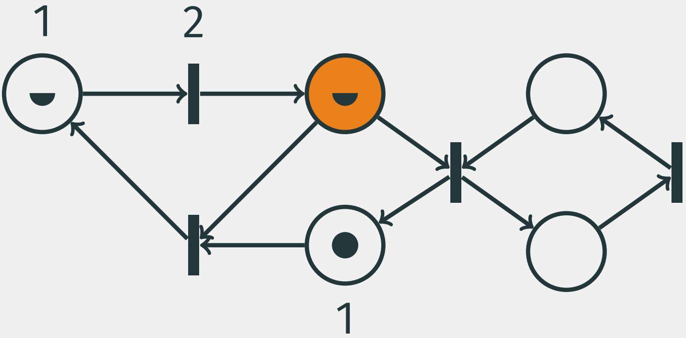
Encoding the firing set conditions



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

Verma, Seidl & Schwentick '05

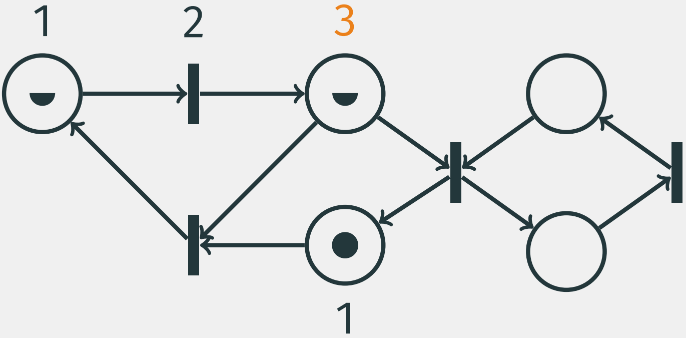
Encoding the firing set conditions



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

Verma, Seidl & Schwentick '05

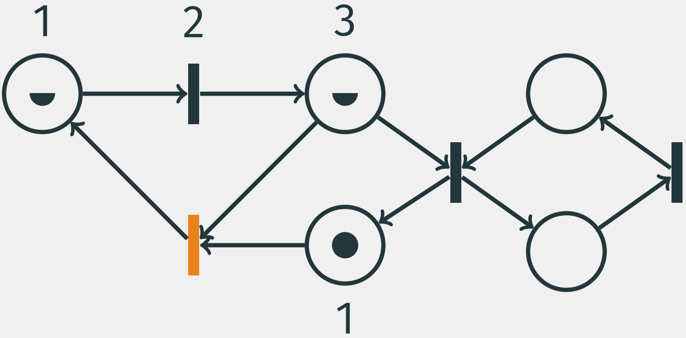
Encoding the firing set conditions



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

Verma, Seidl & Schwentick '05

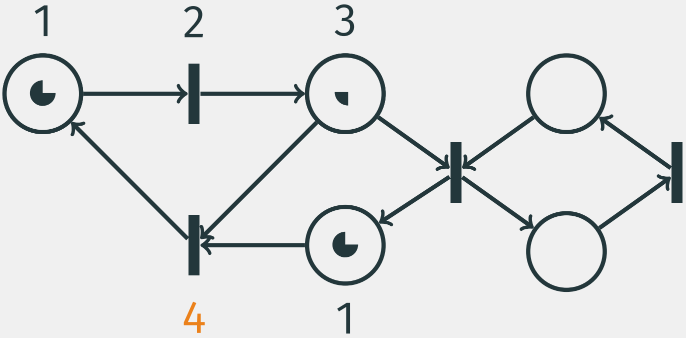
Encoding the firing set conditions



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

Verma, Seidl & Schwentick '05

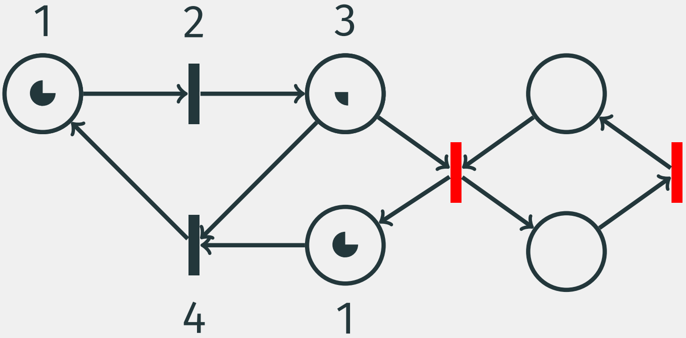
Encoding the firing set conditions



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

Verma, Seidl & Schwentick '05

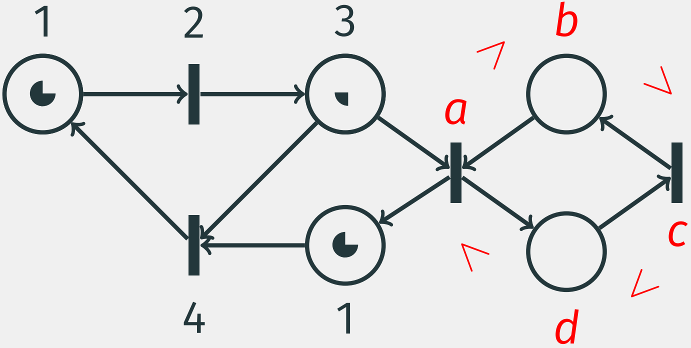
Encoding the firing set conditions



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

Verma, Seidl & Schwentick '05

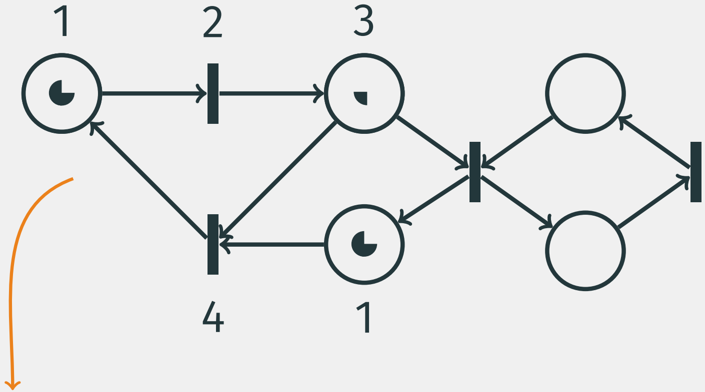
Encoding the firing set conditions



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

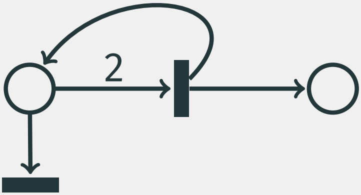
Verma, Seidl & Schwentick '05

Encoding the firing set conditions

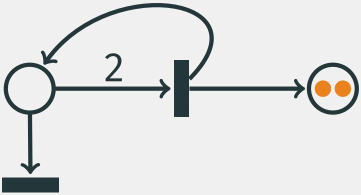


$$\varphi(\boldsymbol{x}) = \exists \boldsymbol{y} : \bigwedge_{p \in P} \boldsymbol{y}(p) > 0 \to \bigwedge_{t \in {}^\bullet p} \boldsymbol{y}(t) < \boldsymbol{y}(p) \cdots$$

From continuous to discrete coverability

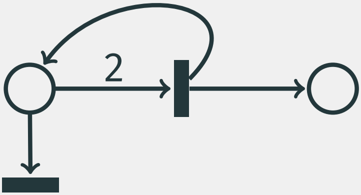


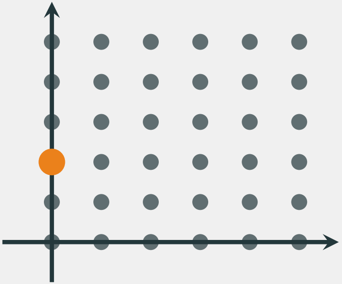
From continuous to discrete coverability



Can $(0, 2)$ be covered from $\boldsymbol{m}_{\text{init}}$?

From continuous to discrete coverability

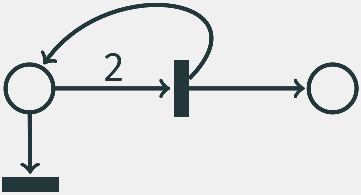


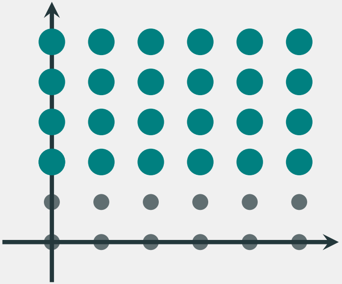


Backward algorithm

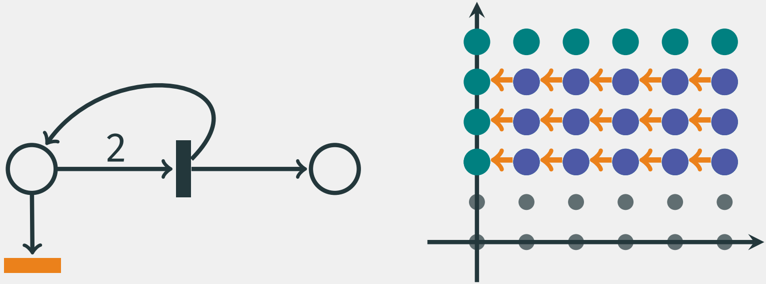
(Arnold & Latteux '78, Abdulla et al. '96)

From continuous to discrete coverability

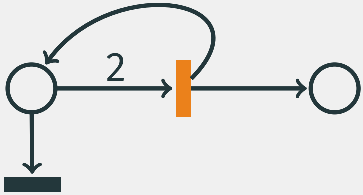


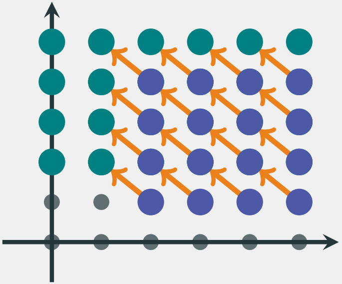


From continuous to discrete coverability

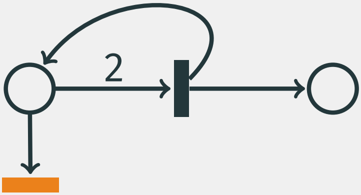


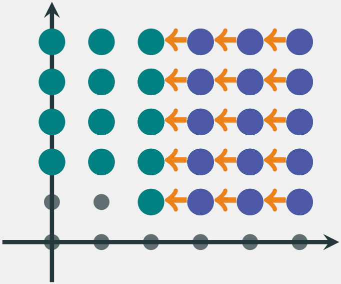
From continuous to discrete coverability



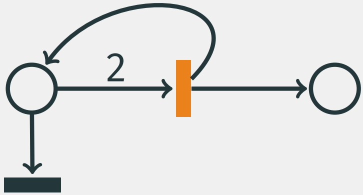


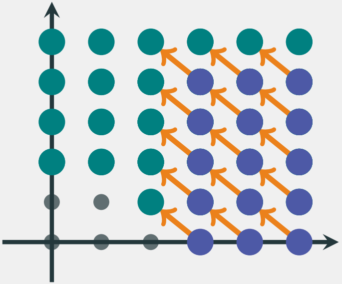
From continuous to discrete coverability



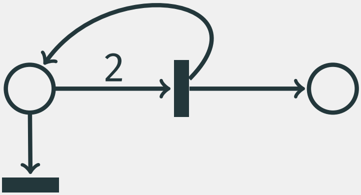


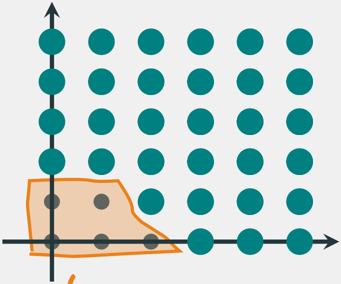
From continuous to discrete coverability



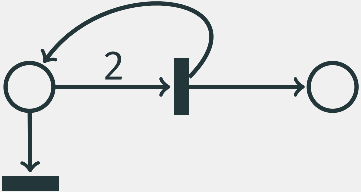


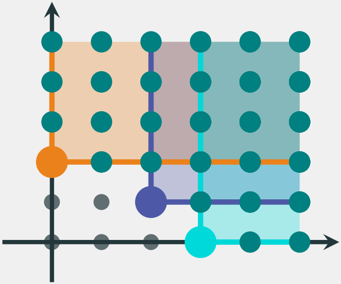
From continuous to discrete coverability





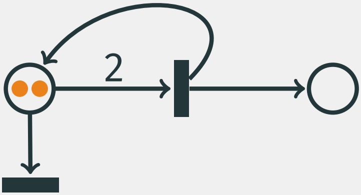
From continuous to discrete coverability

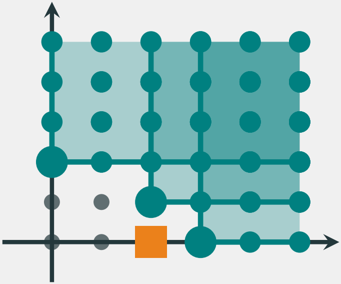




Basis size may become doubly exponential

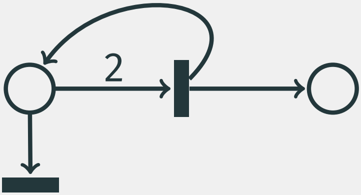
(Bozzelli & Ganty '11)

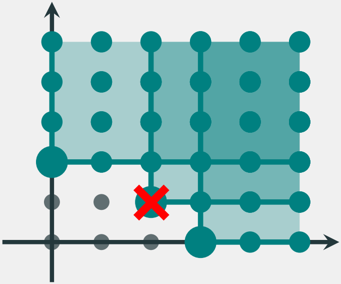




We only care about $\boldsymbol{m}_{\mathrm{init}}$

From continuous to discrete coverability





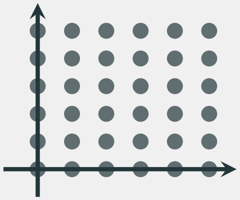
We only care about $\boldsymbol{m}_{\text{init}}$

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

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

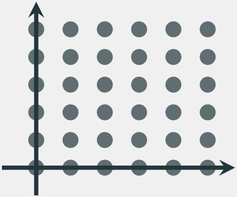
```
if m_target is not ℚ-coverable:
  return false
```

Polynomial time



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

```
if m_target is not Q-coverable:
  return false
X = {m_target}
while (m_init 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 $\mathbb{Q}$-reachability

```
if m_target is not Q-coverable:
  return false
X = {m_target}
while (m_init 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
```

if $\boldsymbol{m}_{\text{target}}$ is not $\mathbb{Q}$-coverable:

  return false

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

while ($\boldsymbol{m}_{\text{init}}$ not covered by X):

  $B =$ markings obtained from X one step backward

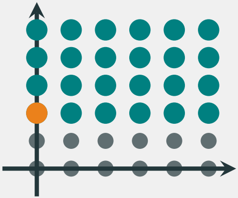
  $B = B \setminus \{\boldsymbol{b} \in B : \neg\varphi(\boldsymbol{b})\}$

  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 $\mathbb{Q}$-reachability

`if` $\boldsymbol{m}_{\text{target}}$ is not $\mathbb{Q}$-coverable:

  `return false`

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

`while` ($\boldsymbol{m}_{\text{init}}$ not covered by X):

  $B =$ markings obtained from X one step backward

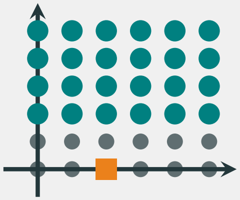
  $B = B \setminus \{\boldsymbol{b} \in B : \neg\varphi(\boldsymbol{b})\}$

  `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 $\mathbb{Q}$-reachability

```
if m_target is not ℚ-coverable:
  return false
```

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

while ($\boldsymbol{m}_{\text{init}}$ not covered by X):

  $B =$ markings obtained from X one step backward

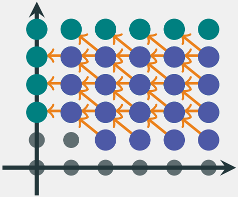
  $B = B \setminus \{\boldsymbol{b} \in B : \neg\varphi(\boldsymbol{b})\}$

  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 $\mathbb{Q}$-reachability

if $\boldsymbol{m}_{\text{target}}$ is not $\mathbb{Q}$-coverable:

  return false

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

while ($\boldsymbol{m}_{\text{init}}$ not covered by X):

  B = markings obtained from X one step backward

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

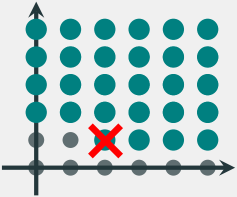
  if $B = \emptyset$: return false

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

  $X = X \cup B$

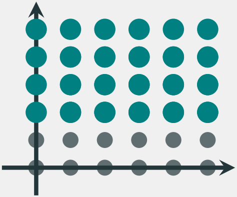
return true

$\mathbb{Q}$-coverability pruning
(better than poly. time)



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

```
if m_target is not Q-coverable:
    return false
X = {m_target}
while (m_init 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 $\mathbb{Q}$-reachability

```
if m_target is not Q-coverable:
    return false
```

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

`while` ($\boldsymbol{m}_{\text{init}}$ not covered by X):

  $B =$ markings obtained from X one step backward

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

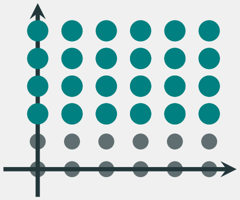
  `if` $B = \emptyset$: `return false`

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

  $X = X \cup B$

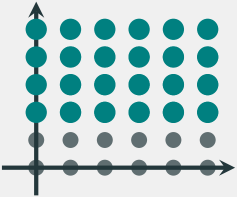
`return true`

SMT solver guidance



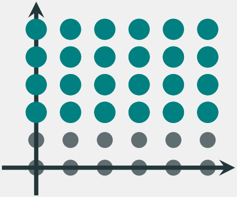
Backward coverability modulo $\mathbb{Q}$-reachability

```
if m_target is not Q-coverable:
   return false
X = {m_target}
while (m_init 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 $\mathbb{Q}$-reachability

```
if m_target is not Q-coverable:
    return false
X = {m_target}
while (m_init 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
```



Our implementation : QCover

B., Finkel, Haase & Haddad '16

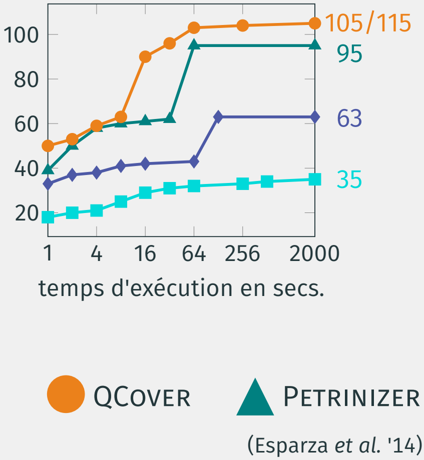
python™ + SMT Solver Z3 (Microsoft Research)

`https://github.com/blondimi/qcover`

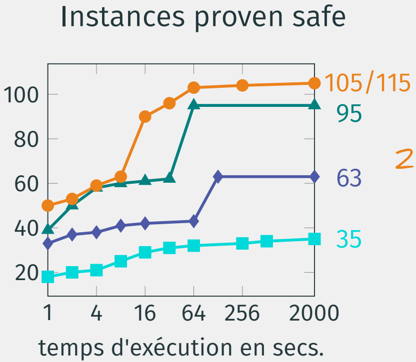
Benchmarked on...

- 176 Petri nets (avg. 1054 places, 8458 trans.)
- multi-threaded C programs with shared-memory
- `Erlang` concurrent programs
- Protocols : mutual exclusion, communication, etc.
- Messages provenance analysis : medical and bug-tracking sys.

Instances proven safe



Our implementation : QCover



Largest nets proven safe:

21143 places
7150 trans.
42 secs.

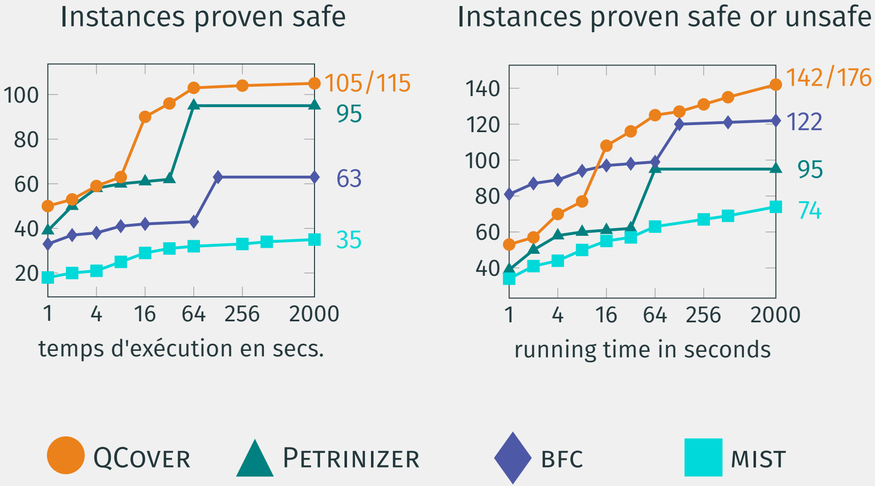
6690 places
11934 trans.
21 secs.

754 places
27370 trans.
3 secs.

QCover | Petrinizer (Esparza *et al.* '14) | BFC (Kaiser *et al.* '12) | MIST (Ganty *et al.* '07)

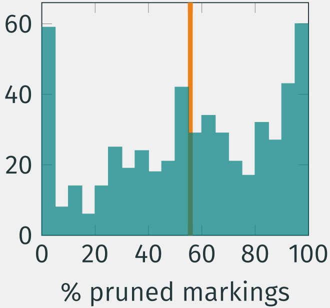
Our implementation : QCover

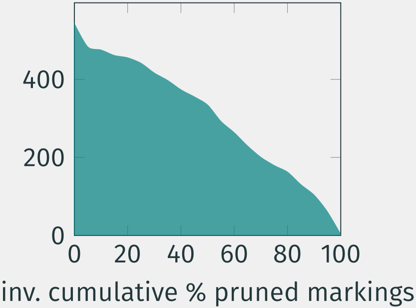
B., Finkel, Haase & Haddad '16



(Esparza *et al.* '14) (Kaiser *et al.* '12) (Ganty *et al.* '07)

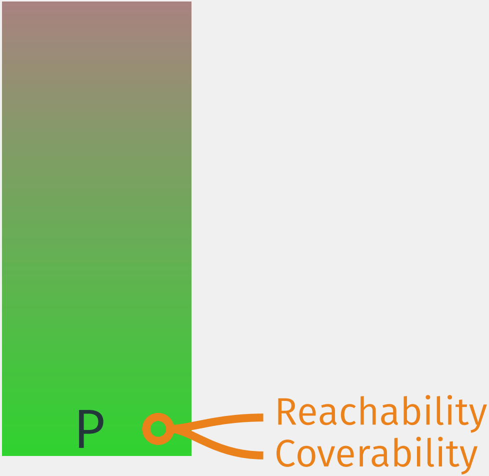
Markings pruning efficiency across all iterations





Another application : from logic to complexity

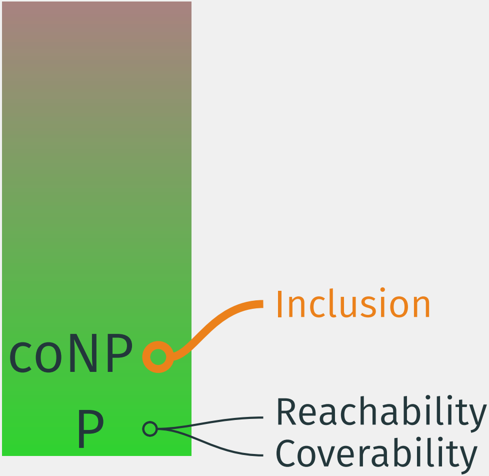
Continuous



(Fraca & Haddad '13)

Another application : from logic to complexity

Continuous



Continuous

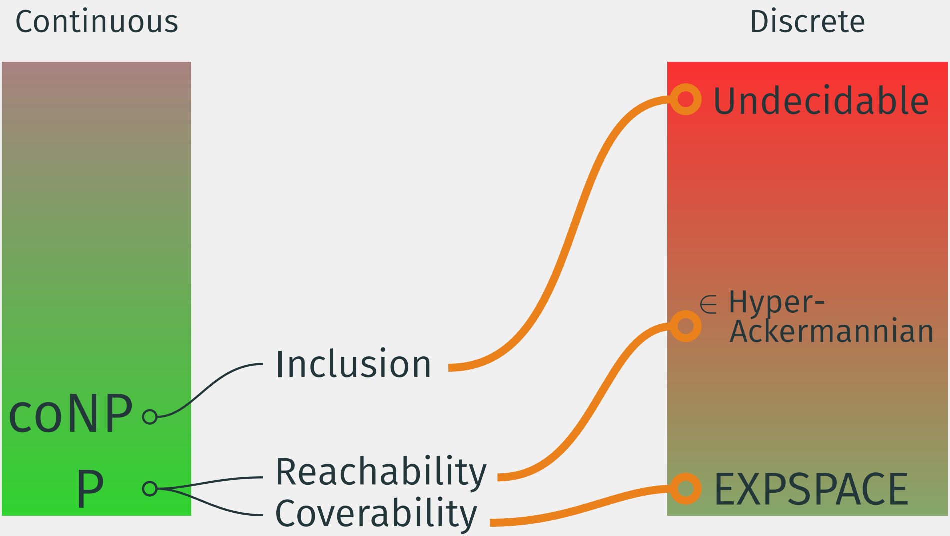
$$\forall \boldsymbol{m}\ \varphi(\boldsymbol{m}_{\text{init}}, \boldsymbol{m}) \rightarrow \varphi'(\boldsymbol{m}'_{\text{init}}, \boldsymbol{m})$$

+ bounds on sys. linear inequalities

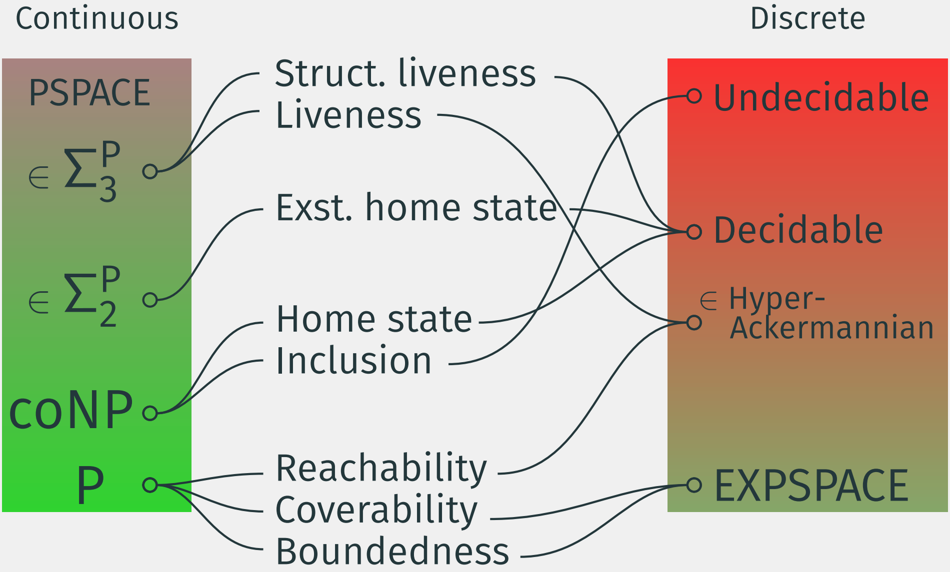
coNP — Inclusion

P — Reachability, Coverability

Another application : from logic to complexity



Another application : from logic to complexity

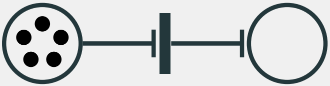


Future work

- Support Petri net extensions : transfers/resets
- Combine our approach with a forward algorithm
- Use upward closed sets data stuctures
  (e.g. sharing trees Delzanno *et al.* '04)
- Continuous vector addition systems with states (VASS)

Future work

- Support Petri net extensions : transfers/resets
- Combine our approach with a forward algorithm
- Use upward closed sets data stuctures
  (e.g. sharing trees Delzanno *et al.* '04)
- Continuous vector addition systems with states (VASS)

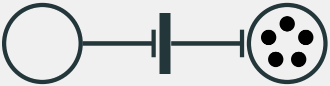


Future work

- Support Petri net extensions : transfers/resets
- Combine our approach with a forward algorithm
- Use upward closed sets data stuctures
  (e.g. sharing trees Delzanno *et al.* '04)
- Continuous vector addition systems with states (VASS)

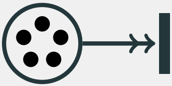
Future work

- Support Petri net extensions : transfers/resets
- Combine our approach with a forward algorithm
- Use upward closed sets data stuctures
  (e.g. sharing trees Delzanno *et al.* '04)
- Continuous vector addition systems with states (VASS)



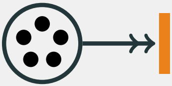
Future work

- Support Petri net extensions : transfers/resets
- Combine our approach with a forward algorithm
- Use upward closed sets data stuctures
  (e.g. sharing trees Delzanno *et al.* '04)
- Continuous vector addition systems with states (VASS)



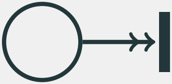
Future work

- Support Petri net extensions : transfers/resets
- Combine our approach with a forward algorithm
- Use upward closed sets data stuctures
  (e.g. sharing trees Delzanno *et al.* '04)
- Continuous vector addition systems with states (VASS)



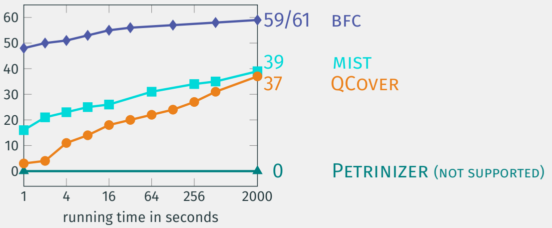
Future work

- Support Petri net extensions : transfers/resets
- Combine our approach with a forward algorithm
- Use upward closed sets data stuctures
  (e.g. sharing trees Delzanno *et al.* '04)
- Continuous vector addition systems with states (VASS)



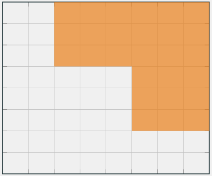
Future work

- Support Petri net extensions : transfers/resets
- Combine our approach with a forward algorithm
- Use upward closed sets data stuctures
  (e.g. sharing trees Delzanno *et al.* '04)
- Continuous vector addition systems with states (VASS)



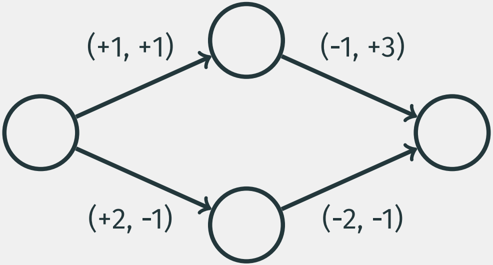
Future work

- Support Petri net extensions : transfers/ resets
- Combine our approach with a forward algorithm
- Use upward closed sets data stuctures
  (e.g. sharing trees Delzanno *et al.* '04)
- Continuous vector addition systems with states (VASS)



Future work

- Support Petri net extensions : transfers/resets
- Combine our approach with a forward algorithm
- Use upward closed sets data stuctures
  (e.g. sharing trees Delzanno *et al.* '04)
- Continuous vector addition systems with states (VASS)



Thank you !

Vielen Dank !