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

Places

Transitions

Pre = (1 2 0 0)

Post = (0 1 0 1)

Marking 1/10
Petri nets

Places

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

Post = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}
Petri nets

Places

Transitions

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

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

Marking 1/10
Petri nets

Places

Transitions

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

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

Marking

2

Marking
Petri nets

Places

Transitions

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

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

Marking = \frac{1}{10}
Petri nets

Places

Transitions

Pre = \( (1, 2, 0, 0) \)

Post = \( (0, 1, 0, 1) \)

Marking: 1/10
Petri nets

Places

Transitions

Pre = \((\begin{array}{cc} 1 & 2 \\ 0 & 0 \end{array})\)

Post = \((\begin{array}{cc} 0 & 1 \\ 0 & 1 \end{array})\)

Marking: 1/10
Petri nets

Places

Transitions

Pre = \((1 \ 2) 0 0\)

Post = \((0 1) 0 1\)

Marking = \(1/10\)
Petri nets

\[ Pre = \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix} \]
Petri nets

Places

Transitions

Pre = \((1\ 2)
\begin{array}{c}0 \\ 0\end{array}\)

Post = \((\begin{array}{c}0 \\ 0 \\ 0 \\ 1\end{array})\)

\(\text{Post} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}\)
Safety verification with Petri nets

Process 1

while true:
  x = true
  while y:
    pass

# critical section
  x = false

Process 2

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

Reachability problem
Hyper-Ackermannian (Leroux & Schmitz '15)
EXPSPACE-complete (Lipton '76, Rackoff '78)

Coverability problem
Processes at both ( )

0

Lamport mutual exclusion algorithm
shared variables: x, y
Safety verification with Petri nets

Process 1

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

Process 2

while True:
  y = True
  if x:
    y = False
  while x:
    pass
goto #

Reachability problem

Hyper-Ackermannian \((\text{Leroux \& Schmitz '15})\)

EXPSPACE-complete \((\text{Lipton '76, Rackoff '78})\)

Coverability problem

Processes at both

\((\) critical sections

Lamport mutual exclusion algorithm

shared variables: \(x, y\)
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

Process 1

\[\text{while true:}
\]
\[\text{x} = \text{true}
\]
\[\text{while } \text{y}: \text{pass}
\]
\[\text{critical section}
\]
\[\text{x} = \text{false}
\]

Process 2

\[\text{while true:}
\]
\[\text{\textbf{\# critical section}}
\]
\[\text{y} = \text{true}
\]
\[\text{if } \text{x} \text{ then:}
\]
\[\text{y} = \text{false}
\]
\[\text{while } \text{x}: \text{pass}
\]
\[\text{goto } \text{\# critical section}
\]
\[\text{y} = \text{false}
\]

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
```
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

<table>
<thead>
<tr>
<th>Process 1</th>
<th></th>
<th>while true:</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = true</td>
<td>⟹</td>
<td>x = true</td>
</tr>
<tr>
<td>while y: pass</td>
<td></td>
<td>pass</td>
</tr>
<tr>
<td># critical section</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x = false</td>
<td>⟹</td>
<td>x = false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process 2</th>
<th></th>
<th>while true:</th>
</tr>
</thead>
<tbody>
<tr>
<td>y = true</td>
<td>★</td>
<td>y = true</td>
</tr>
<tr>
<td>if x then:</td>
<td></td>
<td>y = false</td>
</tr>
<tr>
<td>while x: pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>goto ★</td>
<td></td>
<td></td>
</tr>
<tr>
<td># critical section</td>
<td></td>
<td>y = false</td>
</tr>
</tbody>
</table>
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 y
  # 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

Processes at both critical sections

\[ \text{Processes at both critical sections} \]
Reachability problem

Processes at both critical sections

Lamport mutual exclusion algorithm

shared variables: x, y

Safety verification with Petri nets

while true:

x = true

while true:

y:

pass

# critical section

x = false

while true:

y:

pass

goto #

# critical section

y = false

Hyper-Ackermannian! (Leroux & Schmitz '15)

EXPSPACE-complete! (Lipton '76, Rackoff '78)
Safety verification with Petri nets

Hyper-Ackermannian!  \[(\text{Leroux & Schmitz '15})\]

Reachability problem

Processes at both critical sections

\[\iff \bullet = 1\]
Safety verification with Petri nets

Coverability problem

Processes at both critical sections

Reachability problem

Hyper-Ackermannian \( (\text{Leroux }\& \text{ Schmitz '15}) \)

ExpSPACE-complete \( (\text{Lipton '76, Rackoff '78}) \)

Lamport mutual exclusion algorithm

shared variables: \( x, y \)
Safety verification with Petri nets

EXPSPACE-complete!
(Lipton '76, Rackoff '78)

Coverability problem

Processes at both critical sections

\[ \begin{array}{c}
\text{≥ 1} \\
\text{≥ 0}
\end{array} \]
(Discrete) Petri nets
(Discrete) Petri nets
(Discrete) Petri nets
(Discrete) Petri nets
Petri nets

David & Alla '87

\[
\begin{align*}
2 & \rightarrow 1/2 \\
1 & \rightarrow 2
\end{align*}
\]
Discrete
Petri nets
David & Alla '87
Continuous

[Diagram of a Petri net with places and transitions]
Petri nets

David & Alla '87

Continuous

2

1/4

3/10
\[ m_{\text{target}} \text{ reachable from } m_{\text{init}} \]

\[ \Downarrow \]

\[ m_{\text{target}} \mathcal{Q}\text{-reachable from } m_{\text{init}} \]
\[ m_{\text{target}} \text{ not reachable from } m_{\text{init}} \]

\[ m_{\text{target}} \text{ not } \mathbb{Q}\text{-reachable from } m_{\text{init}} \]
\( m_{\text{target}} \) not reachable from \( m_{\text{init}} \)

\( m_{\text{target}} \) not \( \mathbb{Q} \)-reachable from \( m_{\text{init}} \)

\( \mathbb{Q} \)-reachability \( \in \) PTIME!

(Fraca & Haddad '13)
Logical characterization of $\mathbb{Q}$-reachability

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

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

Fraca & Haddad '13
Fix some continuous Petri net \((P, T, \text{Pre}, \text{Post})\)

\(m_{\text{init}}\) is \(\mathbb{Q}\)-reachable from \(m_{\text{target}}\) iff...

<table>
<thead>
<tr>
<th>Fraca &amp; Haddad '13</th>
</tr>
</thead>
<tbody>
<tr>
<td>there exists (v \in \mathbb{Q}_{\geq 0}^T) such that</td>
</tr>
<tr>
<td>• (m_{\text{target}} = m_{\text{init}} + (\text{Post} - \text{Pre}) \cdot v)</td>
</tr>
</tbody>
</table>
Fix some continuous Petri net \((P, T, \text{Pre}, \text{Post})\)

\[ m_{\text{init}} \text{ is } \mathbb{Q}\text{-reachable from } m_{\text{target}} \text{ iff...} \]

there exists \( v \in \mathbb{Q}^T_{\geq 0} \) such that

- \( m_{\text{target}} = m_{\text{init}} + (\text{Post} - \text{Pre}) \cdot v \)

- an execution from \( m_{\text{init}} \) fires exactly \( \{ t \in T : v_t > 0 \} \)

Fraca & Haddad '13
Logical characterization of $\mathbb{Q}$-reachability

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

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

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

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

\[ \text{Fraca & Haddad '13} \]
Logical characterization of $\mathbb{Q}$-reachability

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

- There exists $v_a, v_b \in \mathbb{Q}_{\geq 0}$ such that
  - $m_{\text{target}} = m_{\text{init}} + (\text{Post} - \text{Pre}) \cdot v$
  - An execution from $m_{\text{init}}$ fires exactly $\{t \in \{a, b\} : v_t > 0\}$
  - An execution to $m_{\text{target}}$ fires exactly $\{t \in \{a, b\} : v_t > 0\}$

$m_{\text{init}} = (2, 0)$
$m_{\text{target}} = (0, 2)$

Fraca & Haddad '13
Logical characterization of $\mathbb{Q}$-reachability

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

there exists $v_a, v_b \in \mathbb{Q}_{\geq 0}$ such that

- $2 - v_a - v_b = 0$
- $v_b = 2$

- an execution from $m_{\text{init}}$ fires exactly $\{ t \in \{a, b\} : v_t > 0 \}$
- an execution to $m_{\text{target}}$ fires exactly $\{ t \in \{a, b\} : v_t > 0 \}$

$m_{\text{init}} = (2, 0)$

$m_{\text{target}} = (0, 2)$

Fracca & Haddad '13
**Logical characterization of $\mathbb{Q}$-reachability**

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

- there exists $v_a, v_b \in \mathbb{Q}_{\geq 0}$ such that
  \[ 2 - v_a - v_b = 0 \quad \implies \quad v_a = 0, \; v_b = 2 \]
  \[ v_b = 2 \]

- an execution from $m_{\text{init}}$ fires exactly $\{t \in \{a, b\} : v_t > 0\}$

- an execution to $m_{\text{target}}$ fires exactly $\{t \in \{a, b\} : v_t > 0\}$

---

$m_{\text{init}} = (2, 0)$

$m_{\text{target}} = (0, 2)$
Logical characterization of $\mathbb{Q}$-reachability

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

there exists $v_a, v_b \in \mathbb{Q}_{\geq 0}$ such that

- $2 - v_a - v_b = 0 \implies v_a = 0, v_b = 2$
- $v_b = 2$

- an execution from $m_{\text{init}}$ fires exactly $\{t \in \{a, b\} : v_t > 0\}$
- an execution to $m_{\text{target}}$ fires exactly $\{t \in \{a, b\} : v_t > 0\}$

$m_{\text{init}} = (2, 0)$
$m_{\text{target}} = (0, 2)$
**Logical characterization of $\mathbb{Q}$-reachability**

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

<table>
<thead>
<tr>
<th>Fraca &amp; Haddad '13</th>
</tr>
</thead>
<tbody>
<tr>
<td>there exists $v_a, v_b \in \mathbb{Q}_{\geq 0}$ such that</td>
</tr>
<tr>
<td>$2 - v_a - v_b = 0 \quad \implies \quad v_a = 0, \quad v_b = 2$</td>
</tr>
<tr>
<td>$v_b = 2$</td>
</tr>
<tr>
<td>• an execution from $m_{\text{init}}$ fires exactly ${t \in {a, b} : v_t &gt; 0}$</td>
</tr>
<tr>
<td>• an execution to $m_{\text{target}}$ fires exactly ${t \in {a, b} : v_t &gt; 0}$</td>
</tr>
</tbody>
</table>

$m_{\text{init}} = (2, 0)$

$m_{\text{target}} = (0, 2)$
Logical characterization of $\mathbb{Q}$-reachability

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

Fracca & Haddad '13

there exists $v_a, v_b \in \mathbb{Q}_{\geq 0}$ such that

1. $2 - v_a - v_b = 0 \implies v_a = 0, v_b = 2$
2. $v_b = 2$

• an execution from $m_{\text{init}}$ fires exactly $\{b\}$

• an execution to $m_{\text{target}}$ fires exactly $\{b\}$

$m_{\text{init}} = (2, 0)$

$m_{\text{target}} = (0, 2)$
Logical characterization of $\mathbb{Q}$-reachability

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

<table>
<thead>
<tr>
<th>Fraca &amp; Haddad '13</th>
</tr>
</thead>
<tbody>
<tr>
<td>there exists $v_a, v_b \in \mathbb{Q}_{\geq 0}$ such that</td>
</tr>
<tr>
<td>$2 - v_a - v_b = 0$ $\implies$ $v_a = 0$, $v_b = 2$ $\checkmark$</td>
</tr>
<tr>
<td>• an execution from $m_{\text{init}}$ fires exactly ${b}$</td>
</tr>
<tr>
<td>• an execution to $m_{\text{target}}$ fires exactly ${b}$</td>
</tr>
</tbody>
</table>
Logical characterization of $\mathbb{Q}$-reachability

$m_{init}$ is $\mathbb{Q}$-reachable from $m_{target}$ iff...

there exists $v_a, v_b \in \mathbb{Q}_{\geq 0}$ such that

- $2 - v_a - v_b = 0 \implies v_a = 0, v_b = 2$
- $v_b = 2$

• an execution from $m_{init}$ fires exactly $\{b\}$

• an execution to $m_{target}$ fires exactly $\{b\}$

$m_{init} = (2, 0)$

$m_{target} = (0, 2)$

Fraca & Haddad '13
Logical characterization of $\mathbb{Q}$-reachability

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

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

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

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

1. $2 - \mathbf{v}_a - \mathbf{v}_b = 0 \implies \mathbf{v}_a = 0, \mathbf{v}_b = 2$

2. $\mathbf{v}_b = 2$

3. an execution from $m_{\text{init}}$ fires exactly $\{b\}$

4. an execution to $m_{\text{target}}$ fires exactly $\{b\}$

Fracca & Haddad '13
Logical characterization of $\mathbb{Q}$-reachability

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

there exists $v_a, v_b \in \mathbb{Q}_{\geq 0}$ such that

1. $2 - v_a - v_b = 0 \implies v_a = 0, v_b = 2 \checkmark$
2. $v_b = 2 \checkmark$

- an execution from $m_{\text{init}}$ fires exactly $\{b\} \checkmark$
- an execution to $m_{\text{target}}$ fires exactly $\{b\}$

$m_{\text{init}} = (2, 0)$

$m_{\text{target}} = (0, 2)$

Fracca & Haddad '13
Logical characterization of $\mathbb{Q}$-reachability

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

- $2 - v_a - v_b = 0$ \implies $v_a = 0$, $v_b = 2$
- $v_b = 2$
- an execution from $m_{\text{init}}$ fires exactly $\{b\}$
- an execution to $m_{\text{target}}$ fires exactly $\{b\}$

$m_{\text{init}} = (2, 0)$

$m_{\text{target}} = (0, 2)$

Not $\mathbb{Q}$-reachable from.

Fraca & Haddad '13

There exists $v_a, v_b \in \mathbb{Q}_{\geq 0}$ such that

$x = \frac{1}{2}$

$y = \frac{1}{4}$

$z = \frac{1}{8}$
**Logical characterization of $\mathbb{Q}$-reachability**

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
<th>Validity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2 - v_a - v_b = 0$</td>
<td>$v_a = 0, v_b = 2$</td>
<td>✓</td>
</tr>
<tr>
<td>$v_b = 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>an execution from $m_{\text{init}}$ fires exactly ${b}$</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>an execution to $m_{\text{target}}$ fires exactly ${b}$</td>
<td></td>
<td>✗</td>
</tr>
</tbody>
</table>

$m_{\text{init}} = (2, 0)$

$m_{\text{target}} = (0, 2)$

Fraca & Haddad '13
Logical characterization of $\mathbb{Q}$-reachability

**Theorem**

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

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

---

**Fraca & Haddad '13**

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

- there exists $v \in \mathbb{Q}^T_{\geq 0}$ such that
  - $m_{\text{target}} = m_{\text{init}} + (\text{Post} - \text{Pre}) \cdot v$
  - an execution from $m_{\text{init}}$ fires exactly $\{ t \in T : v_t > 0 \}$
  - an execution to $m_{\text{target}}$ fires exactly $\{ t \in T : v_t > 0 \}$
## Logical characterization of $\mathbb{Q}$-reachability

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

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

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

---

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

Fraca & Haddad '13

there exists $\mathbf{v} \in \mathbb{Q}^T_{\geq 0}$ such that

- $m_{\text{target}} = m_{\text{init}} + (\text{Post} - \text{Pre}) \cdot \mathbf{v}$
- an execution from $m_{\text{init}}$ fires exactly $\{ t \in T : v_t > 0 \}$
- an execution to $m_{\text{target}}$ fires exactly $\{ t \in T : v_t > 0 \}$
Logical characterization of $\mathbb{Q}$-reachability

**Theorem**

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

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

$m_{init}$ is $\mathbb{Q}$-reachable from $m_{target}$ iff...

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

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

- an execution from $m_{init}$ fires exactly $\{ t \in T : \mathbf{v}_t > 0 \}$

- an execution to $m_{target}$ fires exactly $\{ t \in T : \mathbf{v}_t > 0 \}$

**Fraca & Haddad '13**

**Testing validity $\in \mathsf{NP}$**

B., Finkel, Haase & Haddad '16

4/10
Logical characterization of $\mathbb{Q}$-reachability

**Theorem**

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

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

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

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

- $m_{\text{target}} = m_{\text{init}} + (\text{Post} - \text{Pre}) \cdot v$ \textit{Straightforward}
- an execution from $m_{\text{init}}$ fires exactly $\{t \in T : v_t > 0\}$
- an execution to $m_{\text{target}}$ fires exactly $\{t \in T : v_t > 0\}$
Logical characterization of $\mathbb{Q}$-reachability

<table>
<thead>
<tr>
<th>Theorem</th>
<th>B., Finkel, Haase &amp; Haddad '16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{Q}$-reachability definable by linear size formula of</td>
<td>$\exists \text{FO}(\mathbb{Q}, +, &lt;)$</td>
</tr>
</tbody>
</table>

$m_{\text{init}}$ is $\mathbb{Q}$-reachable from $m_{\text{target}}$ iff...

<table>
<thead>
<tr>
<th>Fraca &amp; Haddad '13</th>
</tr>
</thead>
<tbody>
<tr>
<td>there exists $v \in \mathbb{Q}_{\geq 0}^T$ such that</td>
</tr>
</tbody>
</table>

- $m_{\text{target}} = m_{\text{init}} + (\text{Post} - \text{Pre}) \cdot v$ More subtle
- an execution from $m_{\text{init}}$ fires exactly $\{ t \in T : v_t > 0 \}$
- an execution to $m_{\text{target}}$ fires exactly $\{ t \in T : v_t > 0 \}$

Better approximation! Testing validity $\in \mathbb{NP}$.
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
Simulate a "breadth-first" transitions firing
Simulate a "breadth-first" transitions firing by numbering places/transitions

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

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

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

Verma, Seidl & Schwentick '05
Encoding the firing set conditions

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

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

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

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

Verma, Seidl & Schwentick '05
Encoding the firing set conditions

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

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

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

Verma, Seidl & Schwentick '05
Encoding the firing set conditions

\[ \varphi(x) = \exists y : \bigwedge_{p \in P} y(p) > 0 \rightarrow \bigwedge_{t \in \cdot p} y(t) < y(p) \cdots \]
From continuous to discrete coverability

Cannot cover target marking.

Backward algorithm (Arnold & Latteux '78, Abdulla et al. '96)
Can \((0, 2)\) be covered from \(m_{\text{init}}\)?
From continuous to discrete coverability

Cannot cover target marking.

Backward algorithm (Arnold & Latteux '78, Abdulla et al. '96)
From continuous to discrete coverability

Backward algorithm (Arnold & Latteux '78, Abdulla et al. '96)
From continuous to discrete coverability

2

Backward algorithm (Arnold & Latteux '78, Abdulla et al. '96)
From continuous to discrete coverability

- 2

Backward algorithm (Arnold & Latteux '78, Abdulla et al. '96)
From continuous to discrete coverability

- Cannot cover target marking.
- Backward algorithm (Arnold & Latteux '78, Abdulla et al. '96).
From continuous to discrete coverability

2

Backward algorithm (Arnold & Latteux '78, Abdulla et al. '96)
From continuous to discrete coverability

Cannot cover target marking
From continuous to discrete coverability

Basis size may become doubly exponential

(Bozzelli & Ganty '11)
From continuous to discrete coverability

We only care about $m_{\text{init}}$
From continuous to discrete coverability

We only care about $m_{\text{init}}$

Prune basis with $\mathbb{Q}$-reachability!
if $m_{\text{target}}$ is not $\mathbb{Q}$-coverable:
    return false
if \( m_{\text{target}} \) is not \( \mathbb{Q} \)-coverable:
  return false

\( X = \{ m_{\text{target}} \} \)

while (\( m_{\text{init}} \) not covered by \( X \)):
  \( B = \) markings obtained from \( X \) one step backward
  \( B = B \setminus \{ b \in B : \neg \varphi(b) \} \)
  if \( B = \emptyset \): return false
  \( \varphi(x) = \varphi(x) \land \bigwedge_{\text{pruned}} b \ x \not\subseteq b \)

\( X = X \cup B \)

return true
if $m_{\text{target}}$ is not $\mathbb{Q}$-coverable:
    return false

$X = \{m_{\text{target}}\}$

while ($m_{\text{init}}$ not covered by $X$):
    $B =$ markings obtained from $X$ one step backward
    $B = B \setminus \{b \in B : \neg \varphi(b)\}$
    if $B = \emptyset$: return false
    $\varphi(x) = \varphi(x) \land \bigwedge_{\text{pruned}} b \not\in b$
    $X = X \cup B$
return true
if $m_{\text{target}}$ is not $\mathcal{Q}$-coverable:
    return false

$X = \{m_{\text{target}}\}$

while ($m_{\text{init}}$ not covered by $X$):
    $B =$ markings obtained from $X$ one step backward
    $B = B \setminus \{b \in B : \neg \varphi(b)\}$
    if $B = \emptyset$: return false
    $\varphi(x) = \varphi(x) \land \land_{\text{pruned}} b \not\in b$
    $X = X \cup B$

return true
if $m_{\text{target}}$ is not $\mathbb{Q}$-coverable:

return false

$X = \{m_{\text{target}}\}$

while ($m_{\text{init}}$ not covered by $X$):

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

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

if $B = \emptyset$: return false

$\varphi(x) = \varphi(x) \land \bigwedge_{\text{pruned}} b \; x \not\preceq b$

$X = X \cup B$

return true
if $m_{\text{target}}$ is not $\mathcal{Q}$-coverable:
    return false

$X = \{m_{\text{target}}\}$

while ($m_{\text{init}}$ not covered by $X$):
    $B =$ markings obtained from $X$ one step backward
    $B = B \setminus \{b \in B : \neg \varphi(b)\}$
    if $B = \emptyset$: return false
    $\varphi(x) = \varphi(x) \land \bigwedge_{\text{pruned}} b \ x \not\in b$
    $X = X \cup B$

return true
if \( m_{\text{target}} \) is not \( \mathcal{Q} \)-coverable:

\[ \text{return false} \]

\[ X = \{ m_{\text{target}} \} \]

while \( m_{\text{init}} \) not covered by \( X \):

\[ B = \text{markings obtained from } X \text{ one step backward} \]

\[ B = B \setminus \{ b \in B : \neg \varphi(b) \} \]

if \( B = \emptyset \): return false

\[ \varphi(x) = \varphi(x) \land \bigwedge_{pruned} b \; x \not\preceq b \]

\[ X = X \cup B \]

\[ \text{return true} \]
if $m_{\text{target}}$ is not $\mathcal{Q}$-coverable:
    return false

$X = \{m_{\text{target}}\}$

while ($m_{\text{init}}$ not covered by $X$):
    $B =$ markings obtained from $X$ one step backward
    $B = B \setminus \{b \in B : \neg \varphi(b)\}$
    if $B = \emptyset$: return false
    $\varphi(x) = \varphi(x) \land \bigwedge_{\text{pruned } b} x \not\in b$
    $X = X \cup B$

return true
if \( m_{\text{target}} \) is not \( \mathcal{Q} \)-coverable:

\[
\text{return } \text{false}
\]

\[
X = \{ m_{\text{target}} \}
\]

while (\( m_{\text{init}} \) not covered by \( X \)):

\[
B = \text{markings obtained from } X \text{ one step backward}
\]

\[
B = B \setminus \{ b \in B : \neg \varphi(b) \}
\]

if \( B = \emptyset \): return false

\[
\varphi(x) = \varphi(x) \land \bigwedge_{\text{pruned }} b \not\supset x \not\subset b
\]

\[
X = X \cup B
\]

return true
if $m_{\text{target}}$ is not $\mathbb{Q}$-coverable:
  
  return false

$X = \{m_{\text{target}}\}$

while ($m_{\text{init}}$ not covered by $X$):
  
  $B =$ markings obtained from $X$ one step backward
  
  $B = B \setminus \{b \in B : \neg \varphi(b)\}$

  if $B = \emptyset$: return false

  $\varphi(x) = \varphi(x) \land \bigwedge_{\text{pruned}} b \ x \nleq b$

  $X = X \cup B$

return true
**Our implementation: QCover**

<table>
<thead>
<tr>
<th>Python + SMT Solver Z3 (Microsoft Research)</th>
</tr>
</thead>
</table>

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.
Our implementation: QCover

Markings pruning efficiency across all iterations

Instances proven safe

Largest nets proven safe:
- 21143 places, 7150 trans., 42 secs.
- 6690 places, 11934 trans., 21 secs.
- 754 places, 27370 trans., 3 secs.

temps d'exécution en secs.

QCover
Petrinizer (Esparza et al. '14)
BFC (Kaiser et al. '12)
MIST (Ganty et al. '07)
Our implementation: QCover

Instances proven safe

Largest nets proven safe:
- 21,143 places, 7,150 transitions, 11934 transitions, 21 seconds
- 6690 places, 42 seconds
- 754 places, 27,370 transitions, 3 seconds

Graph showing execution times for different implementations:
- QCover
- Petrinizer
- BFC
- MIST

Implementations:
- QCover (Esparza et al. '14)
- Petrinizer (Kaiser et al. '12)
- BFC (Ganty et al. '07)
Our implementation: QCover

Instances proven safe

- temps d'exécution en secs.
- running time in seconds
- QCover
  - QCover (Esparza et al. '14)
- PetrInizer
  - PetrInizer (Kaiser et al. '12)
- BFC
  - BFC (Ganty et al. '07)

Instances proven safe or unsafe

- 105/115
- 142/176
Markings pruning efficiency across all iterations

- Largest nets proven safe:
  - 21143 places, 7150 transitions, 42 seconds
  - 6690 places, 11934 transitions, 21 seconds
  - 754 places, 27370 transitions, 3 seconds

Our implementation: QCover

B., Finkel, Haase & Haddad '16
Another application: from logic to complexity

Continuous

Reachability
Coverability

(Fraca & Haddad '13)
Another application: from logic to complexity

Continuous

coNP  Inclusion
P     Reachability
Coverability

$\phi(m) \rightarrow \phi'(m')$
Another application: from logic to complexity

- Reachability
- Coverability

\( \forall m \varphi(m_{init}, m) \rightarrow \varphi'(m'_{init}, m) \)
+ bounds on sys. linear inequalities

\( \text{coNP} \)
\( \text{P} \)

Inclusion

Reachability

Coverability
Another application: from logic to complexity

Continuous

coNP
P

Inclusion

Discrete

Undecidable

Hyper-Ackermannian

EXPSPACE

Reachability

Coverability

+ bounds on sys. linear inequalities
Another application: from logic to complexity

Continuous

- PSPACE
- \( \in \Sigma^P_3 \)
- \( \in \Sigma^P_2 \)
- coNP
- P

Discrete

- Undecidable
- Decidable
- \( \in \text{Hyper-Ackermannian} \)
- EXPSPACE

- Struct. liveness
- Liveness
- Exst. home state
- Home state
- Inclusion
- Reachability
- Coverability
- Boundedness
Future work

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