Petri Nets and Model Checking

Natasa Gkolfi

University of Oslo

March 31, 2017
Petri Nets

Petri Nets:

- mathematically founded formalism
- concurrency
- synchronization
- modeling distributed systems
Petri Nets

Petri Nets:

- mathematically founded formalism
- concurrency
- synchronization
- modeling distributed systems

Invented by C.A. Petri
Petri Nets

Petri Nets:
- mathematically founded formalism
- concurrency
- synchronization
- modeling distributed systems

Invented by C.A. Petri

They are consisting of:
- places
- transitions
- arcs
- tokens
- initial marking
Petri Nets - Mutual Exclusion

\[
\begin{align*}
  p_1 & \quad t_1 \quad p_2 \\
  p_3 & \quad t_3 \\
  s & \quad t_2 \quad t_4 \\
\end{align*}
\]
Petri Nets - Mutual Exclusion

\[ p_1 \rightarrow t_1 \rightarrow s \rightarrow t_2 \rightarrow p_2 \]

\[ p_3 \rightarrow t_3 \rightarrow s \rightarrow t_4 \rightarrow p_4 \]
Petri Nets - Mutual Exclusion

Petri Net:
- Places: $p_1$, $p_2$, $p_3$, $p_4$
- Transitions: $t_1$, $t_2$, $t_3$, $t_4$
- Start Place: $p_2$
- Token Distribution: $p_1$: 1, $p_2$: 1, $p_3$: 0, $p_4$: 0

Rules:
1. Transition $t_1$ can fire if $p_1$ and $p_2$ are marked.
2. Transition $t_2$ can fire if $p_2$ and $p_3$ are marked.
3. Transition $t_3$ can fire if $p_3$ and $p_4$ are marked.
4. Transition $t_4$ can fire if $p_4$ and $p_1$ are marked.

The net ensures mutual exclusion, preventing simultaneous access to shared resources.
Petri Nets - Mutual Exclusion

$p_1 \quad p_3$

$t_1 \quad t_3$

$s$

$p_2 \quad p_4$

$t_2 \quad t_4$
Petri Nets - Mutual Exclusion

\begin{figure}
\centering
\includegraphics[width=\textwidth]{petri_net_mutation_exclusion}
\caption{Mutual Exclusion Petri Net}
\end{figure}
Petri Nets - Mutual Exclusion
Petri Nets - Mutual Exclusion

\[
p_1 \rightarrow t_1 \\
p_2 \rightarrow t_2 \\
p_3 \rightarrow t_3 \\
p_4 \rightarrow t_4
\]

\[
t_1 \rightarrow s \\
t_2 \rightarrow s \\
t_3 \rightarrow s \\
t_4 \rightarrow s
\]
Petri Nets - Mutual Exclusion

\[ \begin{align*}
  p_1 & \quad t_1 & \quad p_2 & \quad t_2 \\
  p_3 & \quad t_3 & \quad p_4 & \quad t_4 \\
  s & 
\end{align*} \]
Petri Nets - Mutual Exclusion
Colored Petri nets

High-level Petri nets
The extension of Petri nets (called *place/transition nets*) with abstract data types.

\[
\text{COLORSET}(\text{TYPE}) \quad \longrightarrow \quad \text{Guard} \\
\longrightarrow \quad \text{EXPR} \\
\longrightarrow \quad \text{COLORS}(\text{TYPE VALUES})
\]
Example: Dining Philosophers
Example: Dining Philosophers

```
val n = 5;
color PH = index ph with 1..n;
color CS = index cs with 1..n;
var p: PH;
fun Chopsticks(ph(i)) =
    1`cs(i)+1`cs(if i=n then 1 else i+1);
```
State Space

A directed graph having a node for each reachable marking and an arc for each occurring binding element.
State Space

State Space

A directed graph having a node for each reachable marking and an arc for each occurring binding element.

There is one to one correspondence between the paths in the state space and the occurrence sequences (where all steps consisting of a single binding element)
State Space

State Space

A directed graph having a node for each reachable marking and an arc for each occurring binding element.

There is one to one correspondence between the paths in the state space and the occurrence sequences (where all steps consisting of a single binding element).

The strongly-connected-component graph (SCC graph) is the graph derived from the state space where each node is a SCC of the state space.

SCC graph

- is an acyclic graph
- fewer nodes than the ss mean that there exist infinite occurrence sequences
- more efficient since often much smaller than the ss
Example: Dining Philosophers State Space

- Unused: \(1 \text{`cs}(3)\)
- Think: \(1 \text{`ph}(2) + 1 \text{`ph}(3) + 1 \text{`ph}(5)\)
- Eat: \(1 \text{`ph}(1) + 1 \text{`ph}(4)\)

Petri Nets and Model Checking
Natasa Gkolfi
Behavioral Properties

Boundedness properties
How many and which tokens a place may hold when all reachable markings are considered.

Home Properties
A home marking is a marking that can be reached from any reachable marking

- All the markings in a (single) terminal SCC are home markings
Behavioral Properties

Liveness Properties

A *dead marking* is a marking in which no binding elements are enabled. Similarly *dead transition*. A transition is *live* if, starting from any reachable marking, we can always find an occurrence sequence containing it.
Behavioral Properties

Liveness Properties
A *dead marking* is a marking in which no binding elements are enabled. Similarly *dead transition*.
A transition is *live* if, starting from any reachable marking, we can always find an occurrence sequence containing it.

Fairness Properties
How often transitions occur in infinite occurrence sequences. A transition is *impartial* if it occurs infinitely often in all infinite occurrence sequences.

- *Removal of this transition implies no infinite occurrence sequences!*
Example: Dining Philosophers

<table>
<thead>
<tr>
<th></th>
<th>PH</th>
<th></th>
<th>Nodes</th>
<th>Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>29</td>
<td>112</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>47</td>
<td>208</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>76</td>
<td>378</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>123</td>
<td>680</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1,364</td>
<td>11,310</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
State Space Reduction Methods

▶ Sweep-Line method

A *progress measure* is a function that maps each marking into a *progress value*. For a given marking, the progress value of any successor marking must be greater or equal to its progress value.
State Space Reduction Methods

- **Sweep-Line method**

A *progress measure* is a function that maps each marking into a *progress value*.
For a given marking, the progress value of any successor marking must be greater or equal to its progress value.

- **Symmetry method**

Equivalence classes used for symmetric markings and symmetric binding elements.

- the ss can be significantly reduced
- can check all behavioral properties that are invariant under symmetry
- computing canonical representations of markings and binding elements is computationally expensive
State Space Reduction Methods

- Sweep-Line method

A *progress measure* is a function that maps each marking into a *progress value*. For a given marking, the progress value of any successor marking must be greater or equal to its progress value.

- Symmetry method

Equivalence classes used for symmetric markings and symmetric binding elements.

- the ss can be significantly reduced
- can check all behavioral properties that are invariant under symmetry
- computing canonical representations of markings and binding elements is computationally expensive

- Equivalence method

A generalization of the symmetry method. Here, no requirement that the equivalence relations should be induced by symmetries.
Thank you!