Petri Nets

Jian-Jia Chen
(slides are based on Peter Marwedel)
TU Dortmund,
Informatik 12
2018年 10 月 23 日
## Models of computation considered in this course

<table>
<thead>
<tr>
<th>Communication/local computations</th>
<th>Shared memory</th>
<th>Message passing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undefined components</td>
<td>Plain text, use cases</td>
<td>Synchronous</td>
</tr>
<tr>
<td>Communicating finite state machines</td>
<td>StateCharts</td>
<td>SDL</td>
</tr>
<tr>
<td>Data flow</td>
<td></td>
<td>Kahn networks, SDF</td>
</tr>
<tr>
<td><strong>Petri nets</strong></td>
<td></td>
<td><strong>C/E nets, P/T nets, …</strong></td>
</tr>
<tr>
<td>Discrete event (DE) model</td>
<td>VHDL*, Verilog*, SystemC*, …</td>
<td>Only experimental systems, e.g. distributed DE in Ptolemy</td>
</tr>
<tr>
<td>Von Neumann model</td>
<td>C, C++, Java</td>
<td>C, C++, Java with libraries CSP, ADA</td>
</tr>
</tbody>
</table>

*Classification is based on implementation of VHDL, Verilog, SystemC with central queue*
Introduction

Introduced in 1962 by Carl Adam Petri in his PhD thesis. Focus on modeling causal dependencies; no global synchronization assumed (message passing only).

Key elements:

- **Conditions**
  Either met or not met.

- **Events**
  May take place if certain conditions are met.

- **Flow relation**
  Relates conditions and events.

Conditions, events and the flow relation form a **bipartite graph** (graph with two kinds of nodes).
Interactive Example

http://www.informatik.uni-hamburg.de/TGI/PetriNets/introductions/aalst/
Condition/event nets

Def.: $N=(C,E,F)$ is called a net, if the following holds

1. $C$ and $E$ are disjoint sets

2. $F \subseteq (C \times E) \cup (E \times C)$; is binary relation, ("flow relation")
Pre- and post-sets

Def.: Let $N$ be a net and let $x \in (C \cup E)$.

$\cdot x := \{ y \mid y F x \}$ is called the **pre-set** of $x$,
(or **preconditions** if $x \in E$)

$x^\cdot := \{ y \mid x F y \}$ is called the set of **post-set** of $x$,
(or **postconditions** if $x \in E$)

Example:
Loops and pure nets

Def.: Let \((c,e) \in C \times E\). \((c, e)\) is called a loop if \(cFe \land eFc\).

\[
\text{Def.: Net } N=(C,E,F) \text{ is called pure, if } F \text{ does not contain any loops.}
\]
Simple nets

**Def.** A net is called **simple** if no two nodes $n_1$ and $n_2$ have the same pre-set and post-set.

Example (not simple nets):

![Simple net example 1](image1.png)

![Simple net example 2](image2.png)

**Def.** Simple nets with no isolated elements meeting some additional restrictions are called **condition/event nets** (C/E nets).

![Simple net example 3](image3.png)

![Simple net example 4](image4.png)
Place/transition nets

**Def.:** \((P, T, F, K, W, M_0)\) is called a place/transition net if

1. \(N=(P, T, F)\) is a net with places \(p \in P\) and transitions \(t \in T\)
2. \(K: P \rightarrow (\mathbb{N}_0 \cup \{\omega\}) \setminus \{0\}\) denotes the capacity of places \((\omega\) symbolizes infinite capacity)
3. \(W: F \rightarrow (\mathbb{N}_0 \setminus \{0\})\) denotes the weight of graph edges
4. \(M_0: P \rightarrow \mathbb{N}_0 \cup \{\omega\}\) represents the initial marking of places

defaults:
\[K = \omega\]
\[W = 1\]
Applications

- Modeling of resources;
- modeling of mutual exclusion;
- modeling of synchronization.
Computing changes of markings

“Firing“ transitions $t$ generate new markings on each of the places $p$ according to the following rules:

$$M'(p) = \begin{cases} 
M(p) - W(p,t), & \text{if } p \in t^+ \setminus t^* \\
M(p) + W(t,p), & \text{if } p \in t^* \setminus t^+ \\
M(p) - W(p,t) + W(t,p), & \text{if } p \in t^+ \cap t^+ \\
M(p) & \text{otherwise}
\end{cases}$$
Activated transitions

Transition $t$ is “activated“ if

$$\left( \forall p \in \bullet t : M(p) \geq W(p,t) \right) \land \left( \forall p \in t^\bullet : M(p) + W(t,p) \leq K(p) \right)$$

Activated transitions can “take place“ or “fire“, but don’t have to.
We never talk about “time“ in the context of Petri nets. The order in which activated transitions fire, is not fixed (it is non-determinate).
Shorthand for changes of markings

Slide 12: \[ M'(p) = \begin{cases} 
M(p) - W(p,t), & \text{if } p \in \cdot t \setminus \cdot t^* \\
M(p) + W(t,p), & \text{if } p \in t^* \setminus \cdot t \\
M(p) - W(p,t) + W(t,p), & \text{if } p \in \cdot t \cap t^* \\
M(p) & \text{otherwise}
\end{cases} \]

Let \[ t(p) = \begin{cases} 
-W(p,t) & \text{if } p \in \cdot t \setminus \cdot t \\
+ W(t,p) & \text{if } p \in t^* \setminus \cdot t \\
-W(p,t) + W(t,p) & \text{if } p \in \cdot t \cap t^* \\
0 & \text{otherwise}
\end{cases} \]

\[ \Rightarrow \quad \forall p \in P: \ M'(p) = M(p) + t(p) \]

\[ \Rightarrow \quad M' = M + t \quad +: \text{ vector add} \]
**Matrix $N$ describing all changes of markings**

\[
t(p) = \begin{cases} 
-W(p, t) & \text{if } p \in t \setminus t' \\
+W(t, p) & \text{if } p \in t' \setminus t \\
-W(p, t) + W(t, p) & \text{if } p \in t' \cap t' \\
0 & \text{otherwise}
\end{cases}
\]

Def.: Matrix $N$ of net $N$ is a mapping

$$N: P \times T \to \mathbb{Z} \text{ (integers)}$$

such that $\forall \ t \in T : N(p, t) = t(p)$

Component in column $t$ and row $p$ indicates the change of the marking of place $p$ if transition $t$ takes place.

For pure nets, $(N, M_0)$ is a complete representation of a net.
Place - invariants

Standardized technique for proving properties of system models

For any transition $t_j \in T$ we are looking for sets $R \subseteq P$ of places for which the accumulated marking is constant:

$$\sum_{p \in R} t_j(p) = 0$$

Example:
Predicate/transition nets

Goal: compact representation of complex systems.

Key changes:

- Tokens are becoming individuals;
- Transitions enabled if functions at incoming edges true;
- Individuals generated by firing transitions defined through functions

Changes can be explained by folding and unfolding C/E nets, semantics can be defined by C/E nets.
Example: Dining philosophers problem

\(n > 1\) philosophers sitting at a round table;
\(n\) forks,
\(n\) plates with spaghetti; philosophers either thinking or eating spaghetti (using left and right fork).

How to model conflict for forks?
How to guarantee avoiding starvation?

2 forks needed!
Condition/event net model of the dining philosophers problem

Let $x \in \{1..3\}$
- $t_x$: $x$ is thinking
- $e_x$: $x$ is eating
- $f_x$: fork $x$ is available

Model quite clumsy.
Difficult to extend to more philosophers.
Predicate/transient model of the dining philosophers problem (1)

Let \( x \) be one of the philosophers, let \( l(x) \) be the left fork of \( x \): \( f_x \), let \( r(x) \) be the right fork of \( x \): \( f_{(x \mod 3)+1} \).

\[
\begin{align*}
l(2) &= f_2, & r(2) &= f_3, & l(3) &= f_3, & r(3) &= f_1
\end{align*}
\]

Tokens: individuals.

Semantics can be defined by replacing net by equivalent condition/event net.
Model can be extended to arbitrary numbers of people.
Evaluation

Pros:
- Appropriate for distributed applications,
- Well-known theory for formally proving properties,

Cons (for the nets presented):
- problems with modeling timing,
- no programming elements,
- no hierarchy.

Extensions:
- Enormous amounts of efforts on removing limitations.
Summary

Petri nets: focus on causal dependencies

- Condition/event nets
  - Single token per place

- Place/transition nets
  - Multiple tokens per place

- Predicate/transition nets
  - Tokens become individuals
  - Dining philosophers used as an example