Applications of Petri Nets

Presenter: Chung-Wei Lin

2010.10.28
Outline

- Revisiting Petri Nets
- Application 1: Software Syntheses
  - Theory and Algorithm
- Application 2: Biological Networks
  - Comprehensive Introduction
- Application 3: Supply Chains
  - Example and Experiment
- Summary
Definition of a Petri Net

• A 3-tuple (P,T,F)
  - P: set of **places**
  - T: set of **transitions**
  - F: (P × T) U (T × P) → N, weighted flow relation

  ![Diagram of Petri Net]

• How does it work?
  - Each place holds some (≥ 0) **tokens**
  - A transition is **enabled** if its input places contain at least the required # of tokens
  - The **firing** of an enabled transition results in
    - Consumption of the tokens of its input places
    - Production of the tokens of its output places
More about Petri Nets

- **Marking**
  - A vector representing the number of tokens in all places

- **Properties of Petri Nets**
  - Reachability (of a marking from another marking)
  - Boundedness
    - The numbers of tokens in all places are bounded
  - Conservation
    - The total number of tokens is constant
  - Deadlock-freedom
    - Always at least one transition can fire
  - Liveness
    - From any marking, any transition can fire sometime
  - Schedulability
    - The first paper will discuss this
**T-Invariant and Finite Complete Cycle**

- **T-invariant** is a vector s.t.
  - The i-th component is the number of firing times of transition $t_i$
  - The marking is unchanged if firing them so many times
  - However, it does not guarantee that a transition can be fired
    - Deadlock

- **Finite complete cycle** is a sequence of transitions s.t.
  - The marking is unchanged if firing the sequence

![Diagram of a Petri net with transitions $t_1$, $t_2$, $t_3$, and places $p_1$, $p_2$, and $t_3$.]

One T-invariant: $(4,2,1)$

Some finite complete schedules:

- $\langle t_1, t_1, t_1, t_1, t_2, t_1, t_3 \rangle$
- $\langle t_1, t_1, t_1, t_2, t_1, t_2, t_3 \rangle$
Outline

• Revisiting Petri Nets
• Application 1: Software Syntheses
  – Synthesis of Embedded Software Using Free Choice Petri Nets
• Application 2: Biological Networks
• Application 3: Supply Chains
• Summary
Static, Quasi-Static, and Dynamic Scheduling

- **Scheduling problem**
  - Mapping a functional implementation to real resources
  - Satisfying real-time constraints
  - Using resources as efficiently as possible

- **Static scheduling**
  - Specifications contain only data computations
  - The schedule can be completely computed at compile time

- **Quasi-static scheduling**
  - Specifications contain data-dependent controls, like if-then-else or while-do loops
  - The schedule leaves data-dependent decisions at run-time

- **Dynamic scheduling**
  - Specifications contain real-time controls
Free Choice Petri Net

• Free Choice Petri Net (FCPN) is a Petri net such that every arc from a place is
  – A unique outgoing arc, or
  – A unique incoming arc to a transition

• Two transitions are in equal conflict relation (ECR) if their presets are non-empty and equal

\[ \text{FCPN} \quad \text{Not FCPN} \]

\[ \text{ECR} \quad \text{Not ECR} \]
Let

- \( \Sigma = \{\sigma_1, \sigma_2, \ldots \} \) be a finite set
  - \( \sigma_i = <\sigma_i^1, \sigma_i^2, \ldots> \) is a finite complete cycle containing all source transitions

\( \Sigma \) is a **valid schedule (set)** if

- For all \((\sigma_i^j, t_k)\) s.t.
  - \( \sigma_i^j \neq \sigma_i^h \) for all \( h < j \)
  - \( \sigma_i^j \neq t_k \) and they are in equal conflict relation
- Exist \( \sigma_l \) s.t.
  - \( \sigma_l^m = \sigma_l^m \) for all \( m \leq j \)
  - \( \sigma_l^m = t_k \) if \( m = j \)

In words, a valid schedule is

- A set of finite complete cycles for every possible outcome of a choice

\( \sigma_1 \) in \( \Sigma \) iff \( \sigma_2 \) in \( \Sigma \)
Quasi-Statically Schedulable

• Given
  – A FCPN $N$
  – An initial marking $\mu_0$

• $(N, \mu_0)$ is quasi-statically schedulable if
  – There exists a valid schedule

\[ \Sigma = \{ <t_1, t_2, t_4>, <t_1, t_3, t_5> \} \]

Schedulable

\[ \Sigma = \{ <t_1, t_1, t_2, t_3, t_4> \} \] ?

$\Sigma = \{ <t_1, t_1, t_2, t_3, t_4>, <t_1, t_1, t_3, t_2, t_4> \} \] ?

Non-schedulable
How to Find a Valid Schedule? Step 1

- **T-allocation** is a function that chooses exactly one transition for every place

- **T-reduction** associated with a T-allocation is a set of subnets generated from the image of the T-allocation

\[
A_1 = \{t_1, t_2, t_4, t_5, t_6, t_7, t_8, t_9\} \\
A_2 = \{t_1, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}
\]

- **Step 1**: decompose a net into conflict-free components
  - Compute all T-reductions of the net
    - Reduction algorithm
How to Find a Valid Schedule? Step 2

• A T-reduction is **schedulable** if
  - It has a finite complete cycle that
    - Contains at least one occurrence of every source transition of the net
  - (Definition 3.5)

• Step 2: check if every conflict-free component is statically schedulable
  - Apply the standard techniques for synchronous dataflow networks
    - Solve T-invariant equation
    - Check deadlock by simulation
How to Find a Valid Schedule? Step 3

- Given a FCPN, there exists a valid schedule if and only if every T-reduction is schedulable
  - Note that: a valid schedule $\rightarrow$ quasi-schedulable

- Step 3: derive a valid schedule, if there exists one
  - Compute the union of the finite complete cycles of all T-reduction

<table>
<thead>
<tr>
<th>T-invariants (why needs two?)</th>
<th>finite complete cycle</th>
<th>valid schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1,0,1,0,1,0,0,0)</td>
<td>$&lt;t_1,t_2,t_4,t_6,t_8,t_9,t_6&gt;$</td>
<td>${&lt;t_1,t_2,t_4,t_6,t_8,t_9,t_6&gt;,&lt;t_1,t_3,t_5,t_7,t_8,t_9,t_6&gt;}$</td>
</tr>
<tr>
<td>(0,0,0,0,0,1,0,1,1)</td>
<td>$&lt;t_1,t_3,t_5,t_7,t_8,t_9,t_6&gt;$</td>
<td></td>
</tr>
<tr>
<td>(1,0,1,0,1,0,1,0,0,0)</td>
<td>(0,0,0,0,0,1,0,1,1)</td>
<td></td>
</tr>
</tbody>
</table>
• Derive an implementation directly from a valid schedule

\[ \text{valid schedule} \Rightarrow \{ <t_1,t_2,t_1,t_2,t_4>,<t_1,t_3,t_5,t_5> \} \]

\[
\text{while ( true ) } \{
    t_1;
    \text{if ( } p_1 \text{ ) } \{
        t_2;
        \text{count ( } p_2 \text{ ) ++;}
        \text{if ( count ( } p_2 \text{ ) == 2 ) } \{
            t_4;
            \text{count ( } p_2 \text{ ) --;}
        \}\
    \text{else } \{
        t_3;
        \text{count ( } p_3 \text{ ) += 2;}
        \text{while ( count ( } p_3 \text{ ) >= 1 ) } \{
            t_5;
            \text{count ( } p_3 \text{ ) --;}
        \}
    \}
\} \]
Outline

• Revisiting Petri Nets
• Application 1: Software Syntheses
• Application 2: Biological Networks
  – Petri Net Modeling of Biological Networks
• Application 3: Supply Chains
• Summary
Basic Modeling of Biological Reactions (1/2)

- **Synthesis**
  - Diagrams showing the synthesis of AB from A and B.

- **Decomposition**
  - Diagrams showing the decomposition of AB into A and B.

- **Reversible reaction with stoichiometry**
  - Diagrams showing the reversible reaction of AB to A and B, with a 2:2 stoichiometry.
Basic Modeling of Biological Reactions (2/2)

- Catalyzed reaction

\[ \text{A} + \text{B} \rightarrow \text{AB} \]

- Inhibited reaction

\[ \text{A} + \text{B} \rightarrow \text{AB} \]
Extensions of Petri Nets

• Coloured Petri Net (CPN)
  – Assign data values to the token
  – Define constraints on the token values

• Stochastic Petri Net (SPN)
  – Transitions have exponentially distributed time delays

• Hybrid Petri Net (HPN)
  – Discrete and continuous places
    ■ Marked tokens and concentration levels
  – Discrete and continuous transitions
    ■ Determined and distributed delays

• Functional Petri Net (FPN)
  – Flow relations depend on the marking

• Hybrid Functional Petri Net (HFPN)
More Complicated Modeling

• Biochemical networks
  – Enzymatic reaction chain: CPN
  – Intrinsic noise due to low concentrations: SPN
  – More general pathway: FPN, HFPN

• Genetic networks
  – Response to genes rather than consumption and production
  – Switch Control: logical approach, CPN
  – Concentration dynamics: HPN, HFPN

• Signaling networks
  – Response to signal rather than consumption and production
  – Transition delay: timed PN, timed CPN, SPN

• Discussion
  – Tradeoff between expressiveness and analyzability
  – Spatial properties, hierarchical modeling, other modeling formalisms
Outline

• Revisiting Petri Nets
• Application 1: Software Syntheses
• Application 2: Biological Networks
• Application 3: Supply Chains
  – Performance Analysis and Design of Supply Chains: A Petri Net Approach
• Summary
Supply Chain Networks (SCN)

raw material vendors

inbound logistics

intermediate inventory

OEM

distribution centers

finished goods inventory

outbound logistics

retailers

customers
Configurations & Operational Models of SCN

- **Configurations**
  - Serial structure
  - Divergent structure: petroleum industry
  - Convergent structure: automobiles and air crafts
  - Network structure: computer industry

- **Operational models**
  - Make-to-stock (MTS)
    - Orders are satisfied from stocks of inventory of *finished goods which are kept at retail points*
  - Make-to-order (MTO)
    - *A confirmed order triggers the flow* of the supply chain
  - Assemble-to-order (ATO)
    - Before *decoupling point*, intermediate goods are made-to-stock
    - After decoupling point, goods are made-to-order
  - Tradeoff between holding cost and delayed delivery’s cost
Performance Analysis – Modeling

- Generalized Stochastic Petri Net (GSPN)
  - Random order request
  - Random logistics/interface time

- MTS, MTO, and ATO may have different structures & initial markings
Performance Analysis – Setting

• Cost function
  – Holding cost for inventories: $H_I$
  – Cost of delayed delivery: $H_D$
  – Vary the ratio of $H_D$ and $H_I$ from 1.5 to 40.0

• Apply Stochastic Petri Net Package (SPNP)
• Compare between make-to-stock (MTS) and assemble-to-order (ATO)
### Performance Analysis – Experiment 1

- Change arrival rate of end products on \((W_2)\)

<table>
<thead>
<tr>
<th>Arrival Rate ((W_2))</th>
<th>Total Cost (H_D / H_I)</th>
<th>MTS</th>
<th>ATO</th>
<th>Total Cost (H_D / H_I)</th>
<th>MTS</th>
<th>ATO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(H_D / H_I = 1.5) ((\text{Expensive Holding}))</td>
<td></td>
<td></td>
<td>(H_D / H_I = 40) ((\text{Expensive Delay}))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>22.421</td>
<td>19.815</td>
<td></td>
<td>26.001</td>
<td>257.437</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>21.237</td>
<td>18.610</td>
<td></td>
<td>25.818</td>
<td>237.559</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>20.012</td>
<td>17.714</td>
<td></td>
<td>25.961</td>
<td>224.228</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>18.774</td>
<td>17.016</td>
<td></td>
<td>26.339</td>
<td>214.675</td>
<td></td>
</tr>
</tbody>
</table>

- MTS > ATO reasonable
- U-shape why?
- MTS < ATO reasonable
Performance Analysis – Experiment 2

- Change targeted finished goods inventory (on M)

<table>
<thead>
<tr>
<th>FGI (M)</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MTS</td>
</tr>
<tr>
<td></td>
<td>H_D / H_I = 1.5</td>
</tr>
<tr>
<td>MTS</td>
<td>ATO</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
</tr>
</tbody>
</table>

- Holding costs play more important roles
- Immune but useless
Performance Analysis – Experiment 3

- Change interface times (from S2 to M)

<table>
<thead>
<tr>
<th>Interface Rate (S2 → M)</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MTS</th>
<th>ATO</th>
<th>MTS</th>
<th>ATO</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.566</td>
<td>15.542</td>
<td>24.934</td>
<td>197.185</td>
</tr>
<tr>
<td>22.651</td>
<td>15.640</td>
<td>24.981</td>
<td>197.360</td>
</tr>
<tr>
<td>22.709</td>
<td>15.705</td>
<td>25.038</td>
<td>197.502</td>
</tr>
<tr>
<td>22.780</td>
<td>15.785</td>
<td>25.109</td>
<td>197.659</td>
</tr>
</tbody>
</table>

holding costs increase because interface S1 → M is not fast enough to work with interface S2 → M
Decoupling Point Location Problem – Modeling

- Integrated GSPN-queuing model
  - Amendable for integrated queuing network GSPN analysis and deriving aggregated facility by solving the original product from queuing network (PFQN)
Decoupling Point Location Problem – Setting

- **Cost function**
  - Holding cost (proportional to $H_1$)
    - $H_1$: holding cost for the first stage supplier
    - Increase as moving from the first stage supplier to the distribution center ($H_1, 1.2H_1, 1.2^2H_1, 1.2^3H_1, ...$)
  - Lead time cost (proportional to $H_2$)
    - $H_2$: average lead time cost per unit good per hour

- Consider 5 stage supply chain with the last stage being the retail outlet
- Set the decoupling point at stages 1, 2, 3, and 4
- Solve the PFQNs
### Decoupling Point Location Problem – Experiment

<table>
<thead>
<tr>
<th>Decoupling Point</th>
<th>Total Cost (Base Stock Policy)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Expensive Holding ↔ Expensive Delay</td>
<td><strong>Total Cost</strong>&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;br&gt;</td>
</tr>
<tr>
<td></td>
<td>$H_2 / H_1 = 10$</td>
<td>$H_2 / H_1 = 30$</td>
</tr>
<tr>
<td>1</td>
<td>3.596</td>
<td>5.940</td>
</tr>
<tr>
<td>2</td>
<td>5.215</td>
<td>6.963</td>
</tr>
<tr>
<td>3</td>
<td>8.967</td>
<td>10.237</td>
</tr>
<tr>
<td>4</td>
<td>12.071</td>
<td>12.429</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decoupling Point</th>
<th>Total Cost (Reorder Point Policy)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Expensive Holding ↔ Expensive Delay</td>
<td><strong>Total Cost</strong>&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;br&gt;&lt;br&gt;</td>
</tr>
<tr>
<td></td>
<td>$H_2 / H_1 = 10$</td>
<td>$H_2 / H_1 = 30$</td>
</tr>
<tr>
<td>1</td>
<td>3.427</td>
<td>5.853</td>
</tr>
<tr>
<td>2</td>
<td>4.234</td>
<td>6.060</td>
</tr>
<tr>
<td>3</td>
<td>5.686</td>
<td>6.974</td>
</tr>
<tr>
<td>4</td>
<td>8.055</td>
<td>8.414</td>
</tr>
</tbody>
</table>

As delay cost increases, the decoupling point is moving to right
Outline

- Revisiting Petri Nets
- Application 1: Software Syntheses
- Application 2: Biological Networks
- Application 3: Supply Chains
- Summary
Summary

- Pertti Net and its extension provide a wide range of applications
  - Software synthesis
  - Biological network
  - Supply chain