4. Petri Nets

- Introduction
- Different Classes of Petri Net
- Petri net properties
- Analysis of Petri net models
C.A Petri, TU Darmstadt, 1962

A mathematical and graphical modeling method.

Describe systems that are:

- concurrent
- asynchronous
- distributed
- nondeterministic
Petri Nets

Can be used at all stages of system development:

- modeling
- analysis
- simulation/visualization ("playing the token game")
- synthesis (Petri net versions of SCT)
- implementation (Grafcet)
Application areas

- communication protocols
- distributed systems
- distributed database systems
- flexible manufacturing systems
- logical controller design
- multiprocessor memory systems
- dataflow computing systems
- fault tolerant systems
- ...
A Petri net is a directed bipartite graph consisting of places $P$ and transitions $T$.

Places are represented by circles.

Transitions are represented by bars (or rectangles).

Places and transitions are connected by arcs.

In a marked Petri net each place contains a cardinal (zero or positive integer) number of tokens of marks.
DES

Firing rules

1. A transition $t$ is enabled if each input place contains at least one token.
2. An enabled transition may or may not fire.
3. Firing an enabled transition $t$ means removing one token from each input place of $t$ and adding one token to each output place of $t$.

The firing of a transition has zero duration.
The firing of a sink transition (only input places) only consumes tokens.
The firing of a source transition (only output places) only produces tokens.
DES
DES
DES
DES
DES
Typical interpretations of places and transitions:

<table>
<thead>
<tr>
<th>Input Places</th>
<th>Transition</th>
<th>Output Places</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditions</td>
<td>Event</td>
<td>Postconditions</td>
</tr>
<tr>
<td>Input data</td>
<td>Computation step</td>
<td>Output data</td>
</tr>
<tr>
<td>Input signals</td>
<td>Signal processor</td>
<td>Output signals</td>
</tr>
<tr>
<td>Resources needed</td>
<td>Task or job</td>
<td>Resources needed</td>
</tr>
<tr>
<td>Conditions</td>
<td>Clause in logic</td>
<td>Conclusions</td>
</tr>
<tr>
<td>Buffers</td>
<td>Processor</td>
<td>Buffers</td>
</tr>
</tbody>
</table>
DES

Characteristics

Concurrency

Conflict (decision, choice)

Confusion (conflict + concurrency)

\[ t_1 \text{ and } t_2 \text{ are concurrent while each of } t_1 \text{ and } t_2 \text{ is in conflict with } t_3 \]
The state of a marked Petri net is its marking

\[ M = [m(p_1), m(p_2), \cdots, m(p_n)] \].

The marking can be finite or infinite.

A Petri net can model an infinite system with a finite number of places.
Modeling with Petri Nets

Tradeoff between

Modeling power \iff Formal Analysis Power

*Petri net specializations:*

- restrictions on the allowed net structures
- more powerful analytical results and/or simpler algorithms

*Petri net abbreviations:*

- classes of Petri nets which always can be transformed to ordinary Petri nets
- PN properties are maintained
- “syntactic sugar”
Petri net extensions (generalizations):

- classes of Petri nets with additional transition firing rules
- cannot be transformed back to ordinary PNs
- PN properties are not always maintained
- often Turing machine equivalent
PN Specializations

*State Graphs (state machines):*

- an unmarked PN is a state graph iff every transition has one input place and one output place.
- a marked state graph is equivalent to a automaton state machine iff it only contains one token.
- can model conflicts.
- cannot model concurrency.
PN Specializations

Event graphs (marked graphs, transition graphs):

- a PN is an event graph iff every place has one input transition and one output transition
- cannot model conflicts
- can model concurrency
- the basis for the max-plus algebra approach

Other specializations:

- conflict-free PNs, free-choice PNs, simple PNs,
- pure PNs (have no self-loops)
PN abbreviations

- Generalized Petri nets
- Finite Capacity Petri nets
- High-Level Petri nets
Firing rules:

1. A transition $t$ is enabled if each input place $p$ of $t$ contains at least $w(p,t)$ tokens.

2. Firing a transition $t$ means removing $w(p,t)$ tokens from each input place $p$ and adding $w(t,q)$ tokens to each output place $q$. 
Firing rules:

1. A transition $t$ is enabled if each input place $p$ of $t$ contains at least $w(p, t)$ tokens.

2. Firing a transition $t$ means removing $w(p, t)$ tokens from each input place $p$ and adding $w(t, q)$ tokens to each output place $q$. 

Generalized Petri Nets
Finite-Capacity PN

Capacities (strictly positive integers) associated with places.

Transition firing rule:

- For a transition $t$ to be enabled it is additionally required that the number of tokens in each output place $p$ of $t$ will not exceed its capacity $K(p)$ after firing $t$.

Generalized, finite capacity PNs are sometimes called Place/Transition (P/T) nets.

Finite capacity nets where all places have capacity 1 and where input places are interpreted as preconditions, output places interpreted as postconditions, and transitions interpreted as events are called Condition/Event Systems. Separate theory.
High-Level Petri Nets

Predicate/Transition nets

Coloured Petri Nets

Abstract data types + Petri nets

A token has a type

Several identical Petri nets modeling, e.g., several identical concurrent activities can be folded into a high-level PN.

Can be transformed (unfolded) to an ordinary PN
PN extensions

- FIFO nets
  - each place represents a FIFO queue where the tokens are queued
- Inhibitor arc Petri nets (zero-test PNs)
  - an inhibitor arc connects a place to a transition
  - the inhibitor arc disables the transition when the place contains tokens and enables the transition when the place is empty
- Priority Petri nets
  - PN + a partial order relation on the transitions
Petri net properties

What can we do with the nets?

What properties and problems can be analyzed?

Properties can be divided into

- structural properties – marking independent
- behavioral properties – marking dependent
Reachability

A marking $M$ is reachable from a marking $M_0$ if there exists a sequence of firings that transforms $M_0$ to $M$. Denoted $M_0[T_1 T_2 \cdots T_n \triangleright M$.

$^*M_0 = \text{the set of markings reachable from } M_0$

The Reachability Problem:

- Decide whether a given marking $M \in^* M_0$ or not.
- Decidable but takes at least exponential space and time
**Boundedness:**

- A place $p_i$ is bounded for an initial marking $M_0$ if for all markings reachable from $M_0$, $m(p_i) \leq k$ (positive integer) – $p_i$ is $k$-bounded.

- A net is bounded for an initial marking $M_0$ if all places are bounded for $M_0$ (similarly for $k$-bounded).

- A net is *safe* iff it is 1-bounded

- An unmarked net is *structurally bounded* if the marked PN is bounded for all possible $M_0$

Boundedness and safeness are important if the places model buffers, inventories, etc.
Liveness:

- A transition \( t_i \) is live for an initial marking \( M_0 \) if for every reachable marking \( M_i \in^* M_0 \), a firing sequence \( S \) from \( M_i \) exists, which contains transition \( t_i \).
- A PN is live for \( M_0 \) if all its transitions are live for \( M_0 \).
- Interpretation: No matter what marking has been reached from \( M_0 \), it is possible to ultimately fire any transition of the net.
- A transition \( t \) is quasi-live for \( M_0 \) if there is a firing sequence from \( M_0 \) that contains \( t \).
- A PN is quasi-live for \( M_0 \) if all its transitions are quasi-live.
- A deadlock is a marking such that no transition is enabled.
- A PN is deadlock-free for \( M_0 \) if no reachable marking is a deadlock.
Reversibility:

• In a reversible net one can always get back to the initial marking.

Persistence:

• In a persistent net a transition, once it has been enabled, will stay enabled until it fires.
Let $R$ be a PN and $P$ the set of places. A marking invariant is obtained if there is a subset $P' = \{p_1, \cdots, p_r\}$ of $P$ and a weighting vector $[q_1, \cdots, q_r]$ for which all the weights are positive integers such that

$$q_1m(p_1) + q_2m(p_2) + \cdots + q_rm(p_r) = \text{constant}$$

for every $M$ reachable from $M_0$.

$P'$ is called a conservative component.

The conservative component property is structural.

The value of the constant depends on the marking.

Physical meaning:

- the system is in one and only one state at a time
- a number of entities is maintained
A firing sequence that causes a return to the initial marking is called a repetitive sequence.

\[ M_0[T_1T_2T_3 > M_0] \]

The set of transitions involved in the firing sequence is called a repetitive component.

A repetitive sequence containing all the transitions is a complete repetitive sequence.
Analysis Methods

How determine if a PN has a certain property?

Three types of methods:

- reachability (coverability) methods
- linear algebra methods
- reduction methods
Reachability tree

Create a tree where the nodes represent the reachable markings and the arcs represent the transition firings.

Stop the expansion of a node in the tree if the node already contained in the tree.

Only for bounded nets.

Properties are determined by examining the tree.

Exhaustive method, state space explosion.
DES

\[
\begin{align*}
(1000000) & \xrightarrow{T1} (011100) \\
(0110001) & \xrightarrow{T3} (1000000) \\
(000110) & \xrightarrow{T2} (1000000) \\
(010001) & \xrightarrow{T5} (1000000)
\end{align*}
\]
Reachability Graph

Fold all nodes with the same marking in the reachability tree

The finite state automaton equivalent of the Petri net.

It is possible to define the language generated by a Petri net.
Coverability trees

Unbounded PNs

Introduce a special symbol $\omega$ that corresponds to infinitely many tokens.
Unbounded PNs

Introduce a special symbol $\omega$ that corresponds to infinitely many tokens.
Unbounded PNs
Introduce a special symbol $\omega$ that corresponds to infinitely many tokens.
Coverability trees

Unbounded PNs

Introduce a special symbol $\omega$ that corresponds to infinitely many tokens.
Coverability trees

Unbounded PNs

Introduce a special symbol $\omega$ that corresponds to infinitely many tokens.
Incidence matrix:

\[
W = \begin{bmatrix}
  T_1 & T_2 & T_3 \\
  -1 & 1 & 0 \\
  -1 & 0 & 1 \\
  1 & -1 & 0 \\
  1 & 0 & -1
\end{bmatrix}
\]

Firing sequence: \( S1 = T_1 T_2 T_3 T_1 T_3 \)

Characteristic vector: \( S\overline{1} = [2 \ 1 \ 2]^T \)

Fundamental equation:

\[
M_i = M_0 + W S\overline{1}
\]
\[ M_i = M_0 + WS \]

\[
\begin{bmatrix}
0 \\
0 \\
1 \\
1 \\
0
\end{bmatrix} + \begin{bmatrix}
-1 & 0 & 0 & 1 \\
1 & -1 & 0 & 0 \\
1 & 0 & -1 & 0 \\
0 & 1 & 0 & -1 \\
0 & 0 & 1 & -1
\end{bmatrix} \begin{bmatrix}
1 \\
0 \\
2 \\
1 \\
1
\end{bmatrix} = \begin{bmatrix}
0 \\
0 \\
1 \\
1 \\
0
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
1 \\
-1 \\
1
\end{bmatrix} = \begin{bmatrix}
0 \\
0 \\
1 \\
0 \\
1
\end{bmatrix}
\]
Characteristic vectors can be composed through summation

\[ S = T_1 \]
\[ S' = T_2T_3T_4T_1 \]
\[ SS' = T_1T_2T_3T_4T_1 \]
\[ SS' = S + S' \]

\( S \) is a possible characteristic vector if at least one firing sequence \( S \) corresponds to it.

\( S = (1 \ 1 \ 0 \ 1)^T \) is not a possible characteristic vector

Several firing sequences may correspond to the same characteristic vector, e.g. \( T_1T_2T_3 \) and \( T_1T_2T_2 \) both correspond to \( S = (1 \ 1 \ 1 \ 0)^T \)
P-invariants

Let $F$ be a weighting vector for places (column vector)

Let $P(F)$ be the subset of $P$ whose coefficients in $F$ are nonzero.

Then:

- $P(F)$ is a conservative component iff a weighting vector $F$ exists such that $F^T W = 0$
  
  $F^T M_i = F^T M_0 + F^T W S$

- The vector $F$ is a P-invariant
- $F^T M_i$ is a marking invariant
T-invariants

Consider the characteristic vector $S$ associated with the firing sequence $S$. $T(S)$ is the support of $S$, i.e., the set of transitions appearing in $S$.

A set $D$ of transitions is a repetitive component iff a firing sequence $S$ exists such that $T(S) = D$ and $WS = 0$.

The vector $S$ is a T-invariant.
P- and T-invariants

Only nonnegative P-invariants and T-invariants are considered (P-semi-flows and T-semi-flows).

Minimal conservative components exist from which the other can be constructed by composition.

Algorithms exist for determining the minimal sets (e.g. Farkas algorithm).

Maple or Matlab
Example

Unmarked Net

Marked Net

\[
W = \begin{bmatrix}
-1 & 0 & 0 & 1 \\
1 & -1 & 0 & 0 \\
1 & 0 & -1 & 0 \\
0 & 1 & 0 & -1 \\
0 & 0 & 1 & -1 \\
\end{bmatrix}
\]
DES

P-invariants (structural property):
\[
F = (1, 1, 0, 1, 0) \\
G = (1, 0, 1, 0, 1) \\
F + G = (2, 1, 1, 1, 1)
\]

Conservative components (structural property):
\[
P(F) = \{P_1, P_2, P_4\} \\
P(G) = \{P_1, P_3, P_5\} \\
P(F + G) = \{P_1, P_2, P_3, P_4, P_5\}
\]

Marking invariants (marking dependent):
\[
m_1 + m_2 + m_4 = 1 \\
m_1 + m_3 + m_5 = 1 \\
2m_1 + m_2 + m_3 + m_4 + m_5 = 2
\]
T-invariants (structural property):

\[ S \quad = \quad (1, 1, 1, 1) \]

Repetitive sequences (marking dependent):

\[ S_1 \quad = \quad T_1 T_2 T_3 T_4 \]
\[ S_2 \quad = \quad T_1 T_3 T_2 T_4 \]
Reduction Methods

Idea: transform the net into a simpler net that preserves the properties of interest by successively applying a set of reduction rules.

Different reduction rules preserve different properties

- boundedness and liveness
- invariants

Semi-automatic method
Hierarchical Petri nets

Substitution places

Substitution transitions

Subnet
Modeling Power

- Petri Nets
  - Extended Petri nets
  - Bounded Petri nets
  - Finite state automata
  - Turing machines
Nonautonomous PNs

Petri Nets can be divided into:

- autonomous PNs
  - logical model
  - time is not involved
  - answers the question: HOW

- nonautonomous PNs
  - transition firing synchronized and/or timed
  - timed models
  - answers the question: WHEN
Nonautonomous PNs

- Synchronized Petri Nets
  - events associated with transitions
- Timed Petri Nets
  - time delays associated with places or transitions
- Interpreted Petri Nets
  - Synchronized and timed PN +
    - data processing part for computation of variables (actions) and transition conditions
  - very similar to Grafcet
- Controlled Petri Nets
  - transitions can be enabled and disabled by special control places
  - Supervisory Control Theory for PN
Projects

• Evaluate the analysis possibilities of one of the Petri net tools available in the public-domain. Develop a simple model and see what type of analysis that can be performed. Restrict yourself to standard (noncoloured Petri nets) – Lotta??

• Evaluate Design/CPN, one of the more widely spread tools for coloured Petri nets. Focus on modeling and simulation.