Modeling with Petri nets

prof.dr.ir. Wil van der Aalst
www.vdaalst.com
Recall: Strong point of Petri nets

Math. foundation

Graphical notation

Compactness

Last lecture

Concurrency, locality

Analysis techniques

Tool support
Strong point of Petri nets

- Math. foundation
- Graphical notation
- Compactness

Today’s lecture

- Concurrency, locality
- Analysis techniques
- Tool support
Modeling
Claim 1: There are many ways to model the same system

Example: Modeling of an elevator

Both nets behave exactly the same
Understandability vs compactness
Two models for the train example
Unlike previous slide, the models are different.
Claim 2: Many systems can be modeled in exactly the same way

Model of the four seasons

Model of the activities of an office employee

Reason: A model is an abstraction of reality.
The art of modeling

Understand the system (ask if under-specified)

Service help desk

Purpose of the model

In this lecture

How to map onto a net?

What is relevant?

Requires experience!!!
→ Exercises
→ Assignment
Quality of a model

• The one and only model does not exist!
• Quality depends on
  • simplicity
  • size
  • comprehensibility
• Quality is hard to measure (recall elevator example) and depends on intended use.
Executable models need to be correct!
We represent events as transitions, and we represent states as places and tokens (i.e., markings).
Role of a token

- **a physical object**—for example, a product, a part, a drug, or a person;
- **an information object**—for example, a message, a signal, or a report;
- **a collection of objects**—for example, a truck with products, a warehouse with parts, or an address file;
- **an indicator of a state**—for example, the indicator of the state in which a process is or the state of an object, such as a traffic light;
- **an indicator of a condition**: the presence of a token indicates whether a certain condition is fulfilled.
Role of a place

- a **communication medium**—for example, a telephone line, a middleman, or a communication network;
- a **buffer**—for example, a depot, a queue, or a post bin;
- a **geographical location**—for example, a place in a warehouse, in an office, or in a hospital;
- a **possible state or state condition**—for example, the floor where an elevator is or the condition that a specialist is available.
Role of a transition

• an event—for example, starting an operation, the death of a patient, a change seasons, or the turning of a traffic light from red to green;
• a transformation of an object—for example, repairing a product, updating a database, or stamping a document;
• a transport of an object—for example, transporting goods or sending a file.
Modeling Constructs
Typical network structures

- Causality
- Concurrency (AND-split – AND-join)
- Choice (XOR-split – XOR-join)
- Mutual exclusion
- Iteration (XOR-join – XOR-split)
Causality
Concurrency
Concurrent process instances are indistinguishable (uncolored tokens)
Concurrency: AND-split
Synchronization: AND-join
Choice: XOR-split
Choice: XOR-join
Iteration: 1 or more times

XOR-join before XOR-split
Iteration: 0 or more times

XOR-join before XOR-split
Additional modeling constructs

• Place capacities
  • Feedback loop
  • Mutual exclusion
  • Alternating
• Arc multiplicities
Place capacities

Supply chain example: Delivery of goods to a warehouse

Add complementary place free with:
for all markings \( m, m(\text{free}) + m(\text{in\_stock}) = 3 \)
Complementary place: general pattern

AND-join before AND-split
Complementary place: mutual exclusion

AND-join before AND-split
Capacity constraints: alternating

AND-join before AND-split
We have seen most patterns

mutual exclusion

alternating
Arc multiplicities

consume two tokens from free

consume two tokens from not_delivered

produce two tokens into in_stock

3 short hand for
For a Petri net \((P, T, F)\), let

\[ w: (P \times T) \cup (T \times P) \rightarrow \mathbb{N} \]

the weight function with

\[ w((x, y)) > 0 \text{ if } (x, y) \in F \text{ and } \]

\[ w((x, y)) = 0 \text{ if } (x, y) \notin F, \]

for all \((x, y) \in (P \times T) \cup (T \times P)\).

\[
\begin{align*}
w((p1, t)) &= 1 \\
w((p2, t)) &= 2 \\
w((p3, t)) &= 0 \\
w((t, p1)) &= 0 \\
w((t, p2)) &= 0 \\
w((t, p3)) &= 5
\end{align*}
\]
In a Petri net \((P, T, F)\) with weight function \(w\), a
transition \(t \in T\) is **enabled at marking** \(m: P \rightarrow \mathbb{N}\) if
for all \(p \in \bullet t\), \(m(p) \geq w((p, t))\).

\[
\begin{align*}
m(p1) &= 1 \\
m(p2) &= 2 \\
m(p3) &= 0
\end{align*}
\]

\(t\) is enabled at \(m\)
Transition firing with arc multiplicities (3)

For a Petri net \((P, T, F)\), let \(w\) be the weight function and let \(m: P \rightarrow \mathbb{N}\) be the current marking. A transition \(t \in T\) can fire if it is enabled at \(m\). The firing of \(t\) yields a new marking \(m': P \rightarrow \mathbb{N}\) where for all places \(p \in P\),

\[
m'(p) = m(p) - w((p, t)) + w((t, p)).
\]

\[
\begin{align*}
m'(p_1) &= 1 - 1 + 0 = 0 \\
m'(p_2) &= 2 - 2 + 0 = 0 \\
m'(p_3) &= 0 - 0 + 5 = 5
\end{align*}
\]
A Petri net system \((P, T, F, w, m_0)\) defines the following transition system \((S, TR, s_0)\):

- \(S = M = P \rightarrow N\)
- \(TR = \{ (m, m') \in S \times S \mid \exists t \in T : (\forall p \in \bullet t : m(p) \geq w((p, t))) \land (\forall p \in P : m'(p) = m(p) - w((p, t)) + w((t, p))) \}\)
- \(s_0 = m_0\)
Reachability graph algorithm (unchanged)

1) Label the initial marking $m_0$ as the root and tag it "new".

2) While "new" markings exists, do the following:
   a) Select a new marking $m$.
   b) If no transitions are enabled at $m$, tag $m$ "dead-end".
   c) While there exist enabled transitions at $m$, do the following for each enabled transition $t$ at $m$:
      i. Obtain the marking $m'$ that results from firing $t$ at $m$.
      ii. If $m'$ does not appear in the graph add $m'$ and tag it "new".
      iii. Draw an arc with label $t$ from $m$ to $m'$ (if not already present).

3) Output the graph
Example: Construct reachability graph

Nodes in the reachability graph can be represented by a vector “(3,2)” or as “[3·red,2·black]”. 

http://www.cs.utexas.edu/~EWD/ewd07xx/EWD720.PDF
Limited Expressiveness
Petri Nets are not Turing Complete
Zero testing

Transition $t$ should fire if place $p$ is empty.
Zero testing: solution

Only works if place $p$ is $k$-bounded

$$m_0(q) = k$$
Transition t1 has priority over t2

Solution similar to Zero testing!
Cancelation

Transition $t$ consumes all tokens from place $p$

Problem is similar to zero testing or priority.
Transition $t'$ must have higher priority than $t$
Cancelation: sketch solution

Replacing cancellation (reset arcs) by zero testing (inhibitor arcs)
A bit of theory

• Extensions have been proposed to tackle these problems
  • Inhibitor arcs for zero testing
  • Reset arcs for cancelation
• Extensions increase the modeling power, but limit analysis
• Most problems for Petri nets are decidable
  • Reachability is decidable
  • Equivalence is not decidable
• Most problems for inhibitor arcs undecidable
Reason for undecidability

• Petri nets with inhibitor arcs are Turing complete (can execute any computation).
• Proof idea: with inhibitor arc we can model a net that simulates a 2-counter machine.

• Petri nets are not Turing complete (arc multiplicities and place capacities do not increase expressiveness!).
• Still, certain questions are expensive.
Modeling convenience

Petri nets cannot model data and time easily
Mapping processes/systems to Petri nets

Service help desk

Abstraction may be used to avoid modeling irrelevant details, but there may be relevant aspects that cannot be modeled or modeling them results in spaghetti-like diagrams (data, time, etc.).
Distinguish different objects

• Tokens are indistinguishable (black dots)
• Can only be distinguished by different places (more precisely, place labels)
• Thus, place (or even a complete structure) needs to be duplicated

Euro  Pen  rather than  Euro_Pen
Example: Inventory management system

Duplicate network structure for each product
→ size linear in the number of products

- in1, increase1, bike, decrease1, out1
- in2, increase2, wheel, decrease2, out2
- in3, increase3, bell, decrease3, out3
- in4, increase4, steering_wheel, decrease4, out4
- in5, increase5, frame, decrease5, out5
Data-dependent choice

- Transitions may compete for the same token thus modeling choices, but often the choice depends on properties of the token.
- Transition labels have no formal semantics.

```
condition_holds
```

```
condition_doesNotHold
```
We can insert an additional transition to model the passing of time, but time is a global property and transitions may fire at different speeds.
need for high-level models

now

Later lectures

Tools
Bill of Material (BOM)

- car
- chassis
- engine
- wheel (4x)
- chair (2x)
- backseat
Model the assembly process

Three assembly steps: LA, BA, and FA.
John and Mary are doing LA and FA.
Pete is doing BA.
No need to distinguish individuals, but a person can work on only work on one assembly at a time.
Initial state: assume we need to assemble three chairs and have the inventory required for this.
CPN Tools solution

How to model a company-wide lunch break?

How to model changing roles?

How to model a kanban system?

Better or not?
How to model that FA has priority over LA?
Partial solution

Assumptions:
• no more than 100 LA’s (leg assemblies) waiting for final assembly,
• even when FA is waiting for BA, still keep resources free for FA.
How to model a global reset? 
(immediately stop all activities and restore initial state)
Assuming reset arcs

Without reset arcs a much more complicated model (basically enumerating all states, i.e., many reset transitions).
Two types of products

- Add a second product, a footstool, consisting of two leg assemblies and one seat.
- Production process is driven by availability.
- Model as a Petri net.
- Is this a safe way of modeling multiple products?
Exercise: Train system (1)

- Consider a circular railroad system with 4 (one-way) tracks (1,2,3,4) and 2 trains (A,B). No two trains should be at the same track at the same time and we do not care about the identities of the two trains.
Exercise: Train system (2)

• Consider a railroad system with 4 tracks (1,2,3,4) and 2 trains (A,B). No two trains should be at the same track at the same time and we want to distinguish the two trains.
Exercise: Train system (3)

• Consider a railroad system with 4 tracks (1,2,3,4) and 2 trains (A,B). No two trains should be at the same track at the same time. Moreover the next track should also be free to allow for a safe distance. (We do not care about train identities.)
Solution 3

See problem?
Exercise: Train system (4)

- Consider a railroad system with 4 tracks (1,2,3,4) and 2 trains. Tracks are free, busy or claimed. Trains need to claim the next track before entering.
Solution 4
Exercise: Dining philosophers

- 5 philosophers sharing 5 chopsticks: chopsticks are located in-between philosophers
- A philosopher is either in state eating or thinking and needs two chopsticks to eat.
- Model as a Petri net.
| 18 | 29-4-2013   | Lect. | TU/e closed Modeling with Petri nets (3) |
|    | 2-5-2013    | Inst. | Modeling with Petri nets                  |
|    | 3-5-2013    |       |                                          |
| 19 | 6-5-2013    | Lect. | Extending Petri nets with color and time (4) |
|    | 9-5-2013    |       | TU/e closed                               |
|    | 10-5-2013   |       | TU/e closed                               |
| 20 | 13-5-2013   | Exam  | Colored Petri Nets (5)                    |
|    | 16-5-2013   |       | Colored Petri Nets (6)                    |
|    | 17-5-2013   | Inst. | Pre-exam focusing on classical Petri nets (1 point) |
|    | (14.00-15.30)|       | Explanation “CPN assignment” (3 points)   |
|    | 17-5-2013   |       |                                           |
|    | (15.45-17.30)|       |                                           |

Read Chapter 4 of book.
Make all exercises in Section 2.

Read Chapter 5 of book.

Read Chapter 6 of book.
Read Chapter 6 of book.
Study Chapters 1-4 and all exercises in Sections 1-2.
Start making exercises in Section 3.

Also look at old exams.
After this lesson you should be able to

- Validate whether a given Petri net models a specification of a system or a business process.
- Know the different network structures and apply them.
- Be able to apply modeling constructs such as place capacities, arc multiplicities, mutual exclusion, shared resources, etc..
- Know the limitations of classical Petri nets.
- Be able to model non-trivial systems and processes as Petri nets.