10. Petri Nets

Prof. O. Nierstrasz
Roadmap

> Definition:
  — places, transitions, inputs, outputs
  — firing enabled transitions

> Modelling:
  — concurrency and synchronization

> Properties of nets:
  — liveness, boundedness

> Implementing Petri net models:
  — centralized and decentralized schemes

Roadmap

> **Definition:**
>  — places, transitions, inputs, outputs
>  — firing enabled transitions

> **Modelling:**
>  — concurrency and synchronization

> **Properties of nets:**
>  — liveness, boundedness

> **Implementing Petri net models:**
>  — centralized and decentralized schemes
Petri nets: a definition

A Petri net $C = \langle P, T, I, O \rangle$ consists of:

1. A finite set $P$ of places
2. A finite set $T$ of transitions
3. An input function $I: T \rightarrow \text{Nat}^P$ (maps to bags of places)
4. An output function $O: T \rightarrow \text{Nat}^P$

A marking of $C$ is a mapping $m: P \rightarrow \text{Nat}$

Example:

$P = \{ x, y \}$
$T = \{ a, b \}$
$I(a) = \{ x \}$, $I(b) = \{ x, x \}$
$O(a) = \{ x, y \}$, $O(b) = \{ y \}$
$m = \{ x, x \}$
Firing transitions

To fire a transition $t$:

1. $t$ must be enabled: $m \geq I(t)$
2. *consume* inputs and *generate* output: $m' = m - I(t) + O(t)$
Roadmap

> Definition:
  — places, transitions, inputs, outputs
  — firing enabled transitions

> Modelling:
  — concurrency and synchronization

> Properties of nets:
  — liveness, boundedness

> Implementing Petri net models:
  — centralized and decentralized schemes
Modelling with Petri nets

*Petri nets are good for modelling:*
> concurrency
> synchronization

*Tokens can represent:*
> resource availability
> jobs to perform
> flow of control
> synchronization conditions ...
Concurrency

Independent inputs permit “concurrent” firing of transitions
Conflict

Overlapping inputs put transitions in conflict

Only one of $a$ or $b$ may fire
The two subnets are forced to synchronize
Fork and Join
Producers and Consumers
Bounded Buffers
Roadmap

> **Definition:**
  — places, transitions, inputs, outputs
  — firing enabled transitions

> **Modelling:**
  — concurrency and synchronization

> **Properties of nets:**
  — liveness, boundedness

> **Implementing Petri net models:**
  — centralized and decentralized schemes
Reachability and Boundedness

**Reachability:**
> The reachability set $R(C, \mu)$ of a net $C$ is the set of all markings $\mu'$ reachable from initial marking $m$.

**Boundedness:**
> A net $C$ with initial marking $\mu$ is **safe** if places always hold at most 1 token.
> A marked net is (k-)bounded if places never hold more than $k$ tokens.
> A marked net is **conservative** if the number of tokens is constant.
Liveness and Deadlock

**Liveness:**
- A transition is **deadlocked** if it can *never fire*.
- A transition is **live** if it can *never deadlock*.

This net is both **safe and conservative**.
- Transition a is **deadlocked**.
- Transitions b and c are **live**.
- The reachability set is \{\{y\}, \{z\}\}.

Are the examples we have seen bounded? Are they live?
Related Models

**Finite State Processes**

> Equivalent to *regular expressions*
> Can be modelled by *one-token conservative nets*

The FSA for: $a(blc)^*d$
Some Petri nets can be modelled by FSPs

Precisely which nets can (cannot) be modelled by FSPs?
Zero-testing Nets

*Petri nets are not computationally complete*

> Cannot model “zero testing”
> Cannot model priorities

A zero-testing net: An equal number of a and b transitions may fire as a sequence during any sequence of matching c and d transitions.

\((\#a \geq \#b, \#c \geq \#d)\)
Other Variants

*There exist countless variants of Petri nets*

**Coloured Petri nets:**
> Tokens are “coloured” to represent different kinds of resources

**Augmented Petri nets:**
> Transitions additionally depend on external conditions

**Timed Petri nets:**
> A duration is associated with each transition
Applications of Petri nets

Modelling information systems:
> Workflow
> Hypertext (possible transitions)
> Dynamic aspects of OODB design
> **Definition:**
  — places, transitions, inputs, outputs
  — firing enabled transitions

> **Modelling:**
  — concurrency and synchronization

> **Properties of nets:**
  — liveness, boundedness

> **Implementing Petri net models:**
  — centralized and decentralized schemes
Implementing Petri nets

We can implement Petri net structures in either *centralized* or *decentralized* fashion:

**Centralized:**

> A single “net manager” monitors the current state of the net, and fires enabled transitions.

**Decentralized:**

> Transitions are processes, places are shared resources, and transitions compete to obtain tokens.
Centralized schemes

In one possible centralized scheme, the Manager selects and fires enabled transitions.

Concurrently enabled transitions can be fired in parallel.

What liveness problems can this scheme lead to?
Decentralized schemes

In decentralized schemes transitions are processes and tokens are resources held by places:

Transitions can be implemented as *thread-per-message gateways* so the same transition can be fired more than once if enough tokens are available.
Transactions

Transitions attempting to fire must grab their input tokens as an atomic transaction, or the net may deadlock even though there are enabled transitions!

If $a$ and $b$ are implemented by independent processes, and $x$ and $y$ by shared resources, this net can deadlock even though $b$ is enabled if $a$ (incorrectly) grabs $x$ and waits for $y$. 
Coordinated interaction

A simple solution is to treat the state of the entire net as a single, shared resource:

After a transition fires, it notifies waiting transitions.

How could you refine this scheme for a distributed setting?
Petit Petri — a Petri Net Editor built with Etoys

Firing transitions

A transition is enabled if there are tokens in all its input places.

An enabled transition is fired by removing a token from each input place, and inserting a token in each output place.
Etoys implementation
Examples

- Mutual Exclusion
  - The two philosophers are forced to synchronize their critical sections.

- Fork and Join

- Bounded Buffer
  - Producer
  - Consumer

- Producers and Consumers
  - Producer
  - Consumer

- Dining Philosophers
  - 5 philosophers
  - 5 forks
  - 5 left and right tools

- Dining Philosophers (transactional)

- The Star Game
  - Fire green transitions to get tokens out of all the yellow places and into the purple places.
What you should know!

> How are Petri nets formally specified?
> How can nets model concurrency and synchronization?
> What is the “reachability set” of a net? How can you compute this set?
> What kinds of Petri nets can be modelled by finite state processes?
> How can a (bad) implementation of a Petri net deadlock even though there are enabled transitions?
> If you implement a Petri net model, why is it a good idea to realize transitions as “thread-per-message gateways”?
Can you answer these questions?

> What are some simple conditions for guaranteeing that a net is bounded?

> How would you model the Dining Philosophers problem as a Petri net? Is such a net bounded? Is it conservative? Live?

> What could you add to Petri nets to make them Turing-complete?

> What constraints could you put on a Petri net to make it fair?
License

http://creativecommons.org/licenses/by-sa/2.5/

You are free:
• to copy, distribute, display, and perform the work
• to make derivative works
• to make commercial use of the work

Under the following conditions:

BY: Attribution. You must attribute the work in the manner specified by the author or licensor.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

• For any reuse or distribution, you must make clear to others the license terms of this work.
• Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.