Network analysis

*Every solution* to Hw4 had a serious problem. *BPMN is not a flowcharting language.* Instead, it borrows its meanings from

Petri nets

$\pi$-calculus
Some common BPMN weaknesses included:

- **Deadlock**: two processes wait for each other.
- **Ambiguity**: unless sequencing is explicit, it is not implied by position.
- **Exclusive**: will not proceed.
- **Over-constrained**: C needs messages from both A and B to proceed.

Better alternatives include:

- Combine requests, send one message.
Petri nets

A general model of parallel computation based upon token passing.
- Source nodes generate tokens.
- Sink nodes eat tokens.
- Transition nodes transform tokens in input to tokens on output.

Computation proceeds when tokens are available.
- Consume input tokens
- Generate (one or more) output tokens
Petri Net Notation

Places: can contain state. Notated by circles.

Transitions: change state. Notated as bars.

Example:

A \rightarrow B \rightarrow C

means there is a sequence of transitions between states A, B, C.
Petri sequences

Petri sequences

in represented as

\[ \text{A} \rightarrow \text{B} \rightarrow \text{C} \rightarrow \ldots \]
Petri net tokens

Model of execution: token passing
Tokens are generated by sources, accumulate at places, and are consumed by transitions, generating new tokens.
Petri net rules

Every place node must be connected only to transition nodes.
Every transition node must be connected only to place nodes.
Thus the Petri net is \textbf{bipartite} with two kinds of nodes.

\[\bigcirc = \text{place node} = \text{state}\]
\[\square = \text{transition node}\]
\[\bullet = \text{place node with token}\]
Petri net constructions

\[
\begin{array}{c}
\text{a decision: only } A, B \text{ can fire at a time.}
\\
\text{exclusive-or}
\\
\text{a join: either } C \text{ or } D \text{ can make the transition to } Z
\end{array}
\]
A Petri model of BPMN

Message arrows and precedence arrows become token arrows.
Start nodes become token sources.
End nodes become token sinks.
Implicit transition nodes become explicit.
From BPMN to Petri net

Notes:
Message receipt
causes a state transition, i.e., a token in a new place.
is a prerequisite for the next step.
Or, sorting things out a bit
Petri Net Computation: after 1st transition

Monday, April 12, 2010
2:24 PM
Petri Net Computation: after 3rd transition
There is a minor ambiguity in translating BPMN to Petri Nets

![Diagram]

could be modeled as

![Alternative Diagram]

need B + message before C

need message before B
The four constructions

Monday, April 12, 2010
8:11 PM

exclusive-OR: decision

parallel: both at same time

and: both must have token

either/or: inclusive-OR
Some easy transformations

Monday, April 12, 2010
12:47 PM
Consequences of receiving two messages

Becomes

must have all three to transition
Expressing alternative messages

one & A = B happens.
D reports.

In other words, one state results from two things!
Why the Petri model is important

- Exposes ambiguities in BPMN.
- Allows easy deadlock detection.
- Correctly models message-passing.
Poor BPMN practices reconsidered

either one first, or even in parallel ...

A → O

means

B → O
Deadlock
BPMN concepts and Petri-net extensions

Monday, April 12, 2010
2:57 PM

BPMN and Petri net extensions

[Diagram of BPMN and Petri net extensions]

Some time delay must pass.

Non-standard notation.
Alas, Petri nets aren't everything
Petri nets allow gotos.
BPEL doesn't.
This is a language restriction.
\(\pi\)-calculus

Petri nets aren't very expressive
  No notation as to how data is used.
  No concept of "computation".
\(\pi\)-calculus: attempts to describe computation with a Petri-like model.
  Concepts of data and binding.
  Formal reduction as a computing model.
A parallel version of lambda-calculus.