Petri Nets

ee249 Fall 2000

Marco Sgroi

Most slides borrowed from
Luciano Lavagno’s lecture ee249 (1998)

Models Of Computation
for reactive systems

• Main MOCs:
  – Communicating Finite State Machines
  – Dataflow Process Networks
  – Discrete Event
  – Codesign Finite State Machines
  – Petri Nets

• Main languages:
  – StateCharts
  – Esterel
  – Dataflow networks
Outline

- Petri nets
  - Introduction
  - Examples
  - Properties
  - Analysis techniques
  - Scheduling

Petri Nets (PNs)

- Model introduced by C.A. Petri in 1962
  - Ph.D. Thesis: “Communication with Automata”
- Applications: distributed computing, manufacturing, control, communication networks, transportation…
- PNs describe explicitly and graphically:
  - sequencing/causality
  - conflict/non-deterministic choice
  - concurrency
- Asynchronous model (partial ordering)
- Main drawback: no hierarchy
Petri Net Graph

- Bipartite weighted directed graph:
  - Places: circles
  - Transitions: bars or boxes
  - Arcs: arrows labeled with weights
- Tokens: black dots

Petri Net

- A PN \((N,M_0)\) is a Petri Net Graph \(N\)
  - places: represent distributed state by holding tokens
    - marking (state) \(M\) is an \(n\)-vector \((m_1,m_2,m_3,\ldots)\), where \(m_i\) is the non-negative number of tokens in place \(p_i\).
    - initial marking \((M_0)\) is initial state
  - transitions: represent actions/events
    - enabled transition: enough tokens in predecessors
    - firing transition: modifies marking
- ...and an initial marking \(M_0\).

Places/Transition: conditions/events
Transition firing rule

- A marking is changed according to the following rules:
  - A transition is **enabled** if there are enough tokens in each input place
  - An enabled transition **may or may not fire**
  - The firing of a transition modifies marking by **consuming** tokens from the input places and **producing** tokens in the output places

Concurrency, causality, choice
Concurrency, causality, choice

Concurrent tasks:
- \( t_1 \)
- \( t_2 \)
- \( t_3 \)
- \( t_4 \)
- \( t_5 \)
- \( t_6 \)

Causality, sequencing:
- \( t_3 \rightarrow t_4 \)
- \( t_5 \rightarrow t_6 \)
Concurrency, causality, choice

Concurrent, causality, choice

Concurrent, causality, choice
Confusion

- t1 and t2 are concurrent but their firing order is not irrelevant for conflict resolution (not local choice)
- From (1,1,0,0,0):
  - solving a conflict (t1,t2) (0,0,0,1,0), (0,0,1,0,0)
  - not solving a conflict (t2,t1) (0,0,1,1,0)

Communication Protocol
Communication Protocol

P1

Send msg

Receive Ack

P2

Receive msg

Send Ack

Communication Protocol

P1

Send msg

Receive Ack

P2

Receive msg

Send Ack

P1

Send msg

Receive Ack

P2

Receive msg

Send Ack

P1

Send msg

Receive Ack

P2

Receive msg

Send Ack
Communication Protocol

P1
- Send msg
- Receive Ack

P2
- Receive msg
- Send Ack
- Receive msg
- Send Ack

Communication Protocol

P1
- Send msg
- Receive Ack

P2
- Receive msg
- Send Ack
- Receive msg
- Send Ack

17

18
Communication Protocol

Producer-Consumer Problem
Producer-Consumer Problem

Produce
Buffer
Consume

Producer-Consumer Problem

Produce
Buffer
Consume
Producer-Consumer Problem

Produce
Buffer
Consume

Producer-Consumer Problem

Produce
Buffer
Consume
Producer-Consumer Problem

Produce

Buffer

Consume

Producer-Consumer Problem

Produce

Buffer

Consume
Producer-Consumer Problem

Produce
Buffer
Consume

Produce-Consumer Problem

Produce
Buffer
Consume
Producer-Consumer Problem

Produce

Buffer

Consume
Producer-Consumer Problem

Producer-Consumer Problem

Producer-Consumer Problem
**Producer-Consumer Problem**

![Diagram of Producer-Consumer Problem]

1. Produce
2. Buffer
3. Consume

**Producer-Consumer Problem**

![Diagram of Producer-Consumer Problem]

1. Produce
2. Buffer
3. Consume
Producer-Consumer with priority

Consumer B can consume only if buffer A is empty

Inhibitor arcs

PN properties

- Behavioral: depend on the initial marking (most interesting)
  - Reachability
  - Boundedness
  - Schedulability
  - Liveness
  - Conservation

- Structural: do not depend on the initial marking (often too restrictive)
  - Consistency
  - Structural boundedness
Reachability

• Marking $M$ is reachable from marking $M_0$ if there exists a sequence of firings $\sigma = M_0 t_1 M_1 t_2 M_2 \ldots M$ that transforms $M_0$ to $M$.
• The reachability problem is decidable.

![Diagram of a Petri net with transitions and markings](image)

$M_0 = (1,0,1,0)$
$M = (1,1,0,0)$

Liveness

• Liveness: from any marking any transition can become fireable
  – Liveness implies deadlock freedom, not vice versa

![Diagram of a Petri net with transition and markings](image)

$M_0 = (1,0,1,0)$
$M_1 = (1,0,0,1)$
$M = (1,1,0,0)$
Liveness

• **Liveness:** from any marking any transition can become fireable
  – Liveness implies deadlock freedom, not viceversa

Liveness

• **Liveness:** from any marking any transition can become fireable
  – Liveness implies deadlock freedom, not viceversa
Liveness

- **Liveness**: from any marking any transition can become fireable
  - Liveness implies deadlock freedom, not vice versa

Boundedness

- **Boundedness**: the number of tokens in any place cannot grow indefinitely
  - (1-bounded also called safe)
  - Application: places represent buffers and registers (check there is no overflow)
Boundedness

- **Boundedness**: the number of tokens in any place cannot grow indefinitely
  - (1-bounded also called *safe*)
  - Application: places represent buffers and registers (check there is no overflow)

![Diagram of boundedness](image_url)
**Boundedness**

- **Boundedness**: the number of tokens in any place cannot grow indefinitely
  - (1-bounded also called *safe*)
  - Application: places represent buffers and registers (check there is no overflow)
Conservation

- **Conservation**: the total number of tokens in the net is constant

Not conservative

Conservation

- **Conservation**: the total number of tokens in the net is constant

Not conservative
Conservation

- **Conservation**: the total number of tokens in the net is constant

Analysis techniques

- **Structural analysis techniques**
  - Incidence matrix
  - T- and S- Invariants
- **State Space Analysis techniques**
  - Coverability Tree
  - Reachability Graph
Incidence Matrix

- Necessary condition for marking M to be reachable from initial marking $M_0$:
  
  there exists firing vector $v$ s.t.:
  
  $$M = M_0 + A v$$

State equations

- E.g. reachability of $M = [0 0 1]^T$ from $M_0 = [1 0 0]^T$

$$v_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

but also $v_2 = [1 1 2]^T$ or any $v_k = [1 (k) (k+1)]^T$
Necessary Condition only

Firing vector: (1,2,2)  Deadlock!!

State equations and invariants

- Solutions of $Ax = 0$ (in $M = M_0 + Ax$, $M = M_0$)

  T-invariants
  - sequences of transitions that (if fireable) bring back to original marking
  - periodic schedule in SDF
  - e.g. $x = [0 \ 1 \ 1]^T$

\[
A = \begin{pmatrix}
-1 & 0 & 0 \\
1 & 1 & -1 \\
0 & -1 & 1
\end{pmatrix}
\]
Application of T-invariants

• Scheduling
  – Cyclic schedules: need to return to the initial state

\[
\text{Schedule: } i *k_2 *k_1 + o
\]

T-invariant: (1,1,1,1)

State equations and invariants

• Solutions of \( yA = 0 \)

S-invariants
  – sets of places whose weighted total token count does not change after the firing of any transition \( yM = yM' \)
  – e.g. \( y = [1 \ 1 \ 1] \)

\[
A_T = \begin{bmatrix}
-1 & 1 & 0 \\
0 & 1 & -1 \\
0 & -1 & 1
\end{bmatrix}
\]
Application of S-invariants

- Structural Boundedness: bounded for any finite initial marking $M_0$
- Existence of a positive S-invariant is CS for structural boundedness
  - initial marking is finite
  - weighted token count does not change

Summary of algebraic methods

- Extremely efficient
  (polynomial in the size of the net)
- Generally provide only necessary or sufficient information
- Excellent for ruling out some deadlocks or otherwise dangerous conditions
- Can be used to infer structural boundedness
Coverability Tree

- Build a (finite) tree representation of the markings

Karp-Miller algorithm
- Label initial marking $M_0$ as the root of the tree and tag it as new
- While new markings exist do:
  - select a new marking $M$
  - if $M$ is identical to a marking on the path from the root to $M$, then tag $M$ as old and go to another new marking
  - if no transitions are enabled at $M$, tag $M$ dead-end
  - while there exist enabled transitions at $M$ do:
    • obtain the marking $M'$ that results from firing $t$ at $M$
    • on the path from the root to $M$ if there exists a marking $M''$ such that $M'(p) \geq M''(p)$ for each place $p$ and $M'$ is different from $M''$, then replace $M'(p)$ by $\omega$ for each $p$ such that $M'(p) > M''(p)$
    • introduce $M'$ as a node, draw an arc with label $t$ from $M$ to $M'$ and tag $M'$ as new.

Boundedness is decidable with coverability tree
Coverability Tree

- Boundedness is decidable with *coverability tree*

```
1000
↓ t1
0100
```

```
0011
↓ t3
```

Coverability Tree

- Boundedness is decidable with *coverability tree*

```
1000
↓ t1
0100
```

```
0011
↓ t3
```
Coverability Tree

- Boundedness is decidable with coverability tree

Cannot solve the reachability and liveness problems
Coverability Tree

- Boundedness is decidable with *coverability tree*

- Cannot solve the reachability and liveness problems

Reachability graph

- For bounded nets the Coverability Tree is called Reachability Tree since it contains all possible reachable markings
For bounded nets the Coverability Tree is called Reachability Tree since it contains all possible reachable markings.

Reachability graph

- For bounded nets the Coverability Tree is called Reachability Tree since it contains all possible reachable markings.
Reachability graph

- For bounded nets the Coverability Tree is called Reachability Tree since it contains all possible reachable markings.

Subclasses of Petri nets

- Reachability analysis is too expensive
- State equations give only partial information
- Some properties are preserved by reduction rules, e.g., for liveness and safeness.
- Even reduction rules only work in some cases
- Must restrict class in order to prove stronger results.
Subclasses of Petri nets: SMs

- State machine: every transition has at most 1 predecessor and 1 successor
- Models only causality and conflict
  - (no concurrency, no synchronization of parallel activities)

Subclasses of Petri nets: MGs

- Marked Graph: every place has at most 1 predecessor and 1 successor
- Models only causality and concurrency (no conflict)
- Same as underlying graph of SDF
Subclasses of Petri nets: FC nets

- Free-Choice net: every transition after choice has exactly 1 predecessor

Free-Choice Petri Nets (FCPN)

Free-Choice (FC)

Confusion (not-Free-Choice)  Extended Free-Choice

Free-Choice: the outcome of a choice depends on the value of a token (abstracted non-deterministically) rather than on its arrival time.

Easy to analyze