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 \(M_0\).
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

Concurrency

Causality, sequencing

Concurrent, causality, choice
Concurrency, causality, choice

Concurrent processes and their interactions:
- t1, t2, t3, t4, t5, t6
- Temporal order and parallel execution
- Conflict resolution

Diagram illustrating the sequence of events and decision points.
Confusion

- \( t_1 \) and \( t_2 \) are concurrent but their firing order is not irrelevant for conflict resolution (not local choice)
- From \((1,1,0,0,0)\):
  - solving a conflict \((t_1,t_2)\) \((0,0,0,0,1),(0,0,1,1,0)\)
  - not solving a conflict \((t_2,t_1)\) \((0,0,1,1,0)\)

```
\[3\]
```

Communication Protocol

```
\[4\]
```
Communication Protocol

P1
Send msg
Receive Ack
Send Ack
Receive msg

P2

Communication Protocol

P1
Send msg
Receive Ack
Send Ack
Receive msg

P2
Communication Protocol

Producer-Consumer Problem
Producer-Consumer Problem

Produce
Buffer
Consume
Producer-Consumer Problem

Producer → Buffer → Consume

Produce

Buffer

Consume

Producer-Consumer Problem

Produce → Buffer → Consume

Produce

Buffer

Consume
Producer-Consumer Problem

Produce -> Buffer -> Consume

Producer-Consumer Problem

Produce -> Buffer -> Consume
Producer-Consumer Problem

Producer

Buffer

Consumer

Produce

Buffer

Consume
Producer-Consumer Problem

Produce

Buffer

Consume

Producer-Consumer Problem

Produce

Buffer

Consume
Producer-Consumer Problem

Producer

Buffer

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.

Liveness

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

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

![Diagram of liveness and deadlock freedom](image_url)
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)
• **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

Conservation

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

Not conservative

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.:
  \[ M = M_0 + A v \]

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

\[ t_1 \quad t_2 \quad t_3 \]

State equations

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

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

\[ t_1 \quad t_2 \quad t_3 \]

\[ v_1 = \begin{bmatrix}
1 \\
0 \\
1 \\
\end{bmatrix} \]

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. $x = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^T$

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$
Application of T-invariants

- Scheduling
  - Cyclic schedules: need to return to the initial state

\[ T \text{-invariant: } (1,1,1,1,1) \]
\[ \text{Schedule: } i \cdot k_2 \cdot k_1 + o \]

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. \( y = [1 \ 1 \ 1]^T \)

\[ A^T = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \]
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 M₀ 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)≥M″(p) for each place p and M′ is different from M″, then replace M′(p) by 0 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*.

Coverability Tree

- Boundedness is decidable
  
  with *coverability tree*
Coverability Tree

- Boundedness is decidable with coverability tree

\[ \text{Coverability Tree} \]

\[ \text{Boundedness is decidable with coverability tree} \]

\[ \begin{align*}
\text{Coverability Tree} & \quad \text{Boundedness is decidable} \\
& \quad \text{with coverability tree} \\
\end{align*} \]
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
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
Free-Choice nets

- Introduced by Hack (‘72)
- Extensively studied by Best (‘86) and Desel and Esparza (‘95)
- Can express concurrency, causality and choice without confusion
- Very strong structural theory
  - necessary and sufficient conditions for liveness and safeness, based on decomposition
  - concurrency, causality and choice relations are mutually exclusive
  - exploits duality between MG and SM

MG (& SM) decomposition

- An Allocation is a control function that chooses which transition fires among several conflicting ones (A: P → T).
- Eliminate the subnet that would be inactive if we were to use the allocation...
- Reduction Algorithm
  - Delete all unallocated transitions
  - Delete all places that have all input transitions already deleted
  - Delete all transitions that have at least one input place already deleted
- Obtain a Reduction (one for each allocation) that is a conflict free subnet
MG reduction and cover

- Choose one successor for each conflicting place:
MG reduction and cover

* Choose one successor for each conflicting place:
MG reduction and cover

• Choose one successor for each conflicting place:

MG reductions

• The set of all reductions yields a cover of MG components (T-invariants)
MG reductions

- The set of all reductions yields a cover of MG components (T-invariants)

SM reduction and cover

- Choose one predecessor for each transition:
SM reduction and cover

- Choose one predecessor for each transition:

• The set of all reductions yields a cover of SM components (S-invariants)
Hack’s theorem (’72)

- Let N be a Free-Choice PN:
  - N has a live and safe initial marking (well-formed) if and only if
    - every MG reduction is strongly connected and not empty, and the set of all reductions covers the net
    - every SM reduction is strongly connected and not empty, and the set of all reductions covers the net

Hack’s theorem

- Example of non-live (but safe) FCN
Hack’s theorem

- Example of non-live (but safe) FCN
Hack’s theorem

• Example of non-live (but safe) FCN

![Diagram]

Hack’s theorem

• Example of non-live (but safe) FCN

![Diagram]
Hack’s theorem

- Example of non-live (but safe) FCN
Hack’s theorem

- Example of non-live (but safe) FCN

Hack’s theorem

- Example of non-live (but safe) FCN
Hack’s theorem

- Example of non-live (but safe) FCN
Hack’s theorem

- Example of non-live (but safe) FCN

Hack’s theorem

- Example of non-live (but safe) FCN
Hack’s theorem

• Example of non-live (but safe) FCN

Hack’s theorem

• Example of non-live (but safe) FCN
Hack’s theorem

• Example of non-live (but safe) FCN

![Not live graph]

Other results for LSFC nets

• Let $t_1$ and $t_2$ be two transitions of a live and safe Free-Choice net.

Then $t_1$ and $t_2$ are:
  – **sequential** if
    - there exists a simple cycle to which both belong
  – **concurrent** if
    - they are not ordered, and
    - there exists an MG component to which both belong
  – **conflicting** otherwise
Summary of LSFC nets

- Largest class for which structural theory really helps
- Structural component analysis may be expensive
  (exponential number of MG and SM components in the worst case)
- But...
  - number of MG components is generally small
  - FC restriction simplifies characterization of behavior

Petri Net extensions

- Add interpretation to tokens and transitions
  - Colored nets (tokens have value)
- Add time
  - Time/timed Petri Nets (deterministic delay)
    - type (duration, delay)
    - where (place, transition)
  - Stochastic PNs (probabilistic delay)
  - Generalized Stochastic PNs (timed and immediate transitions)
- Add hierarchy
  - Place Charts Nets
Summary of Petri Nets

• Graphical formalism
• Distributed state (including buffering)
• Concurrency, sequencing and choice made explicit
• Structural and behavioral properties
• Analysis techniques based on
  – linear algebra (only sufficient)
  – structural analysis (necessary and sufficient only for FC)