Petri Nets

ee249

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
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: bar or boxes
  - Arcs: arrows labeled with weights

- Tokens: black dots
A PN (N, M₀) is a Petri Net Graph N

- places: represent distributed state by holding tokens
  - marking (state) M is an n-vector (m₁,m₂,m₃…), where mᵢ is the non-negative number of tokens in place pᵢ.
  - initial marking (M₀) is initial state

- transitions: represent actions/events
  - enabled transition: enough tokens in predecessors
  - firing transition: modifies marking

...and an initial marking M₀.
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

\[ \begin{align*}
\text{t1} & \\
\text{t2} & \\
\text{t3} & \text{t4} & \text{t5} & \text{t6}
\end{align*} \]
Concurrency, causality, choice

Concurrency

Concurrent processes and their interactions.
Concurrency, causality, choice

Causality, sequencing

\[ t_1, t_2, t_3, t_4, t_5, t_6 \]
Concurrency, causality, choice

Choice, conflict
Concurrency, 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) to (0,0,1,1)
  - solving a conflict (t1,t2)
  - not solving a conflict (t2,t1)
Confusion

- T1 and t2 are concurrent but their firing order is not irrelevant for conflict resolution (not local choice)
- From (1,1,0,0) to (0,0,1,1)
  - solving a conflict (t1,t2)
  - not solving a conflict (t2,t1)
Confusion

- T1 and t2 are concurrent but their firing order is not irrelevant for conflict resolution (not local choice)
- From (1,1,0,0) to (0,0,1,1)
  - solving a conflict (t1,t2)
  - not solving a conflict (t2,t1)
Confusion

- T1 and t2 are concurrent but their firing order is not irrelevant for conflict resolution (not local choice)
- From (1,1,0,0) to (0,0,1,1)
  - solving a conflict (t1,t2)
  - not solving a conflict (t2,t1)
Communication Protocol

P1

Send msg

Receive Ack

Send Ack

Receive msg

P2
Communication Protocol

P1

Send msg

Receive Ack

Receive msg

Send Ack

P2

Receive Ack
Communication Protocol

P1
- Send msg
- Receive Ack
- Send Ack
- Receive Ack

P2
- Receive msg
- Send Ack
Communication Protocol

P1:
- Send msg
- Receive Ack
- Send Ack
- Receive msg

P2:
- Receive msg
- Send Ack
- Receive Ack
Communication Protocol

P1
  ↓
  ⊙
  ↓
  ⊙
  ↓
  ⊙
  ↓
  ⊙
  ↓
  ⊙

Send msg

Receive Ack

Send Ack

Receive msg

P2
Communication Protocol

P1

Send msg

Receive Ack

Send Ack

Receive msg

P2

Receive Ack

Send Ack
Producer-Consumer Problem

Produce

Buffer

Consume
Producer-Consumer Problem

Produce
Buffer
Consume
Producer-Consumer Problem
Producer-Consumer Problem
Producer-Consumer Problem
Producer-Consumer Problem

Produce

Buffer

Consume
Producer-Consumer Problem
Producer-Consumer Problem

Produce

Buffer

Consume
Producer-Consumer Problem

Produce

Buffer

Consume
Producer-Consumer Problem
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.

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

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

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

- **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

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

Not conservative
(Strict) Conservation: the total number of tokens in the net is constant

Not conservative
(Strict) 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
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 \( M = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \) from \( M_0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \)

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

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

but also \( v_2 = \begin{bmatrix} 1 & 1 & 2 \end{bmatrix}^T \) or any \( v_k = \begin{bmatrix} 1 & (k) & (k+1) \end{bmatrix}^T \)
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

T-invariant: (1,1,1,1,1)
Schedule: i *k2 *k1 + o
State equations and invariants

- Solutions of $yA = 0$

**S-invariants**

- sets of places whose total token count does not change after the firing of any transition ($y \, M = y \, M'$)
- 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
Boundedness is decidable with coverability tree
Coverability Tree

- Boundedness is decidable
  with *coverability tree*
Coverability Tree

- Boundedness is decidable with coverability tree
Boundedness is decidable
with *coverability tree*
Coverability Tree

- Boundedness is decidable with *coverability tree*

Cannot solve the reachability and liveness problems
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.
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
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
- Studied by Commoner et al. (‘71)
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)**
  - The outcome of a choice depends on the value of a token (abstracted non-deterministically) rather than on its arrival time.
  - Easy to analyze

- **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 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)
SM reduction and cover

- Choose one predecessor for each transition:
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
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
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
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
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)
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)
    - control (weak, strong)
  - Stochastic PNs (probabilistic delay)
  - Generalized Stochastic PNs (timed and immediate transitions)

- Add hierarchy
  - Place Chart Nets
PNs and SDF

IIR 2nd order filter
\[ o(n) = k_1 o(n-1) + k_2 i(n) \]

Homogeneous SDF network

Marked Graph

Fast Fourier Transform

SDF network

Weighted T-System
Switch/Select vs. choice/merge
PNs: No correlation between different choices
PNs and BDF

BDF network

Petri Net

PNs are not-determinate
PNs and BDF

**BDF network**

**Petri Net**

PNs are not-determinate
PNs and BDF

**BDF network**

**Petri Net**

PNs are not-determinate