Design representations

Control Oriented Models

Kris Kuchcinski
Krzysztof.Kuchcinski@cs.lth.se

Models

“A theory has only the alternative of being right or wrong. A model has a third possibility: it may be right, but irrelevant.”

Manfred Eigen (1927 - )

Control Flow Models

- FSM (Mealy and Moore)
- FSM extensions
  - Codesign FSM
  - Communicating FSM
- Petri nets
- StateCharts
- Discrete Event
- CCS, CSP, …
- …
Application Areas

- Reactive systems
- Control functions
- Protocols (telecom, computers, ...)
- ...

FSM basics

Different communication mechanisms:
- synchronous (classical FSM's, Moore, Mealy)
- asynchronous (CCS, Milner '80; CSP, Hoare '85)

Mealy and Moore state machines FSM = <S, I, O, δ, λ>

- S is set of states
- I is set of inputs (conditions)
- δ is a next-state function, δ: S x I → S
- λ is an output function, λ: S x I → O for Mealy machine
- λ: S → O for Moore machine

FSM Model for Elevator Controller

Mealy Machine
Moore vs. Mealy machines

- Theoretically, same computational power (almost)
- In practice, different characteristics
- Moore machines:
  - non-reactive (response delayed by 1 cycle)
  - easy to compose (always well-defined)
- Mealy machines:
  - reactive (0 response time)
  - hard to compose (problem with combinational cycles)

Problems with FSM’s

- How to reduce the size of the representation?
- Solution — hierarchical concurrent finite state machines
- Example — Harel’s StateCharts
- 3 orthogonal exponential reductions
  - hierarchy,
  - concurrency,
  - non-determinism.
Petri Nets

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

Petri Net example

Definition

A Petri net structure, C, is a four tuple

\[ C = (P, T, I, O) \]

- \( P = \{ p_1, p_2, ..., p_n \} \) is a finite set of places, \( n \geq 0 \);
- \( T = \{ t_1, t_2, ..., t_m \} \) is a finite set of transitions, \( m \geq 0 \);
- \( P \cap T = \emptyset \);
- \( I: T \rightarrow P^* \) is the input function, a mapping from transitions to bags of places;
- \( O: T \rightarrow P^* \) is the output function, a mapping from transitions to bags of places;
Petri Net Marking

A marking m of a Petri net C = (P, T, I, O) is a function from the set of places P to the non-negative integer : P → N

A marking represents an assignment of tokens to the places.

A marking m can also be defined as an n-vector, μ = (μ₁, μ₂, ..., μₙ), where n = |P| and μᵢ ∈ N, i = 1, 2, ..., n. The number of tokens in the place pᵢ is denoted by μᵢ.

A marked Petri net M = (C, μ) is a Petri net structure C = (P, T, I, O) and a marking μ.

Summary

A (C, μ₀) is a Petri Net Graph N

- places: represent distributed state by holding tokens
- marking (state) μ is an n-vector (μ₁, μ₂, μ₃, ...), where μᵢ is the non-negative number of tokens in place pᵢ.
- initial marking (μ₀) is initial state
- transitions: represent actions/events
  - enabled transition: enough tokens in predecessors
  - firing transition: modifies marking
- ...and an initial marking μ₀.
- Place/Transition ↔ conditions/events

Firing Rules

The execution of a Petri net is carried out by firing transitions, which moves tokens from places to places.

A transition tᵢ ∈ T in a marked Petri net C = (P, T, I, O) with marking m is enabled if for all pᵢ ∈ I(tᵢ)

μ(pᵢ) ≥ #(pᵢ, I(tᵢ))

A transition tᵢ in a marked Petri net with marking μ may fire whenever it is enabled.

Firing an enabled transition tᵢ results in a new marking μ' defined by

μ' = μ - #(pᵢ, I(tᵢ)) + #(pᵢ, O(tᵢ))

#(pᵢ, I(tᵢ)) is the number of occurrences of pᵢ in I(tᵢ) or (O(tᵢ))
An Example

Sequencing

Concurrency
Conflict

The firing order is not irrelevant.

Communication Protocol

Produces/Consumer
Properties of Petri Nets

Most important analysis problems for Petri nets:
- reachability and coverability
- liveness
- boundness
- safeness
- conservation

Reachability

Marking \( \mu \) is reachable from marking \( \mu_0 \) if there exists a sequence of firings \( s = \mu_0, t_1, \mu_1, t_2, \mu_2, \ldots, \mu \) that transforms \( \mu_0 \) to \( \mu \).

The reachability problem is decidable.

Liveness

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

- **Boundness**: 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

Analysis Techniques

- State Space Analysis techniques
  - Reachability Tree or Coverability Graph
- Structural analysis techniques
  - Incidence matrix
  - T- and S- Invariants
We will try to limit the tree to the finite size (notice, however, that it will usually result in the lost of information).

We introduce three types of nodes:
- frontier
- terminal
- duplicate
An extended marking

Let us assume that after sequence of transitions $\sigma$ we will end up in marking $\mu'$ from $\mu$
and $\mu' > \mu$ since transition firing can be repeated it can lead to $\mu''$
$$\mu'' = \mu' + (\mu'' - \mu) > 0$$

or after $n$ times we can produce a marking
$$\mu' + n(\mu'' - \mu)$$
which, in fact, produces infinite marking.

Infinite Marking

Infinite number of markings is represented by $\omega$ symbol with following properties:
- $\omega + a = \omega$
- $\omega - a = \omega$
- $\omega < \omega$
- $\omega \leq \omega$

Finite Reachability Tree
Analysis Techniques Based on Reachability Tree

- It can be used for analyzing several properties such as
  - safeness and boundness
  - conservation
  - coverability
- It cannot be used, in general, to solve problems such as
  - reachability
  - liveness
  - determine which firing sequences are possible
- Limitations => loss information by the use of $\omega$ symbol

Matrix Equations

- An alternative definition of Petri nets
  - Instead of defining $(P, T, I, O)$, we define $(P, T, D^-, D^+)$, where two matrices $D^-$ and $D^+$ represent input and output function.
  - We define
    \[
    D^-[i, j] = \text{#}(p_i, I(t_j)) \\
    D^+[i, j] = \text{#}(p_i, O(t_j))
    \]
  - The transition $t_j$ is represented by the unit $m$-vector $e[j]$.

Matrix Equations (cont’d)

- Transition $t_j$ is enabled in a marking $\mu$ if
  \[ \mu \geq e[j]D^- \]
- The result of firing transition $t_j$ in marking $\mu$, if it is enabled, is
  \[
  \delta(\mu, t_j) = \mu - e[j](D^- + e[j]D^+) = \mu + e[j](-D^- + D^+) = \mu + e[j]D
  \]
  where $D = D^- - D^+$. 


Matrix Equations - Example

\[ \begin{align*}
D & = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \\
D' & = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 2 & 0 & 1 \end{bmatrix} \\
D'' & = \begin{bmatrix} 2 & 1 \ -1 & 0 \\ 0 & 1 \ 0 & 0 \end{bmatrix} \\
D''' & = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}
\end{align*} \]

Examples

- firing \( t_3 \)
\[ \mu = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \]

- firing \( t_3 t_2 t_1 \)
\[ \mu = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 0 \end{bmatrix} \]

Examples (cont’d)

- To determine if the marking \((1, 8, 0, 1)\) is reachable from the marking \((1, 0, 1, 0)\), we have the equation
\[ \begin{align*}
[8 & 0 1] - [0 & 1 0] x = [0 & 2 & 1 & -1] \\
[0 & 0 & 0 & 1] x & = [0 & -1 & -1 & 0] \\
[0 & 8 & -1 & 1] x & = [0 & 2 & 1 & -1] \\
[0 & 0 & 0 & 1] x & = [0 & 0 & 0 & 1] 
\end{align*} \]

- which has a solution \(x = (0, 4, 5)\) and sequence the \( S = t_1 t_2 t_3 t_4 t_1 t_2 t_3 t_2 t_1 \).
Matrix Equations - Problems

- matrix D by itself does not properly reflect the structure of the Petri net - self loops disappear.
- lack of sequencing information in the firing vector.
- a solution of equation $\mu' = \mu + x \cdot D$ is necessary for reachability but it is not sufficient.

Summary of Petri Nets

- Graphical formalism
- Distributed state (including buffering)
- Concurrency, sequencing and choice made explicit
- Structural and behavioral properties
- Analysis techniques available

Petri Nets Extensions

- Add interpretation to tokens and transitions
  - Colored nets (tokens have value)
- Add time
  - time/timed Petri Nets (deterministic delay)
  - stochastic (duration, delay)
  - control (weak, strong)
- Stochastic PNs (probabilistic delay)
- Generalized Stochastic PNs (timed and immediate transitions)
- Add hierarchy
- Place Chart Nets
Time Petri Nets (TPN’s)

- \( a \) (0 ≤ \( a \)), is the minimal time that must elapse, starting from the time at which transition \( t_i \) is enabled, until this transition can fire,
- \( b \) (0 ≤ \( b \) ≤ \( \infty \)), denotes the maximal time during which transition \( t_i \) can be enabled without being fired.


TPN example

Some Properties of TPN’s

- The reachability and boundness problems for TPN’s are undecidable.
- There exist subclasses of TPN’s which are bound.
Literature