Petri Nets – Lecture 1

Chen Chen

Sep. 28th, 2011
Outline

• History of Petri Nets
• Basic Terminologies
• Comparing with FSM
• Petri Net Properties
Invention of Petri Nets


- **IEEE Computer Pioneer Award (2008)**

  “For establishing Petri net theory in 1962, which not only was cited by hundreds of thousands of scientific publications but also significantly advanced the fields of parallel and distributed computing”

Carl Adam Petri
1926 – 2010
Introduction to Petri Nets Theory

• A theory of systems for modeling concurrency and synchronization

• Feature
  ➢ Graphical representation
  ➢ Simplicity
  ➢ Expressiveness for concurrency and asynchronous operations
History of Petri Nets

- C.A. Petri (1962) - Invention of Petri Nets
- Ramchandani (1973) - Timed Petri Nets
- Hack (1972) - Free-choice Petri Nets
- Holt and Commoner (1970) - Graphical expression of PN Marked Graph
- Kurt Jensen (1981) - Colored Petri Nets
- Bail, Alla, David (1991) - Hybrid Petri Nets
- E. P. Dawis, J. F. Dawis, Wei-Pin Koo(2001) - Dualistic Petri Nets
- Buchs and Guelfi (1991) - Object Oriented Petri Net
- Haddad & Poitrenaud (2007) - Recursive Petri Nets
- ISO/IEC 15909 (2011) - Petri Nets Standard

- First PN
- Sub-class of PN
- Extension of PN
- Normalization of PN
Outline

- History
- **Basic Terminologies**
- Comparing with FSM
- Petri Net Properties
Example: Critical Section

- 3 users try to access the same CS
- Only one user can access CS each time

1 token = lock

3 tokens = 3 users

Critical Section

place

transition

arc

token
Example: Critical Section

- Multiple users try to access the same CS
- Only one user can access CS each time

One user entered CS
The others have to wait
Example: Critical Section

- Multiple users try to access the same CS
- Only one user can access CS each time

One user completes access.
Another one can enter now.
Example: Producer-Consumer

Producer produces tasks and put the tasks in the task buffer. Consumers take tasks from the task buffer and execute them. One task can only be executed by one consumer – either A or B.
Example: Producer-Consumer

Producer

Task Buffer

Consumer A
Consumer B

Producer produced one task.
Either A or B can be fired. But only one will be fired!
Example: Producer-Consumer

After firing A or B, the task is executed.
What’s the difference between Petri Nets and Dataflow?
Definition of a Petri Net (Self-reading)

• A Petri Net is a bipartite graph \((P, T, A)\) that comprises of:
  - A set of transitions: \(T\)
  - A set of places: \(P\)
  - A set of directed arcs: \(A\)

\[\text{a transition} \quad \text{a place}\]
Firing Rules (Self-reading)

- Enabling on transition $t$
  \[ \forall p_i \in P, \ # \ of \ tokens \ in \ p_i \geq \ # \ of \ arcs \ from \ p_i \ to \ t \]

- Firing on transition $t$
  
  - $t$ MAY fire whenever it is enabled.
  
  - $\forall a_i = (p,t) \in A$, a token is removed from $p$
  
  - $\forall a_i = (t,p) \in A$, a token is deposited to $p$
Petri Net Marking (Self-reading)

- Tokens
- Marking – PN states
- A formal definition of a marked PN
  - \( PN = (P, T, A, M) \), \( M \) is the marking
  - \( P = \{p_1, p_2, \ldots, p_n\} \)
  - \( T = \{t_1, t_2, \ldots, t_n\} \)
  - \( A = \text{a multiset of } (T \times P) \cup (P \times T) \)
  - \( M = \{m_1, m_2, \ldots, m_n\} \), where \( m_i = \# \text{ of tokens in place } p_i \)
Outline

• History
• Basic Terminologies
• Comparing with FSM
• Petri Net Properties
Example: Two’s Complement

- What is two’s complement?
- How to compute two's complement?
Example: Two's Complement (cont.)

- What is two's complement?
- How to compute two's complement?
  - flip all bits, and then plus a carry bit
  - Assume $B = 110$, then $-B = 001 + 1 = 010$
Example: Two's Complement (cont.)

- What is two's complement?
- How to compute two's complement?
  - flip all bits, and then plus a carry bit
  - Assume \( B = 110 \), then \( -B = 001 + 1 = 010 \)
  - Compute two's complement of 110 from low bit
    - \( \text{flip}(0) + \text{carry bit} \Rightarrow 1 + 1 = 0 \) & carry bit
    - \( \text{flip}(1) + \text{carry bit} \Rightarrow 0 + 1 = 1 \)
    - \( \text{flip}(1) \Rightarrow 0 \)
Example: Two's Complement (cont.)

State $q_1$: need to add carry bit
- $\text{flip}(0) + \text{carry bit} \rightarrow 1 + 1 = 0 + \text{carry bit}$
- $\text{flip}(1) + \text{carry bit} \rightarrow 0 + 1 = 1$

State $q_2$: no need to add carry bit
- $\text{flip}(0) \rightarrow 1$
- $\text{flip}(1) \rightarrow 0$

R means reset of computation
Example: FSM (cont.)

Now try to compute two's complement of 110
Example: FSM (cont.)

Input sequence: R110
Output sequence: —
Computing from low bit
R indicates reset

Two's complement of 110 is 010
Example: FSM (cont.)

Input sequence : R11
Output sequence: 0

Computing from low bit
R indicates reset

Two's complement of 110 is 010
Example: FSM (cont.)

Input sequence : R1
Output sequence: 10

Computing from low bit
R indicates reset

Two's complement of 110 is 010
Example: FSM (cont.)

Input sequence: R
Output sequence: 010

Computing from low bit
R indicates reset

Two's complement of 110 is 010
Example: FSM (cont.)

Input sequence: —
Output sequence: R010

Computing from low bit
R indicates reset

Two's complement of 110 is 010
Finite State Machine (FSM) Self-reading

- A FSM is a five tuple \((Q, \Sigma, \Delta, \delta, \Gamma)\) where
  - \(Q\) : a finite set of states \(\{ q_1, q_2, \ldots \}\)
  - \(\Sigma\) : a finite input alphabet
  - \(\Delta\) : a finite output alphabet
  - \(\delta : Q \times \Sigma \rightarrow Q\) is the next state function, mapping the current state and current input into the next state
  - \(\Gamma : Q \times \Sigma \rightarrow \Delta\) is the output function, mapping the current state and input into the output symbol.
Important Features of FSM

What are the features?
Important Features of FSM

What are the features?

FSM is always at a single active state!

A single input event triggers state transition!
Important Features of FSM

What are the features?

FSM is always at a single active state!
How to represent multiple active states as a single active state?
A single input event triggers state transition!
How to represent synchronization between multiple states?
Another Example: 2-Stage Pipeline Modeling

Input sequence → Unit1 → Unit2 → Output sequence

Each unit can be modeled by a three-state FSM
Another Example: 2-Stage Pipeline Modeling (cont.)

How to model the two units together in FSM?
Another Example: 2-Stage Pipeline Modeling (cont.)

How to model the two units together in FSM?

– cross-functional states

State aa’: Unit1 is ready for input & Unit2 is ready for input
State ba’: Unit1 is busy & Unit2 is ready for input
State ca’: Unit1 is ready to output & Unit2 is ready for input

... State cc’: Unit1 is ready to output & Unit2 is ready to output
The structure of the FSM for 2-stage pipeline modeling. Inputs and outputs are omitted.
Problems of FSM Modeling of the Pipeline

- Number of states grows exponentially with the number of stages
- The identity of the individual stage lost
- The structure information is obscured
- Concurrency / synchronization information cannot be represented
Try Again: 2-Stage Pipeline Modeling by PN

Each unit can be modeled by 3 places and 3 transitions
Try Again: 2-Stage Pipeline Modeling by PN (cont.)

Input sequence → Unit1 → Output sequence

Done with processing

Ready for input → Busy → Ready to output → Pickup input data

Unit1 transfers intermediate result to Unit2

Ready for input → Busy → Ready to output → Output result

Pipeline

PN
Scenario: D1 in the entrance of Unit 1
D2 in the entrance of Unit 1
Try Again: 2-Stage Pipeline Modeling by PN (cont.)

**Scenario:** D1 being processed by Unit1
D2 in the entrance of Unit1
Try Again: 2-Stage Pipeline Modeling by PN (cont.)

Scenario: D1 is done by Unit1
D2 in the entrance of Unit1
Scenario: D1 being processed by Unit2
D2 in the entrance of Unit1
Try Again: 2-Stage Pipeline Modeling by PN (cont.)

Scenario: D1 is done by Unit2
D2 being processed by Unit1
Scenario: Result of D1 is outputted by Unit2
D2 is done by Unit1
Try Again: 2-Stage Pipeline Modeling by PN (cont.)

Scenario: Result of D1 is outputted by Unit2
D2 being processed by Unit2
Scenario: Result of D1 is outputted by Unit2
D2 is done by Unit2
Try Again: 2-Stage Pipeline Modeling by PN (cont.)

Scenario: Result of D1 is outputted by Unit2
Result of D2 is outputted by Unit2
Conclusion: Compare PN with FSM

• PN is powerful in constructing composite machines. FSM must do cross product.
• PN is powerful in expressing concurrency. (FSM cannot express concurrency).
• PN is powerful in expressing synchronization.
Outline

• History
• Basic Terminologies
• Comparing with FSM
• Petri Net Properties
Petri Net Properties

- Reachability
- Safeness and Boundness
- Conservation
- Liveness
- Persistency
- Consistency