Petri nets: Process modeling

Rushikesh K Joshi
CSE
IIT Bombay
Basic Petri-net Elements

- **Places**
  - Represent states

- **Transitions**
  - Represent actions

- **Token**
  - Current State

- **Arrows**
  - Flow elements
Elementary Petri-Nets
- at most one token per place
Transitions

- **Enabled** if all its preplaces has a token
- All post places are empty
  - No token in them
Transitions

- Enabled transition **fires**, and it moves the token(s) downstream
- Into all post places
Places

- A token may enable one or more transitions
- It gets consumed by firing one of the enabled transitions
Firing Sequence = sequence of transition firings (not markings)

- Following firing sequences are possible (considering one transition firing at a time)
  - t1 --> t3
  - t2 --> t4 --> t5
  - t2 --> t5 --> t4
State Space
= set of all possible markings and the transitions through them

The Net

The State Space
The same Petri-net for different purposes with different initial markings

Either XOR Fork/XOR Join ({P1} as initial)
Or AND Fork/Multi-join

In Multi-join, multiple tokens get collected one after the other ({P2} as initial)
Reachability Graph
= state space given initial marking

Reachability Graph
for initial marking as \{P1, P2, P3\}
Infinitely enabled net

Reachability graph
With initial marking=UNDEF

P0 has inf tokens
The number written inside represents those many (initial) tokens in the given state of the machine.

Capacity of P1 is 1,
capacity of P0,
Capacity of P2 is inf
Inf is default capacity in classical nets
A Problem: Rules of one masters program

- It's 2 semester program
- At most 10 courses
- Minimum 8 courses
- Per semester max 5 courses
- One R&D project can be taken as one course
- One masters project can be taken in place of 3 courses.
- MTP 1 in sem 1: counted as 3 courses, MTP 2 in sem 2: counted as 2 courses
- 1 seminar in 1st sem is must
- Backlogs of sem1 can be taken in sem2
- If any backlogs remain, one more semester is granted
Simplified Version of the Problem

• 5 courses
• 2 semester
• 2 courses per semester: total exactly 4 courses
• A course cannot be taken twice

• Make your net to represent this system
• Try to reduce the no. of transitions
• Model with: One transition representing one course
Vending machine Problem

- Insert Rs. 25: currency Rs. 5, Rs. 10 coins accepted e.g. 5+5+5+5+5, 10+5+10 etc. (design as small a machine as possible)
- Choose the drink
- Pick it up
- The machine is ready for the next task
- Time out of 1 minute, the machine returns all the coins inserted and resets to initial state
- Invalid coin is rejected, and all the coins are returned, the machine resets with a spoken message of invalid coin.
Workflow Nets
Workflow Nets

- Unique source place
- Unique sink place
- Connected
- Unique initial marking, unique terminal marking
- Well-formed – every transition is reachable, every marking is reachable, every marking terminates
Workflow Patterns: SEQUENCE

Sequence

T1  T2  T3
Workflow Patterns: XOR

Diagram showing a workflow pattern with nodes labeled T1, T2, T3, T4, T5, and T6, connected by arrows indicating the flow of the workflow.
Workflow Patterns: 2/3 Choice
Workflow Patterns: AND
Workflow Patterns: Iteration

which one is correct? Which one is incorrect? Why?
Workflow Patterns: Multi-merge

Tokens can arrive both ways, and they are all sent down
Split/Join

- Parallel split

- Synchronization (parallel merge/AND join)
Split/Join

- Parallel split

- Synchronization (parallel merge/AND join)
Roles as tokens in places

- P1 = Ready to register
- P2 = not approved
- P3 = permitted to Attend classes
- P4 = state of opinion
- P5 = approved
- P6 = Process Terminated
- P7 = facad is Ready To approve

Steps:
1. Submit registration
2. Get it approved
3. Attend classes
4. Submit adjustment
5. Timeout activity
Types of activities

- Automatic Activity
  - Computer can execute it fully
    - (when enabled, it is executed automatically such as by an algorithm, script task etc.)
- User Activity
  - A human being executes it
    - (though enable, it is done manually)
- Message Activity
  - An external message triggers the task instance
    - (though enabled, it requires a message to trigger it)
- Time Triggered Activity
  - Task needs to be trigged at a particular time, or after a certain period of timeout
    - (though enabled, time has to trigger it)
Types of Activities
An Example

- **Submit registration**
- **Get it approved**
- **Attend classes**
- **P1=Ready to register**
- **P2=not approved**
- **P3=permitted to Attend classes**
- **P4=state of opinion**
- **P5=approved**
- **P6=Process Terminated**
- **P7=facad is Ready To approve**

**User activity**
- **Submit adjustment**
- **Attend classes**
- **Get it approved**
- **P1=Ready to register**

**Time based activity**
- **Timeout activity**
Types of Activities

An Example

- **P1 = Ready to register**
- **P2 = not approved**
- **P3 = permitted to Attend classes**
- **P4 = state of opinion**
- **P5 = approved**
- **P6 = Process Terminated**
- **P7 = facad is Ready To approve**

Activities:

- **Fill up form**
- **Evaluate Registration form**
- **Submit registration**
- **Get it approved**
- **Submit adjustment**
- **Attend classes**
- **Evaluate Registration form**
- **Form Filled**
- **Form Evaluated**

States:

- **Submit adjustment**
- **Form Evaluated**
- **Submit registration**
- **Form Filled**
- **Get it approved**
- **Attend classes**
- **Evaluate Registration form**
- **Fill up form**
- **P1 = Ready to register**
- **P2 = not approved**
- **P3 = permitted to Attend classes**
- **P4 = state of opinion**
- **P5 = approved**
- **P6 = Process Terminated**
- **P7 = facad is Ready To approve**
Problem

- Re-engineer the above net to reflect the process that we are actually following
- Make sure all traces that we actually take are in
- Also make sure the traces that we donot take are not in
- i.e. the aim is to get the exact model that is necessary and sufficient to represent the present process.
Classical Petri Nets

- A place can contain 0 or more tokens (more than 1)
  - The state then needs to mention the count of tokens held in places
    - e.g. \{1 p1, 2 p3, 4 p4\}
    - This state is different from \{2 P1, 2 P3, 4 p4\}
Places with multiple tokens

Build a state space

Initial marking: \{P1\}