Formal Process Modelling

Petri Net
Behaviour Net Model
Event-driven Process Chains
Formalisation

Håvard D. Jørgensen
Materiale fra:
Jon Atle Gulla, NTNU
Folker den Braber, SINTEF
Anders Moen, Norsk Regnesentral
Outline

- Petri nets
  - Simple formal process models
  - Execution semantics
- Behaviour Net Model
  - Petri nets with data models
- Formalization of ARIS process model (EPC) with Petri nets
  - Mapping from EPC to Petri nets
  - Verification of EPCs
Petri Nets - Basics

- Build up from places and transitions
- Arrows between them (never from place→place or from transition→transition)
Petri nets is a tuple (P, T, F):
- P is a finite set of places
- T is a finite set of transitions (P \cap T = \emptyset)
- F \subseteq (P \cap T) \cup (T \cap P) is a set of arcs (flow relations)

Notation:
- \( t_3 = \{p_2, p_3\} \) Input places to transition
- \( p_2^\bullet = \{t_2, t_3\} \) Output transitions form place

Place p1 is an input place for transition t1
Place p2 is an output place for transition t1

Petri nets and formalization:
- Petri nets have a mathematical foundation
- Formal semantics
- Automatic model analysis
Petri Nets - Dynamics

- Start configuration (initial marking)
- A transition $t$ can fire if all input places of $t$ contain tokens and all output places of $t$ are empty

Initial marking
Petri Nets - Dynamics

**Tokens:**
- A place contains zero or more tokens

**Firing rule:**
- A transition $t$ is said to be enabled iff each input place $p$ of $t$ contains at least one token
- An enabled transition may fire. If transition $t$ fires, then $t$ consumes one token from each input place $p$ of $t$ and produces one token in each output place $p$ of $t$
- Alternatively: must fire immediately

**State $M$ (Marking):**
- Distribution of tokens over places
- $M: p ? \mid 0$

**Example:** State $M = p1 + 2p3$
- $t1$ is enabled, $t3$ is not enabled
Example: Producer - Consumer

Initial Configuration

- **Producer 1 (p1)**
  - Producing
  - Filling

- **Producer 2 (p2)**
  - Filling

- **Consumer 1 (c1)**
  - Emptying

- **Consumer 2 (c2)**
  - Consuming

- Buffer
Mutual Exclusion

Initial Configuration
Petri Nets - Expressiveness

- Choice (XOR)
  - Branch: Place with N output transitions
  - Merge: Place with N input transitions

- Concurrency, Synchronisation
  - Branch: Transition with N output places
  - Synchronise: Transition with N places

- Later: Simplification
Petri Nets - Applications:

- Modelling
- Analysis
- Execution
- Simulation
- Visualisation
- Monitoring

- Concurrent distributed processes
  - Workflow
  - Communication systems
  - Operating systems
Petri Nets - Variations

- Time
  - Delayed firing
- Stochastic
  - Branching and time probabilities
- Coloured
  - Different data types
- Modular
  - Decomposition and abstraction
- Object-oriented
  - Inheritance, encapsulation
- etc.
Example: Oslo Subways

- **Oslo subways**
  - 5 lines
  - 101 stations
  - Trains run every 15 minutes
  - 69.8 million passengers
  - 10624 passengers/hour in average
  - 118.7 km track

- **Modelled as OO, timed Petri net**
  - Tokens: Trains, passengers
  - Subnets: Stations, line segments, signalling systems
  - 2926 places, 2217 transitions, 6971 arcs

Kristoffersen, Anders Moen, Hallstein Asheim Hansen: *the Oslo Subway by hierarchic, coloured, object-oriented, timed Petri Nets with viewpoints*
Example: Subway Line segment

- Line segment are places
- Trains = tokens
- Move-transitions

Safe Petri net
- Never more than one token in each place (no collisions)
User Oriented Visualisation
Petri Nets - Weaknesses

- Closed system assumption
- No model change during execution
- Uncertainty (e.g. OR-branching)
- Large, complex models
- Poor support for abstraction, decomposition
- Theoretical constructs
- Map activity to place or transition
- Just control flow
BNM: Behaviour Net Model

- Petri nets + ER
- Places have types from data model
  - Instances are tokens
  - Classes are tokens
- Tokens are consumed or referenced when transition fires
- Transitions have pre- and post-conditions for firing
- Simulation in IS development
  - Validate model, find bottleneck
  - Path analysis, matrices
BNM Examples

- Fig 14.24 Payroll computation
- Fig 14.30 Path analysis
- Fig 14.25 Order management
ARIS Conceptual Model

- Multi-view conceptual language for modeling business processes and organizational structures
  - Organizational view
  - Function view
  - Data view
  - Control view
  - (Output view)

- Pre-defined models for ERP functionality
  - Low-level processes correspond to specific transactions in ERP system
  - Organizational structures correspond to structural elements in ERP system
ARIS Control view (EPC)

- Event (or state)
- Function (process)
- Logical connector (AND, OR, XOR)

Data elements and elements from the organizational view may also be added.
Formalization with Petri net

- Event-driven Process Chains (EPC)
  - Process model (control view) of ARIS
  - Used to describe business processes

- Formalization of EPC syntax
- Mapping EPCs onto Petri nets
  - Provides a formalization of EPC
  - Defines the semantics of EPC
  - Enables checks for consistency, completeness, soundness, etc.

Do not take into consideration links to elements or organizational elements
The Structure of EPC Model

- **Function**: check availability
  - **Logical connector** (AND, XOR, OR)
    - **Event**: articles available
      - **Function**: XOR
        - **Event**: articles need to be produced
          - **Function**: XOR
            - **Function**: ship order
              - **Function**: purchase material
              - **Function**: make production plan
Formalization of EPC Syntax

EPC = (E, F, C, T, A):
- E is a finite set of events
- F is a finite set of functions
- C is a finite set of logical connectors
- T : C \rightarrow \{AND, XOR, OR\} is a function which maps each connector onto a connector type
- A : (E \cup F) \cup (F \cup E) \cup (E \cup C) \cup (C \cup E) \cup (F \cup C) \cup (C \cup F) \cup (C) is a set of arcs

Syntactic properties:
- The sets E, F, and C are pairwise disjoint
- For each e \in E: |\•e| \leq 1 and |e\•| \leq 1
- There is at least one start event and one final event
- For each f \in F: |\•f| = 1 and |f\•| = 1
- For each c \in C: |\•c| \geq 1 and |c\•| \geq 1
- The graph induced by EPC is weakly connected
- C_J and C_S partition C, i.e. C_J \cup C_S = C and C_J \cap C_S = \emptyset
- C_EF and C_FE partition C, i.e. C_EF \cup C_FE = C and C_EF \cap C_FE = C
Formalization of EPC Semantics

- Defining EPC semantics in terms of Petri nets

- Basic mapping of EPC to Petri nets:

- Mapping of connectors more complex
  - Depends on type of connector
  - Depends on events and/or functions linked to connector
Mapping of connectors

f_i: function
e_i: event
Mapping from EPC to Petri net
Mapping from EPC to Petri net
An Erroneous EPC

Model not sound:

- Assume that *no billing needed* event holds.
- *Produce article* will never be fired
Verification of EPC Models

- **Regular:**
  An EPC is regular iff
  - EPC has two special events: estart and efinal
  - Every node \( n \in N \) is on a path from \( e_{\text{start}} \) to \( e_{\text{final}} \).

- **Sound:**
  A regular EPC is sound iff
  - For every state \( M \) reachable from the initial state, there exists a firing sequence leading from state \( M \) to the final state.
  - The final state is the only state reachable from the initial state where event \( e_{\text{final}} \) holds.
  - There are no dead functions.

- **Well-structured**
  An EPC is well-structured iff for any pair of connectors \( c_1, c_2 \) such that one of the nodes is is \( C_{\text{AND}} \) and the other in \( C_{\text{XOR}} \) and for any pair of elementary paths \( p_1 \) and \( p_2 \) leading from \( c_1 \) to \( c_2 \), \( ?(p_1) \cap ?(p_2) = \{c_1, c_2\} \) and \( p_1 = p_2 \).
Erroneous EPCs

Two models to the right are not well-structured
Erroneous?

Does not mean that it is not useful

Proper functioning cannot be guaranteed
OR-Connectors

- Represent uncertainty (1 or more)
- Exponential complexity growth in Petri nets ($2^n - 1$)
Conclusions

- ARIS rooted in informal languages
- Semantic analysis of ARIS
  - Petri nets
    - Formalization of EPC semantics by mapping EPC diagrams to Petri nets
  - BWW ontological model
- Evaluation of modeling language with respect to the deep structure of information systems
- Strengths and weaknesses of formal languages
  - Remember: AND not XOR