Formal Requirements Specification

By
SATYANANDARAM N

Agenda

- Formal technique
- Formal specification language
- Model-oriented and Property-oriented
- Operational semantics
- Merits of Formal requirements specifications
- Limitations
- Axiomatic specifications

Introduction

- Formal languages are easiest way to write specifications those are precise, clear and concise
- We say that a system is correctly implemented when it satisfy its specification

Formal Technique

- Is a mathematical method to specify a system
- It verify whether a specification is achievable or not
- It verifies that an implementation satisfies its specification
- We have to prove the properties of system without necessarily running it

Formal Specification Language

- Anyone who can learn to read and write programs can learn to read and write formal specifications
  - Programming languages are formal languages
- It consists of two sets SYN and SEM and a relation SAT between them
  - Syntactic domain
  - Semantic domain
  - Satisfaction relation (relation of SYN and SEM)

Syntactic Domains

- The syntactic domain of a formal specification language consists of:
  - an alphabet of symbols
  - a set of formation rules to construct well-formed formulas.
  - The well-formed formulas are used to specify a system.

Semantic domains

- It indicates how language represents the system requirements
- Developers have to specify algorithms that transforms input to output
- Syntax and semantics can be developed to specify states and state transition, events and their effects on state transition

Satisfaction relation

- Given the model of a system, it is important to determine:
  - whether an element of the semantic domain satisfies the specifications.

Model Vs. Property Oriented approaches

- Formal methods are classified into two categories
- Model-oriented style:
  - Constructing a model of the system in terms of mathematical structures such as tuples, relations, functions, sets, sequences etc…
  - Suitable for later phases of life cycle
  - Minor changes to a specification may leads to drastic changes to the entire specification

Model Vs. Property Oriented approaches

- Property-oriented approach:
- The system's behaviour is defined indirectly:
  - by stating its properties:
    - usually in the form of a set of axioms(logics).
    - Examples: logic-based, algebraic specification, etc.
  - Property-oriented approaches are suitable for requirements specification
- Property-oriented approaches specify a system:
  - as a conjunction of axioms,
  - make it easier to alter/increase specifications at a later stage

Example: producer/ consumer system

- List the properties of the system:
  - the consumer can start consuming only after the producer has produced an item,
  - the producer starts to produce an item only after the consumer has consumed the last item.

Example: producer/ consumer system

- In a model-oriented approach:
  - Define the basic operations, p (produce) and c (consume).
  - Then state that
    - S ==>S1+p, (if S, then s1+p)
    - S1 ==> S+c. (if S1, then S+c)
- In a Property-oriented approach:
  - Producing ==> No items exist for consumption
  - consuming ==> Item exists for consumption

Operational Semantics

- Operational semantics of a formal method:
  - the way it represents computations:
    - i.e. the exact sequence in which the different computations are carried out
  - according to what is meant by a single run of the system
  - how the runs are grouped together to describe the behaviour of the system

Linear semantics

- A run of a system:
  - described by a sequence (possibly infinite) of events or states.
  - The concurrent activities representation cannot possible
- A concurrent activity a parallel b
  - represented by the set of sequential activities a;b and b;a.
- This is simple:
  - but rather unnatural representation of concurrency.

Branching semantics

- The behaviour of a system can be represented by a directed graph:
  - Nodes of the graph represent the possible states in the evolution of a system.

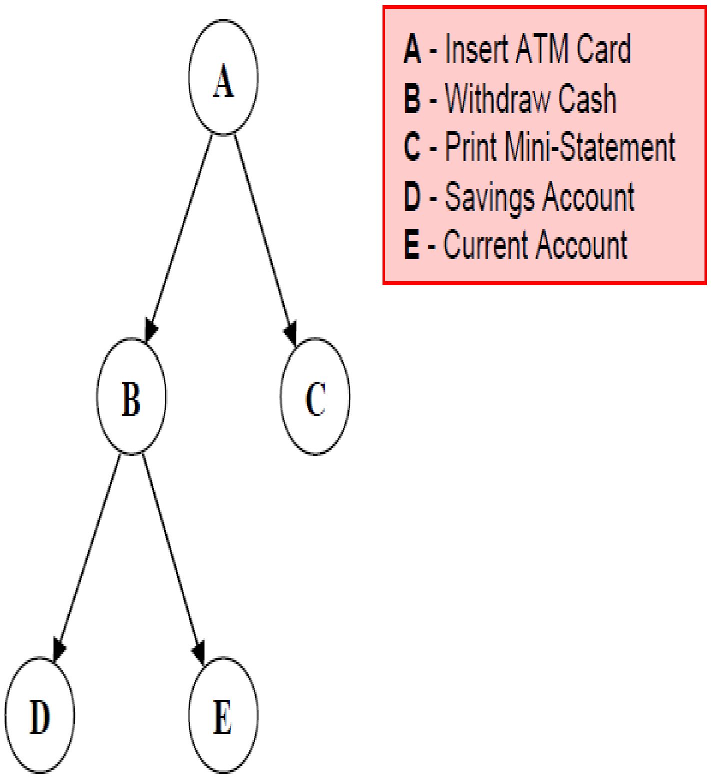


Fig. 3.7: Branching semantics

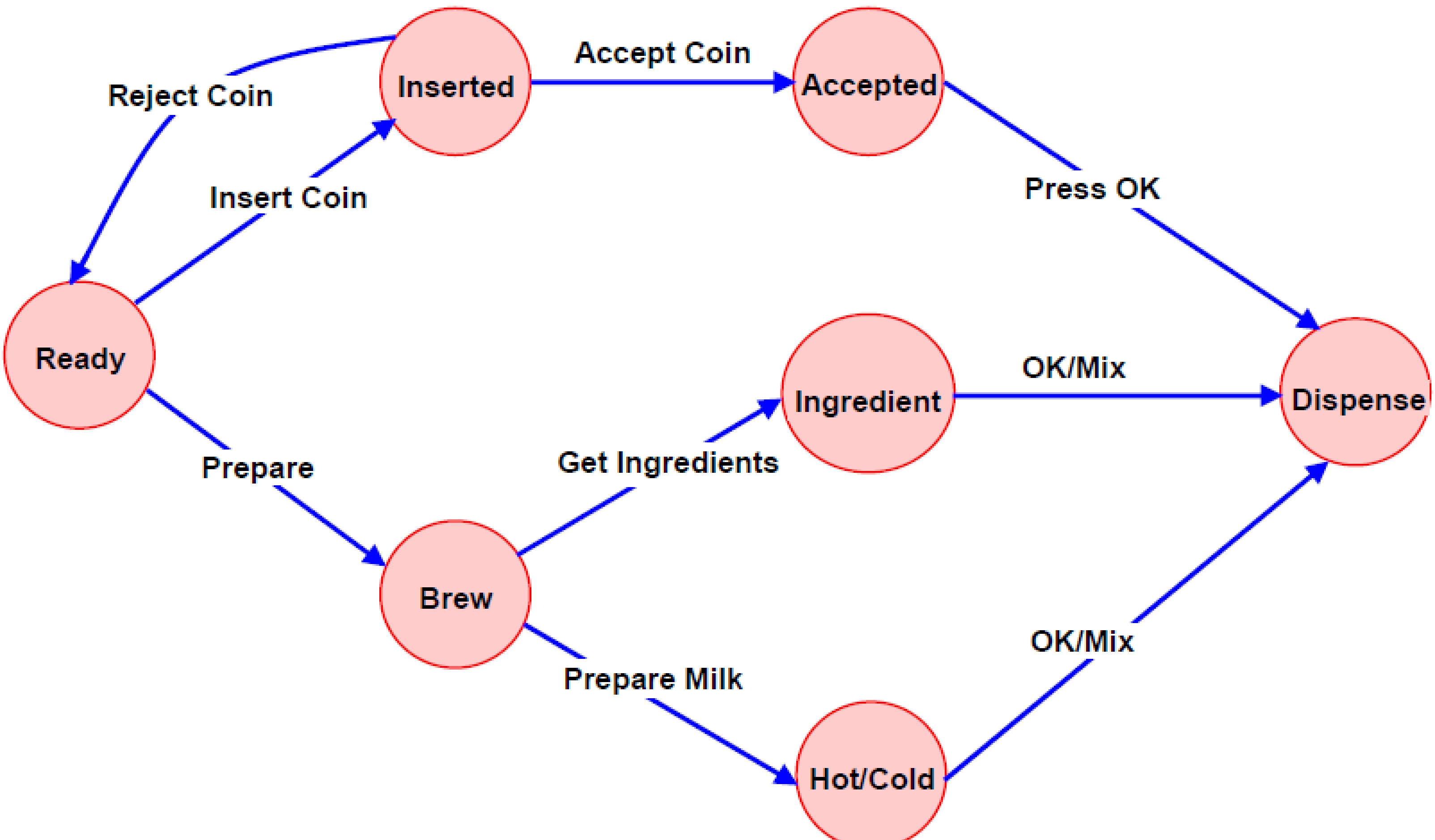
Maximally parallel semantics

- All concurrent actions enabled at any state
  - are assumed to be taken together.
- This is also not a natural model of concurrency.

Partial order semantics

- The partial order represents a precedence ordering among events:
  - It holds some events to occur only after some other events have occurred;
  - while the occurrence of other events is considered to be incomparable.

Partial order semantics



Merits of FRS

- It helps the exact formulation of specifications
- It encourages the quality specifications
- Construction of a quality specification clarifies several aspects of system behaviour
- Because of well-defined semantics:
  - Ambiguity in specifications is automatically avoided
- For large and complex systems like real-time systems:
  - 80% of project costs and most of cost overruns result from iterative changes required due to improper requirements specification

Limitations of FRS

- it is impossible to check absolute correctness of systems using theorem proving techniques
- Formal techniques are not able to handle complex problems
- a large unstructured set of mathematical formulas are difficult to understand

Axiomatic Specifications

- write pre-conditions and post-conditions to specify operations
- Pre-conditions:
  - What are the requirements on the parameters of the function?
- Post-conditions:
  - What are the requirements when the function is completed?

Example: Axiomatic Specification

- Function search(X: Integer Array; key: Integer): Integer
- pre: exists i in X'First ...X'last: x(i)=key
- post: X"(search(X,key))=key and X=X"
- Error: search(X,key)=X'last+1