Protocol Technology

Protocol Engineering

Gusztáv Adamis
BME TMIT
2016

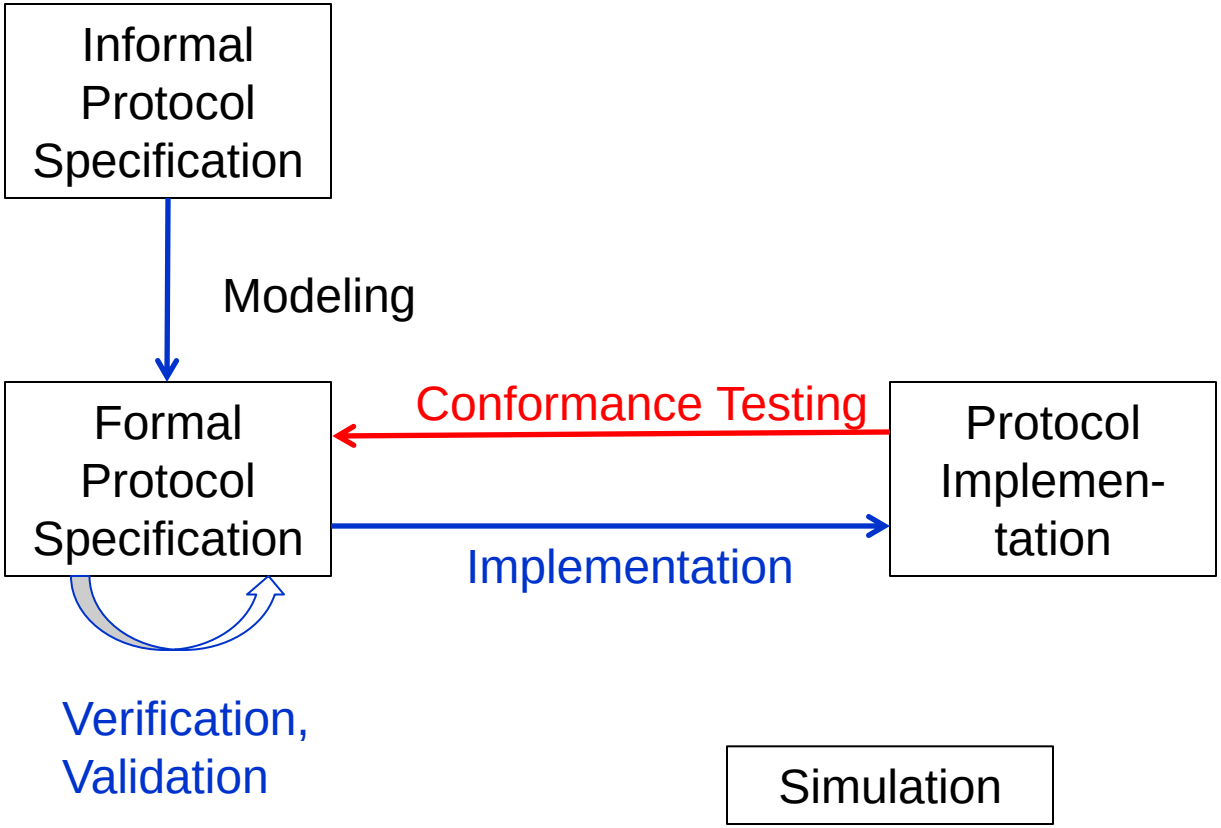
Protocols

- Protocols
  - Controlling Communication
- Static part
  - Messages
- Dynamic part
  - Message flow

Protocol terminology

- SAP – Service Access Point
- PDU – Protocol Data Unit
  - messages between peer entities
- ASP – Abstract Service Primitive
  - messages between different protocol layers

Protocol Engineering



Specification

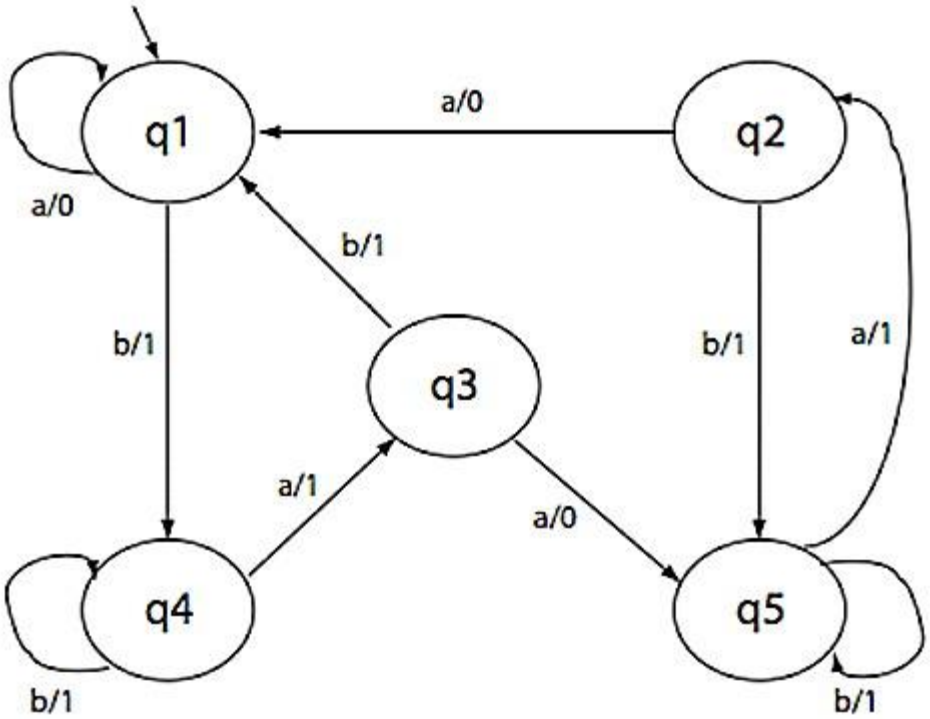
- Informal
  - Human language
  - Tables
  - Arrow sequences
- Shall be:
  - Unambiguous
    - Everyone shall understand the same
  - Complete
    - Rules for every possible situation
  - Able to check automatically
  - Able to implement
- ➔ Formal Description Techniques (FDT)

FDT

- Specification Languages, Notations
  - well-defined syntax
  - well-defined semantics
- Typical models
  - Finite State Machines (FSM)
  - Extended Finite State Machines (EFSM)
  - Communicating FSM/EFSM
  - Graph models (e.g. Petri net)
  - Algebraic models (e.g Calculus of Communicating Systems – CCS)

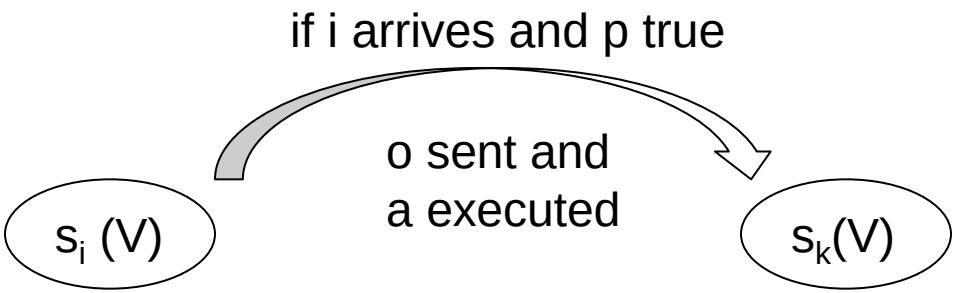
FSM

- FSM = S, I, O, s_0, f(s,I), where
  - S: finite set of states
  - I, O: finite set of input/output messages
  - $s_0 \in S$: start state
  - f(s,i): transition function: if the FSM is in state $s \in S$, recevies an $i \in I$ message then which $o \in O$ message is sent and which $s' \in S$ will be the next state



EFSM

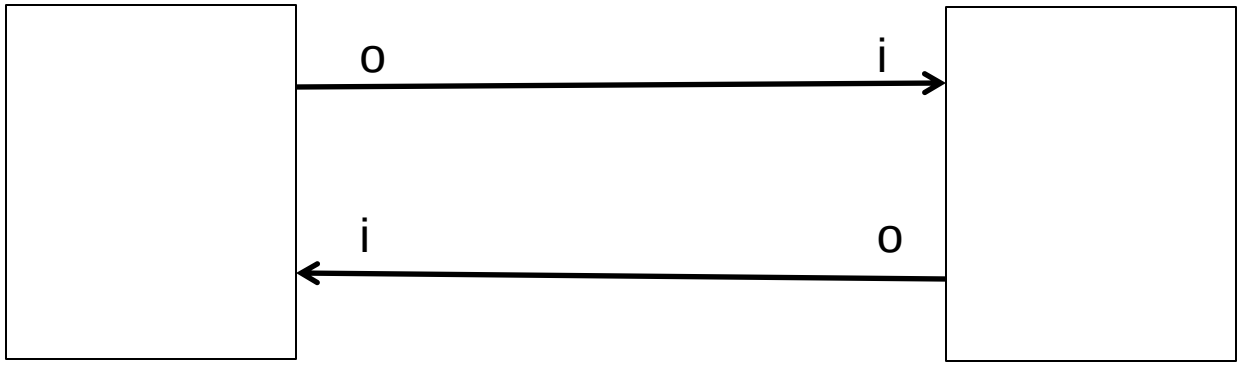
- EFSM = S, I, O, V, P, A, s_0, f(s,i,p), where
  - S: finite set of states
  - I, O: finite set of input/output messages
  - V: finite set of variables
  - P: finite set of predicates (conditions)
  - A: finite set of actions (e.g. value assignment)
  - $s_0 \in S$: start state
  - f(s,i,p): transition function: if the EFSM is in state $s \in S$, recevies an $i \in I$ message and predicate (condition) $p \in P$ satisfies, then which $a \in A$ action is executed, which $o \in O$ message is sent and which $s' \in S$ will be the next state



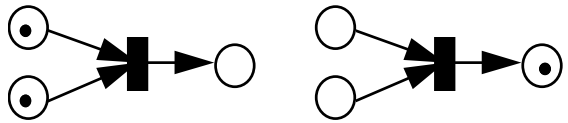
Communicating EFSM

- Lot of description techniques
  - UML state charts
  - SDL

- Communicating EFSMS
  - output of an EFSM is input of another



Net models



- Two node types
  - Condition
  - Execute
- Tokens
  - If all condition ha token -> execution fires
- Good for describing parallelism
- Good for correctness checking
  - validation

Modified Net models

- More tokens in a condition
- Coloured tokens
- Numerical nets
  - tokens have value
  - execution states have memory
  - actions, computations
- Timed nets

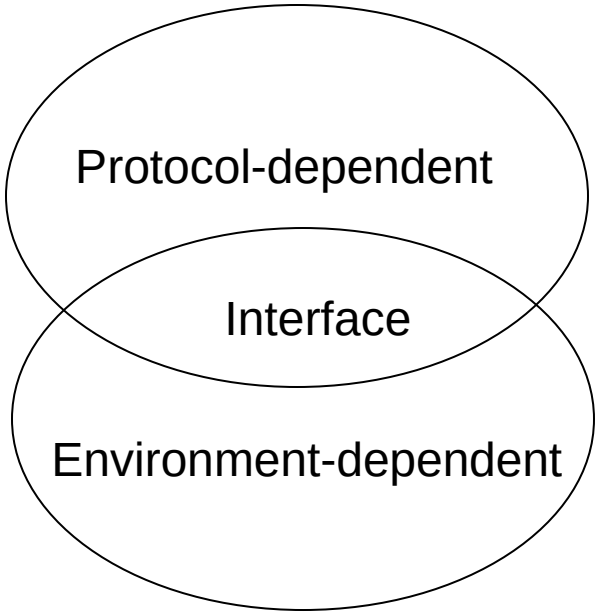
CCS

- Describes only observable events at interfaces
  - no information about how to implement
  - PCO: Point of Control and Observation
  - !: message sending
  - ?: waiting for a message
- Good for Conformance Testing
  - TTCN-3

Data Description

- Structure only
  - ASN.1
- Abstract Data Types
  - ACT ONE – in SDL
  - Data structures + operations (~class)

Protocol Implementation



- Manual
- (Semi-) automatic

- Env.-dep. part:
  - memory handling
  - buffer handling
  - real-time support (timers)
  - scheduling
  - communication between units
  - event processing
  - error handling

Semi-automatic implementation