COMMUNICATION PROTOCOLS

CHAPTER 8

Protocol Synthesis
Protocol Synthesis

- Protocol Synthesis is very important technique for efficiently designing of correct protocol specification.
- Protocol Synthesis completes protocol specification so that interaction among various software modules in service specifications satisfy the properties required for the correctness.
- Consider the connection management protocol. There are 2 processes a and b that share access to a full-duplex data link, indicated with 2 arrows.
- Processes a and b should coordinate the beginning and ending of data transfers across the link.
- Designer is asked to supply 2 process specifications for a well defined set.

Connection management protocol Interface
Protocol Synthesis Problem

• Protocol specifications will be synthesized which consist of following parameters:

  – Protocol entities.

  – Interactions among protocol entities.

  – Transitions and states in a protocol entity.

  – Internal events.

  – Tasks or service execution in an entity.

  – Interfaces.
**Synthesis Methods**

- Service oriented and non-service oriented
- General features:
  - The starting point of method
  - The modeling formalism used to describe the protocol
  - The constraints on communication model
  - Mode of interaction with the designer (automatic or interactive)
  - Protocol properties guaranteed by the methods
  - Specific protocol functions in the design
  - Complexity of the method

**Mode of interaction with designer is classified as:**

automatic and interactive
Interactive Synthesis Algorithm

- Components of the synthesis algorithm:
  - Set of production rules:
    - define the rules for producing global states, arcs and transitions between global states
  - Set of deadlock avoidance rules:
    - checks for non-occurrence of following conditions:
      - the two process states are both receiving states, or one is a receiving state and the other is a final state,
      - the two channels are empty.
  - Construction of the global state transitions graph.
Steps in Synthesis

- Take the inputs from the designer about the message interactions, processes behavior in an interactive way.
- Construct a global state transition graph by using the production rules and deadlock avoidance rules.
- Decompose the global state transition graph into processes (protocol entities).
- Perform validation of global state transition graph by using perturbation technique to check for design errors.
- The processes are given as output of the algorithm.
Automatic Synthesis Algorithm

- Project the service specification onto each SAP to obtain the projected service specifications.

- Apply the transition synthesis rules to each transition in the projected service specifications to obtain protocol entity specifications.

- Following rules apply for transitions labeled by SP whereas the rest rules are applied for the transitions indicated by
  
  - It implies that flow of control need not be transferred to another protocol entity
  
  - Synchronization messages must be sent to all other protocol entities to synchronize at the initial state in each of the respective protocol entities
  
  - Service Primitive (SP) is originating from the protocol entity. SP is a result of a reset protocol message, and therefore there is no need to transmit any other protocol message.
Automatic Synthesis Algorithm

- SP is originating from the service user at SAPi. Following the occurrence of this SP, synchronization message should transfer the flow of control to other corresponding protocol entities.

- Protocol entity specifications (PE-SPECs) must not expect any message at this point, since according to the service specification, no service action is expected at SAPi.

• Reception transitions in PE-SPECs are synthesized and correspond to the rules 2 and 4.

• Apply atomicity and optimization rules to PE-SPECs. Here some transitions and states are eliminated and enforces the atomicity of the action and reaction of the protocol entity.

• Remove the redundant cycles and transitions to obtain reduced and equivalent FSMs.
Automatic Synthesis of SDL from MSC

Elements of a basic MSC diagram
Automatic Synthesis of SDL from MSC

Elements of a HMSC Chart
Synthesis of Executable SDL Specifications from MSC

- *event automata*

- 3 categories of MSC events: input, active and idle events.

- idle event is a trivial (empty) event

- input events require synchronization with other instances, decision about event is taken by another instance:
  - input of message m by instance i: in( i,m )
  - create instance i by instance j: create(i,j)
  - timeout of timer t: timeout(t)

- active events do not require synchronization with other instances, decision is local to the current instance:
  - output of message m by instance i: out( i,m );
  - action a: action( a );
  - set timer t for duration d: set( t,d );
  - reset timer t: reset( t );
  - stop action: stop;
  - local condition over variable v with condition c: check( v, c );
Synthesis of Executable SDL Specifications from MSC

- The following algorithm can be used to construct MSC slices:
  - initial states of the event automaton correspond to symbols at HMSC graph with idle events;
  - for each basic MSC a (sub)sequence of states is created, corresponding to the sequence of events involving the instance $i$;
  - each MSC reference is replaced by the corresponding (sub)sequence of states;
  - the start symbol of the event automaton corresponds to the HMSC start symbol;

- Synthesis algorithm explained consists of the following steps:
  - Integrate HMSC model,
  - Construct MSC slices,
  - Make MSC slices deterministic,
  - Minimize MSC slices,
  - Generate SDL behavior,
  - Generate SDL structure.
Synthesis of Executable SDL Specifications from MSC

Example

Input use cases:

External actor (sender)

System actor (receiver)

Assumed composition rule:

Example
Synthesis of Executable SDL Specifications from MSC

- Example

Generating SDL process from Event Automaton
Synthesis of Executable SDL Specifications from MSC

Steps of the synthesis algorithm
Synthesis Methodology

- The synthesis methodology is an iterative process, consisting of the following four phases.
  - Preparation
  - Dynamic collection of probe traces
  - Synthesis of SDL model
  - Investigation of the SDL model
Synthesis Methodology

- **Preparation Phase**
  - Analyze code
  - Select modeling viewpoint
  - Set coverage goal and select probes.
  - Collect known primary scenarios + regression tests.

- **Dynamic collection of probe traces**
  - Instrument legacy
  - Run legacy code to generate probe traces.

- **Synthesis of SDL model**
  - Translate probe traces into event-oriented MSCs
  - Add conditions to MSCs.
  - Synthesize SDL model.

- **Investigation of the synthesized SDL model**
  - Terminating criteria.