Protocol Synthesis
Protocol Synthesis

- Protocol Synthesis is very important technique for efficiently designing of correct protocol specification. Protocol synthesis attempts to formalize and automate the process of designing communications protocols. 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.
• Protocols constitute the backbone of any communication system upon which there is an increased reliance to provide distributed, reliable and safety-critical communication services. In communication systems, the concepts of services and protocols play key roles. A communication system is viewed as a service provider which offers some specified services to a number of service users (U1, U2 .... Un) who access the system through many distributed service access points (SAP1, SAP2 .... , SAPn). Service is considered as High level abstraction.
Figure 1. Communication service
At lower level abstraction, communication services consist of number of cooperating Protocol Entity which interact among themself and message that not observable to users at SAP.

PE interacts with service user through upper SAP.

PE exchange message over reliable communication medium according to FIFO discipline.

These PE have their own service access points for assessing FIFO medium called lower SAPs.

Communication service specification: describes distributed functions provided by communication system to its service users.

Communication protocol specification: describes behavior of PE each servicing a particular SAP.

PE specification describes behavior of that entity with respect ot upper interface with service users w.r.t to its lower interface with underlying service provider.
Figure 2. Communication protocol.
There are three properties must be guaranteed in protocol design.
Safety properties so that protocol never enters undesirable state. (bad wont happen) freeness from deadlock, liveness, completeness.
Liveness properties of protocol to ensure protocol enters a desirable state (Good things happen) meaning protocol performs its intended functions wrt service specification. Responsiveness properties to ensure protocol has following features.
a. timeliness wrt response time
b. fault tolerance or stability that is possibility of recovering from transient failure.
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: Synthesis process can either start from scratch with only informal specification requirements or from complete specification of one PE, or from partial specification of PE or from complete service specification.

  − The modeling formalism used to describe the protocol-FSM
  
  − The constraints on communication model: Interaction among entities, number of entities that can be synthesized, ordering of messages and reliability of the medium.
  
  − 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 :: Designer specifies one of the two protocol and then the synthesis tool automatically construct the other one.

Interactive: Designer simultaneously builds two fsm interactively with synthesis tools.
Interactive Synthesis Algorithm

- Components of the synthesis algorithm:

- Set of production rules - rules define rules for producing global states, arcs and transition states.
  - Set of deadlock avoidance rules - check for non occurrence of following conditions:
    - 2 process states are both receiving states or one is a receiving state and
    - other is a final state and 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

- Beginning from FSM, specification of service automatically derives the protocol entities that provide the set of services given in the service specifications.

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:
- 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.
There are 2 kinds of MSC diagrams- Basic Message Sequence Charts. (BMSC), and High Level Message Sequence Charts (HMSC).
MSC diagram describes behavior of several instances. Each instance has an instance head, which contains the name of the instance. Instance Axis correspond to the timeline of the instance.
Instances exchange messages which is shown as arrows between two instances, or between an instance and the frame of the diagram. An instance can be created by another instance, which is shown as a dashed arrow pointing at the instance head of the child instance.
Automatic Synthesis of SDL from MSC

Elements of a basic MSC diagram
MSC uses timers. How timer T is set by instance C is shown in figure and then expires resulting in timeout.

Events on each instance axis are ordered. If the first event occurs higher on the instance axis than the second one, the first event occurs before the second one. MSC standard has syntax for some data related aspects of specification, but no semantics are described for them.

Composition of basic MSC is represented graphically using HMSC. An HMSC diagram contains references to basic MSC diagrams and flowlines.
Automatic Synthesis of SDL from MSC

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

- Synthesis algo is based on concept event automata that corresponds to MSC instance. Such that the alphabet of the input symbols for the automation is same as alphabet of MSC events of the given MSC instance.

- Synthesis algo constructs a particular kind of event automata, which we call MSC slices. MSC slice (corresponding to an MSC instance i) is an event automation, accepting all valid event sequences for the instance i.

- 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). 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); 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,
Example

- Consider a simple MSC model, it contains 2 cases Wait and Reply. Instance R receiver, corresponds to the system actor and instance S is an external actor. Sender S initiates both cases. The case, wait is started by sender S sending message Y. Receiver has to respond with message W.

Steps of synthesis algorithm for constructing an MSC slice for instance S.
1. constructs an event automaton from HMSC graph. Event automaton is constructed from instance S at MSC Wait. This has only one transition labeled with an active event out \((x,r)\). Then corresponding transition in the initial event automaton is substituted for newly created event automaton for instance S from MSC wait. Process instance S from MSC Reply. And named NEA.

The non-deterministic MSC slice for instance S is made deterministic and minimized DEA S. SDL graph is then generated for process Synthesized.
Synthesis of Executable SDL Specifications from MSC

- Example

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:

1. msc composition
2. msc wait
3. msc reply

4. NEA S
5. out(x,r) in(w,r)
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.
Protocol Re synthesis

- Consist of set of rules that are applied to different protocol entities after a modification to the service specification in order to produce new synthesized. For each simple modification made on the service specification, its corresponding automatic synthesis rules are defined.