Basic Step Algorithm

- Algorithm for executing single step of Statecharts
- Inputs
  - Status of system
  - Current time (time unit defined by clock rate)
  - External changes (events, changes in values of conditions and data-items)
- Output:
  - New system status
- Representation of system status
  - Active states
  - System activities
  - Current values of conditions and data-items
  - Internal events generated in previous step
  - Scheduled actions
  - Timeout events
  - History of States
Basic Step Algorithm
Stage 1: Step Preparation

• Add external events to event list
• Modify values of conditions, data-items, activities according to external changes
• Evaluate scheduled events
  – Action sc(a,d) adds (a, next-a) to scheduled event list
  – Execute scheduled action ‘a’ if next-a <= current time
• Evaluate timeout events
  – (E, next-E) in event list
  – E = tm(e, d)
  – If trigger ‘e’ generated, set next-E
  – Generate timeout event E when next-E <= current time
Basic Step Algorithm
Stage 2: Compute Contents of the Step

• Find set of enabled compound transitions (CTs)
• Eliminate conflicting CTs with lower priorities
• Construct non-conflicting transition sets from remaining CTs
• For each non-conflicting set, determine enabled static reactions (SRs)
• Possible Scenarios
  – No enabled CTs or SRs – empty step
  – Single set of CTs and SRs
  – Multiple sets of CTs and SRs - nondeterminism
Basic Step Algorithm
Stage 3: Execute the CTs and SRs

• For each enabled SR, execute associated action
• For each enabled CT
  – Update history for parents of exited states
  – Delete exited states from list of active states
  – Execute actions associated with
    • States exited by CT
    • CT itself
    • States entered by CT
  – Add entered states to list of active states
Basic Step Algorithm
Example for Computing Nonconflicting Sets

• Enabled non-conflicting sets
  – \{t3, t4, t7, sr1, sr2, sr3\}
  – \{t3, t4, t8, sr1, sr2, sr3\}
  – \{t3, t4, t9, sr1, sr2, sr3\}
  – \{t3, t6, t7, sr1, sr2, sr3\}
  – \{t3, t6, t8, sr1, sr2, sr3\}
  – \{t3, t6, t9, sr1, sr2, sr3\}
Basic Step Algorithm
Implementation Issues

• Assignment of values in two stages:
  – Create list \{element, new-value\}
  – Assign all values at end of step
  – Insensitive to order of execution

• Conflicting assignments
  – Element assigned value twice in single step
  – Write-write racing
  – Last assignment counts

• Chain reactions
  – Default: activity and subactivities executed in same step
  – Activity with controls: subactivities activated on following step
Basic Step Algorithm
Implementation Issues

• Events generated in a step collected in list as input to next step

• Scheduled events
  – Action sc(a,d)
  – Added to scheduled events list as (a, current-time + d)

• Termination Connector
  – Considered basic state
  – When entered, no more steps executed
Models of Time in Statecharts

• STATEMATE supports two timing models:
  – Synchronous
  – Asynchronous

• Synchronous time model
  – Assume single step executed every time unit
  – Reacts to external changes occurred since end of previous step

• Asynchronous time model
  – Reacts when external changes occur
  – Allows several changes to occur simultaneously
  – Allows several steps at once – superstep

• In both models, execution of step takes zero time
Synchronous Time Model

• Suited for systems synchronized on system clock
• In STATEMATE, single step executed using ‘GO’ command
• Semantics for GO (assume previous step at t)
  – Execute external changes since end of last step
  – Increment system clock
  – Execute timeout and scheduled actions with due time in interval \((t, t+1]\)
  – Execute basic step algorithm
Asynchronous Time Model

• Suited for reactive systems
• Execution of step does not advance time
• In STATEMATE, GO-REPEAT command executes superstep
• Semantics for GO-REPEAT
  – Execute external changes since end of last step
  – Execute timeout and scheduled actions that are due
  – Execute basic step algorithm until system in stable state (no generated events, no enabled CTs, SRs)
• GO-REPEAT may result in infinite loop
Asynchronous Time Model

• Example of GO-REPEAT execution
  – Assume C1, C2, C3 false, and event e generated
Asynchronous Time Model

- To advance time, need to execute GO-ADVANCE
- Assume current time = t, want to advance to time t + n
- Semantics for GO-ADVANCE
  - Execute external changes since end of last step
  - Repeat until time = t + n
  - Execute due timeout and scheduled events
  - Execute GO-REPEAT
  - Increment time to nearest timeout or scheduled event
Asynchronous Time Model

• Other commands supported by STATEMATE:
  – GO-STEP
  – GO-NEXT
  – GO-EXTENDED
• GO-STEP: perform single step without advancing time
• GO-NEXT: advance clock to nearest timeout or scheduled event
• GO-EXTENDED: combination of GO-NEXT and GO-REPEAT
  – Advance clock to nearest timeout or scheduled event
  – Execute timeout and scheduled events
  – Execute superstep
STATEMATE Generated Code

• RTL code style
  – VHDL or Verilog code
  – Steps sensitive to rising or falling edge of clock
  – Similar to synchronous mode

• Behavioral code style
  – Reacts to input changes when they occur
  – Similar to asynchronous mode

• Two schedulers
  – CPU clock time – steps take more than zero time
  – Simulated clock – advances time only when system reaches stable state
Racing Conditions

• Two cases:
  – Value of element modified more than once at single point in time
  – Value of element modified and used in single point in time

• Transitions or actions in one step can be in conflict with another step

• Racing conditions in superstep
  – Enabling order: each transition must be executed after the transition that enabled it
  – If two executions obeys enabling order but produce different results, then racing detected
Racing Conditions

• Example of racing

![Diagram showing racing conditions with examples of transitions and state changes.]
Multiple Statecharts

- Can be interpreted as orthogonal components in a superstate
- Used to represent concurrency
- Termination of one statechart does not affect others
- Statecharts can be reactivated in default configuration
- Asynchronous model – All statecharts executes superstep at same time
- Synchronous model – Each statechart may have different time units for evaluating time expressions
Multiple Statecharts

• Example of multiple statecharts