State Machines and Statecharts

Formal Specifications
CS236368
What is a State?

- A snapshot of the system
- Set of values for variables
- Control location(s)
- Contents of channels
- May be written as a tuple

\[ x = 17, \quad y = 0, \quad z = 8 \]

light_on

alarm_off

msg = “abc”

(x=17, y=0, z=8)
A Simple State Machine
An abstract watch:

- Initial state: battery-off
- Transition states: time → date, insert → battery-off, date → insert, battery-off → time
- States: time, date, insert, battery-off
- Transitions: p, rem

Diagram:
- Initial state: battery-off
- States: time, date, insert, battery-off
- Transitions: p, rem
What if the state is complex?

• Write values of variables in the state

• For every possible combination of values, there is a different state

• Alternative terms:
  • State Machine Graph
  • State Transition Graph
  • Explicit State Graph

• Paths in the graph = Traces = Executions
Execution sequences

• State view:
  < time, date, time, date, bat-off, time >

• Transition view:
  < p, p, p, rem, insert >
Formal Definition

- A state machine is a 4-tuple \((S, A, I, T)\) with
  - \(S\): set of possible states
  - \(A\): set of possible labels on transitions
  - \(I\): set of initial states (subset of \(S\))
  - \(T\): transition relation over \(S \times A \times S\)

- \((s_1, a, s_2)\) is in \(T\) when

- \(T\) is a relation because of nondeterminism
Example

- $S = \{ \text{time, date, bat-off} \}$
- $A = \{ \text{p, rem, insert} \}$
- $I = \{ \text{time} \}$
- $T = \{ (\text{time, p, date}), (\text{date, p, time}), (\text{time, rem, bat-off}), (\text{date, rem, bat-off}), (\text{bat-off, insert, time}) \}$
When are State Machines Used?

• When it is easy to abstract away irrelevant details, leaving only a small number of states

• When we want to precisely examine every possibility using model checking

• For communication protocols and complex distributed algorithms

• For safety critical, or firmware that can’t be changed later
The State Explosion Problem

• State machines can get huge for very simple systems

• Need a more compact representation!

• Describe the set of states from which a transition appears through a predicate

• Describe the `target’ through the changes from the source
• Use text and not pictures
Z describes State machines!

- A collection of operation schemas, with explicit preconditions, can be seen as a state machine representation.

- There are many other notations, that mainly treat concurrency.

- To generate explicit version from schema, start from initial states, apply relevant operations. Repeat until nothing is added (a fixed point is reached).
A Z State Machine

• $P = [ \text{st, st}' | \text{st} = \text{time or st} = \text{date and} \notag$
  
  $(\text{st} = \text{time}) \Rightarrow \text{st}' = \text{date and} \notag$

  $(\text{st} = \text{date}) \Rightarrow \text{st}' = \text{time} ]$

• $\text{rem} = [ \text{st, st}' | (\text{st} = \text{time or st} = \text{date}) \text{ and}$

  $(\text{st}' = \text{bat-off}) ]$

• $\text{insert} = [ \text{st, st}' | \text{st} = \text{bat-off and}$

  $(\text{st}' = \text{time}) ]$

• $\text{new} = [ \text{st}' | \text{st}' = \text{time} ]$
Hints on translating SM to Z

• Can do one-to one translation,
  • gather the arrows that correspond to a single label into a schema with the name of the label
  • preconditions are union of the source states
  • predicate has source => target for each arrow

• Can often combine into general predicates and state transformations

• Explicit graph means a different state for every possible combination of values
Statecharts

• Graphic state machine representation

• Compact--- avoids state explosion

• Uses zoom-in, zoom-out

• Separates concerns

• By David Harel, while consulting for industry (avionics)
Causes of clutter

• Pulling eject lever leads to a special situation (many arrows, can double the number of states)

• Changing gears and adjusting the lights are independent of each other (multiplies the number of states describing gears by the number describing light settings)

• There is a time display updated every 0.1 second, and a similar stopwatch display
Overcoming the Problems

• Group states into Superstates that can be entered or left under uniform conditions

• Keep independent parts separate -- do not take the cross product

• Hide and separate out most values of variables

• Keep `activities’ and data manipulation outside the statecharts
Superstate (without parallel)

- Composed of a lower level statechart
- in $S \equiv$ in time or in date
- In a superstate $\Rightarrow$ in one of its substates
- Arrow from boundary $=$ arrow from each (top level) interior state

- Still can have direct exit from a state
How to Enter

• Arrow to the boundary, and enter the default state (denoted by a short arrow)

• Direct to an inner state

• Unspecified at this level
More details on transitions

• Form: name [ cond ] / action
  • Only do the transition if the cond holds
  • Do the indicated action as a side effect

• Actions can include
  • Updating a variable
  • Sending a message
  • Starting or stopping an external “activity”
  • Output

• Conditional entrance to targets
  (and the alternatives have predicates)
A Statechart for Power Failure

- **Normal**
- **Alternate**
- **Test**

- **EPF (in (test))**
- **EPF (not in (test))**

- **Mode**
  - **Normal**
  - **Alternate**
  - **Test**

- **Power off/Mode := 'stop'**

- **Power up**

- **Warm_start**
- **Cold_start**

- **Done**

- **Reconfig**

- **Long_p**

- **Toolbox**

- **Take_down**

- **Battery**
Parallel Processes

• For describing independent components

• Dotted lines between statecharts, each with a default

• in S == in each component of S
• arrow to boundary == enter each default
• Arrow from boundary == leave all components

• Can have multi-headed or -tailed arrows
Transitions in Parallel Components

• Transitions are used to `coordinate’ components when needed
• Recall: name [ cond ] / action
• Conditions can include being in a certain state in a parallel component
• Actions can include
  • Updating a variable
  • sending a message
  • starting or stopping an activity
  • output
  • activating a transition in a parallel component
Joint Transitions

If in A and in C, s is a joint transition to (B,D), but if in A and in D, s just does A→B

If in E and in B, t does E→F, otherwise nothing happens when t ‘occurs’
Parallel "means" Almost Independent

- mode
  - regdisp
    - normal
      - p
        - done
    - alternate
      - p
        - long_p
      - test
        - p
  - toolbox
    - p
  - total_kw_cum
    - displayable
      - (in regdisp)
    - hidden
      - (not in regdisp)

- security
Two Equivalent Statecharts

\[ \text{B, F} \]
\[ \text{B, E} \]
\[ \text{B, G} \]
\[ \text{C, E} \]
\[ \text{C, F} \]
\[ \text{C, G} \]

\[ k \rightarrow \text{B, F} \]
\[ k \rightarrow \text{B, E} \]
\[ k \rightarrow \text{C, F} \]
\[ g \rightarrow \text{B, G} \]
\[ g \rightarrow \text{C, E} \]
\[ g \rightarrow \text{C, G} \]
\[ e \rightarrow \text{B, E} \]
\[ e \rightarrow \text{C, E} \]
\[ e \rightarrow \text{C, G} \]
\[ h \rightarrow \text{B, F} \]
\[ h \rightarrow \text{B, G} \]
\[ h \rightarrow \text{C, G} \]

\[ \text{f [in G]} \]

\[ e \rightarrow \text{B, E} \]
\[ e \rightarrow \text{C, E} \]
\[ e \rightarrow \text{C, G} \]
\[ h \rightarrow \text{B, F} \]
\[ h \rightarrow \text{B, G} \]
\[ h \rightarrow \text{C, G} \]
Mini-steps in Transitions

• An action can be to activate another transition (from a parallel component).

\[ e / f \] could mean that transition e activates f, so both occur in the same step....

In UML version: e now activates f in the next step -- avoids problems
(‘chain-reactions’ or even cycles)

If only f occurs, there is no influence in the other direction.
History

- H (in a circle) means the state from which the superstate was last exited is `remembered’, and an arrow from outside to the H returns the superstate to the one last left (on the uppermost level)

- H* as above, but to lowest level of nesting

- (Warning: different tools use H differently—so check what happens in the one you use…)

- clh = `clear history’ A special action that erases the history, so the default is used for the next entry.
An Example: David Harel’s Watch

Citizen Quartz multi-alarm III

- main
  - displays
  - alarms - beep
    - alarm 1 - status
      - disabled
        - d (in alarm 1 on)
        - d (in alarm 1 off)
      - enabled
    - alarm 2 - status
      - disabled
        - d (in alarm 2 on)
        - d (in alarm 2 off)
      - enabled
  - chime - status
    - disabled
      - d (in chime on)
      - d (in chime off)
    - enabled
      - quiet
        - T is whole hour
        - 2 sec in beep
        - beep
    - power
      - OK
      - OK
      - blink
    - light
      - off
      - on
      - -b
      - b
      - batt. inserted
      - batt. removed / clh(main*)
      - weak batt. dies / clh(main*)
      - dead
      - dead
Zooming in on Displays in the watch
The full watch
Tools

- Statemate, from I-Logix, uses statecharts as the basic design/specification notation, adds activities and module decomposition separately, emphasizes simulation.

- Rhapsody, from I-Logix, and Rose, from Rational (both now inside IBM), incorporate statecharts into UML tools, for design level.

- All UML tools have drawing capabilities, only a few have simulations.
Scenarios

• Statemate gives feedback to clients through its simulation facility.
  
  • When can a beep occur?
  • If in states....we press d and then b without letting go, what will happen?
  • Where does an action have an effect?

• Can `test’ use-case scenarios and `debug’ the specification relative to client needs
Object Oriented Statecharts

- In UML, Statecharts describe internal behavior of objects, while sequence charts show scenarios of message calls among different objects.

- Activations are usually message calls, and follow usual rules of delayed activation, and even queueing: different from original.

- Different versions have somewhat different rules for ordering/synchronizing—see tirgul.

- Less use of parallelism than in pure statecharts.
Guard Conditions

- Act as preconditions
Statecharts: referring to states

• The operation oclInState returns true if the object is in the specified state.—gives ‘in’ used previously
• object.oclInState( statename )
Fuzzy parts

• Real-time will be treated later

• Updating the variables is `behind the scenes’ and not handled very well in Statecharts

• Note that in the watch statechart, the actual time is not treated!

• Connections to other notations (e.g., more informal UML parts) is so-so
Summary on Statecharts

- A graphic notation: visual/Gestalt
- The precision of automata, without the clutter
- For precise design of key/complex protocols
- For control and modes of operation—not data manipulation
- Seems lower-level (assembly language of formal specifications?)
- First mention of parallel---for (almost) independent components
- Debug using simulation of use cases
- Can sometimes synthesize into code skeletons