System Analysis and Design

Statecharts

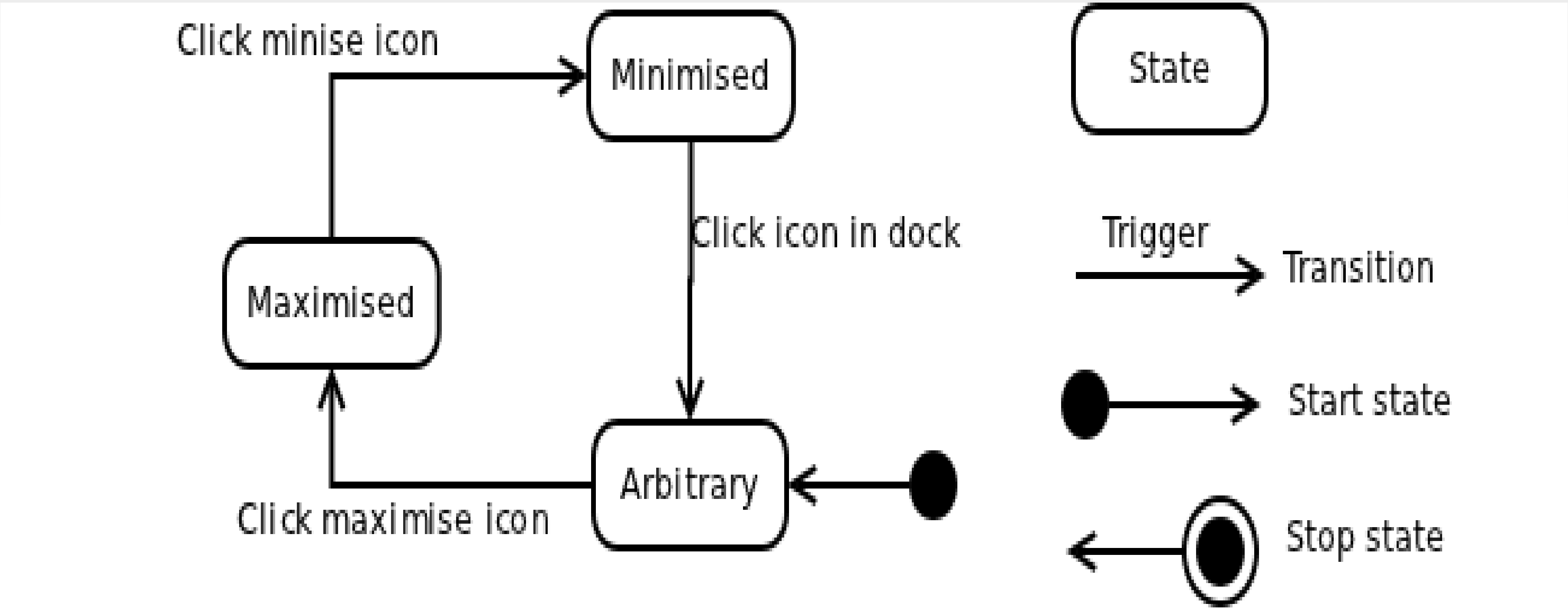
Salahaddin University
College of Engineering
Software Engineering Department
2011-2012

Amanj Sherwany
http://www.amanj.me/wiki/doku.php?id=teaching:su:system_analysis_and_design

Statechart Modelling

- Looking at a system as a set of *states* and *transitions* between the states is a powerful abstraction.
- A state can be understood as a mode of operation, and a transition simply when the system switches from one mode to another.
- A trivial example is a window on a computer screen, that can be either *minimised*, *maximized* or *have an arbitrary size*

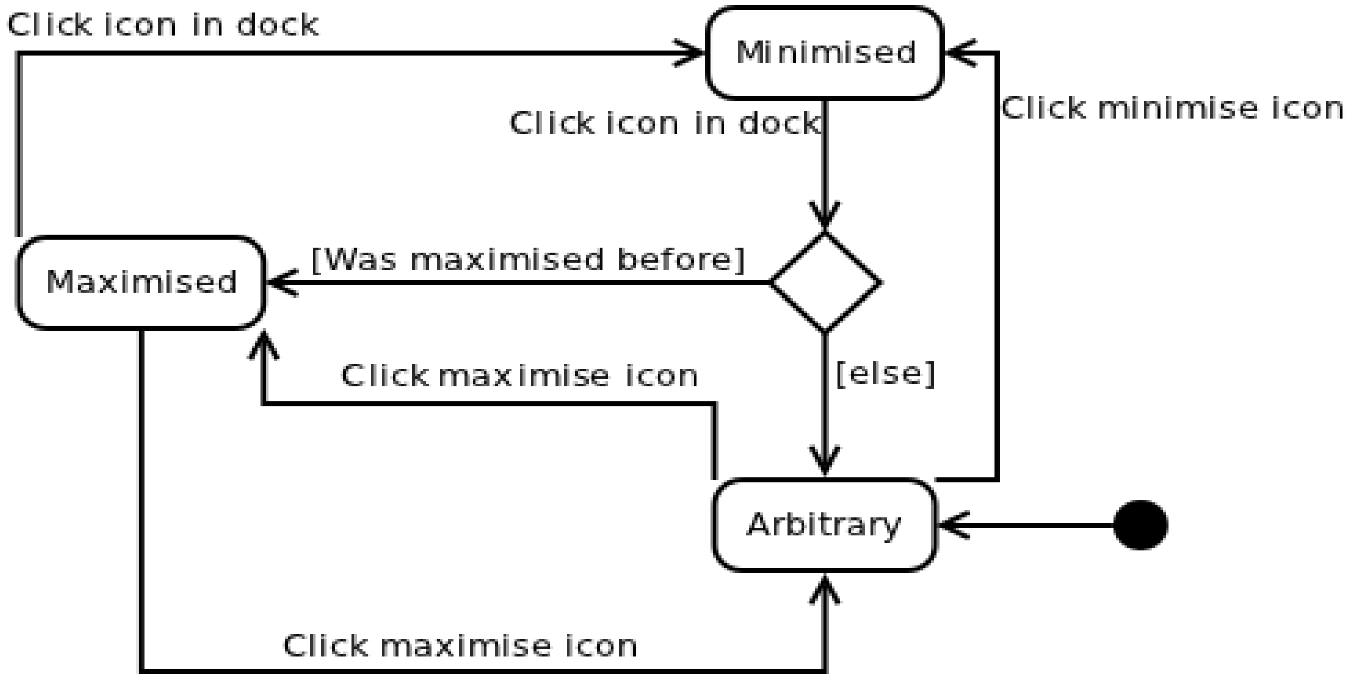
Statechart Modelling, *Cont'd*



Statechart Modelling, *Cont'd*

- Problems with the previous model:
  - There is no way to minimise an arbitrary-sized window without first maximising it.
  - If you minimise a maximised window, when you display it again, it is no longer maximised.
  - The starting state captures the initial state of the system –a window starts at arbitrary size.
- *Note:* the absence of stop state, why?

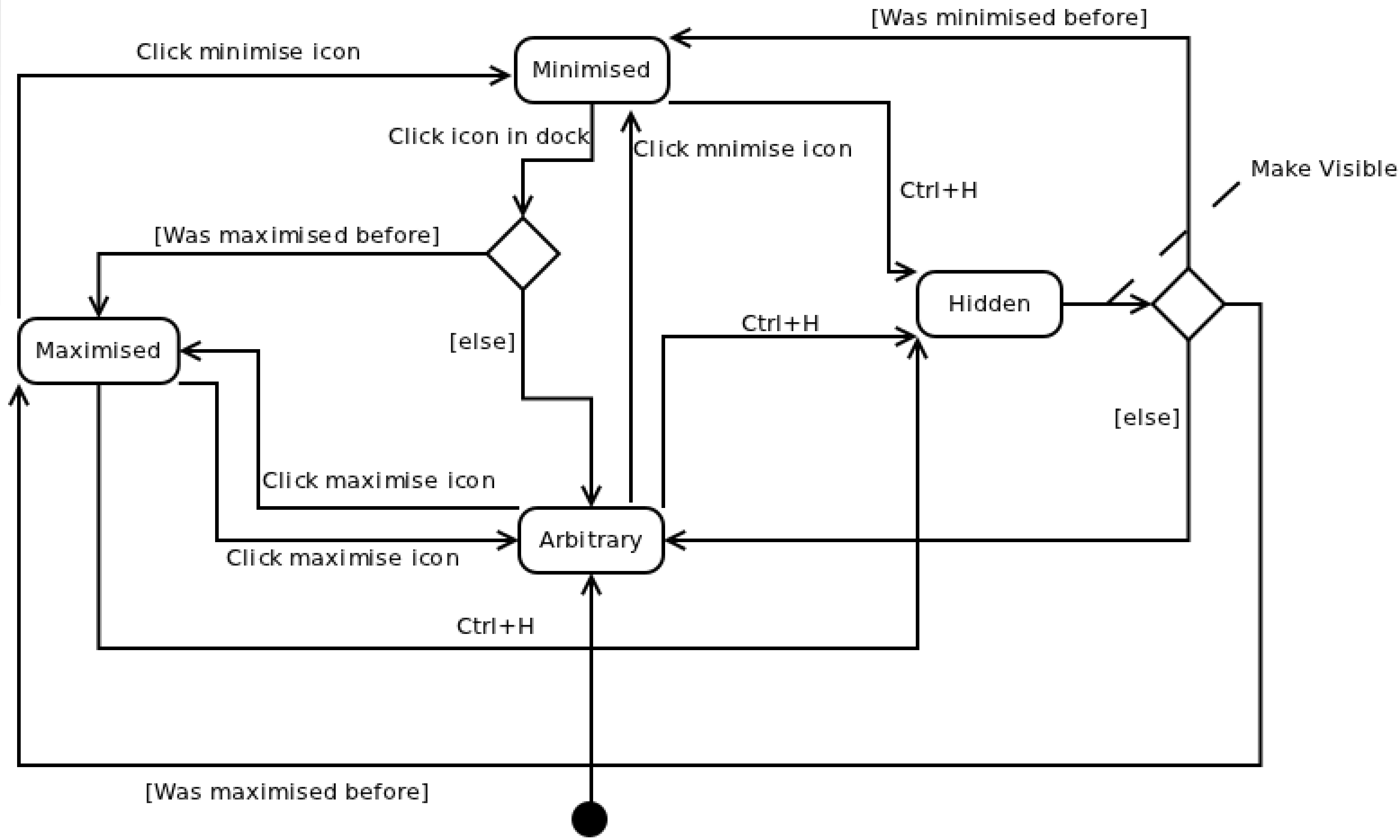
Statechart Modelling, *Cont'd*



Statechart Modelling, *Cont'd*

- In the previous model, when leaving *Minimised* state:
  - If the window was previously maximised, we go back to that state
  - Otherwise, we go to the arbitrary state.
- The *guards* that must be true for a transition to take place are written in […] brackets.
- If you are running Mac OS X, a program can be either visible or hidden, which can be modelled in several ways.

Statechart Modelling, *Cont'd*



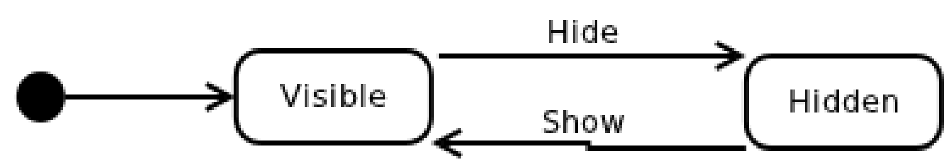
Statechart Modelling, *Cont'd*

- There are several ways to tackle the complexity of the previous model.
  - One elegant way is to use a hierarchical statechart, where states can be nested inside each other states.

Statechart Modelling, *Cont'd*

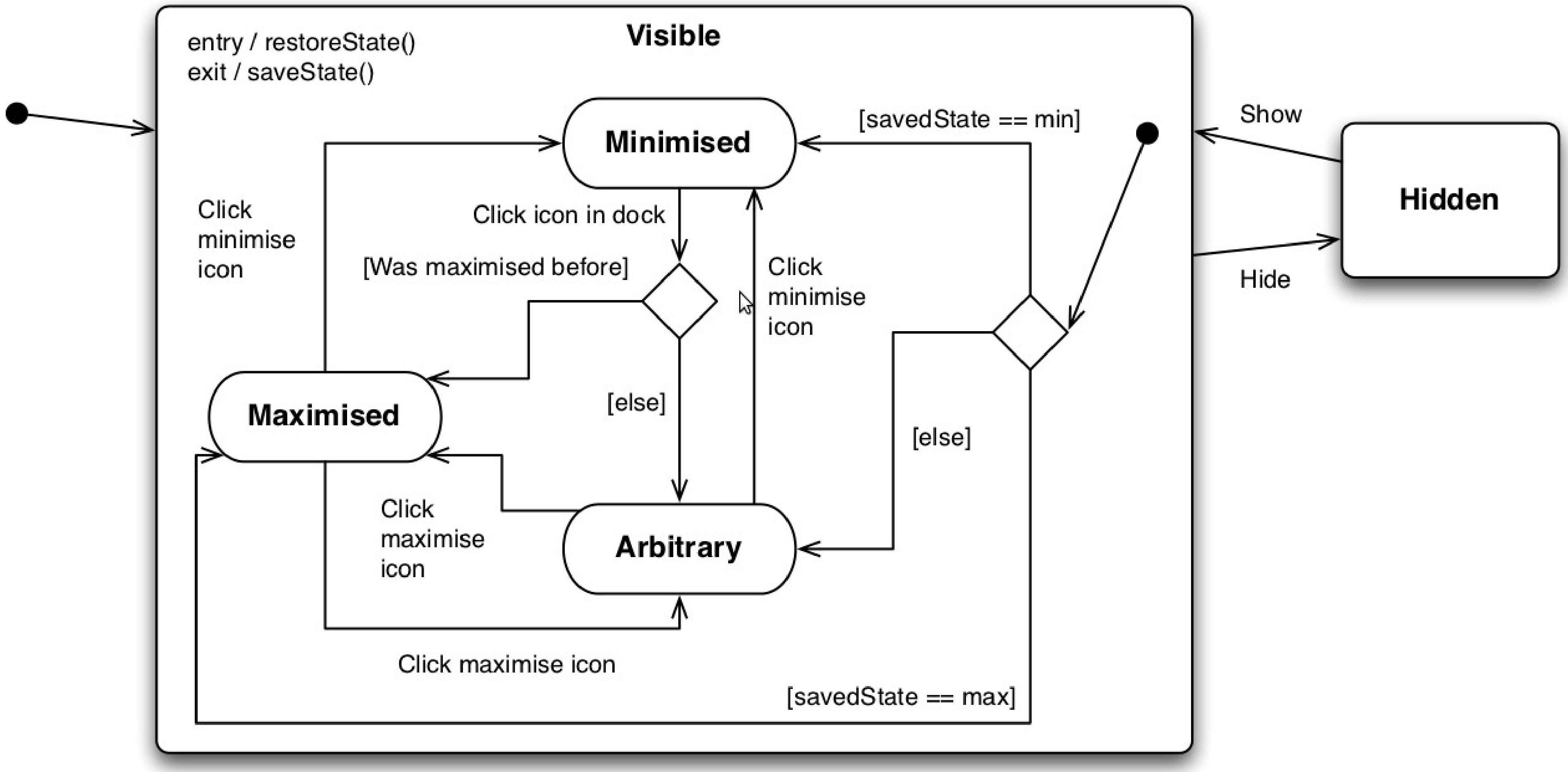
- First we can see that there is no way that you can change the size of a hidden window.
- From this we infer that a program is either hidden or visible.
- Only in the visible state can windows' size be manipulated.
- Starting from this slightly different angle, we model the system.

Statechart Modelling, *Cont'd*



- Then, we specify in greater detail what can happen when a window is visible, by exploding the *Visible* sate.
- At this point we add *entry and exit actions*, written in the top-left corner of the sate.

Statechart Modelling, *Cont'd*



Multiple Simultaneous States

- In statecharts, it is possible to be in several states at the same time: the state *Maximised* implies the state *Visible*.
- But *this only happens for sub states*!
- In all other cases, states are considered mutually exclusive, with a notable exception for “orthogonal regions”.

When to Use Statechart Models

- It is very frequently used in real-time systems and embedded systems.
- It can help in understanding time-critical behaviour, and model the number of steps from an incoming event until it is processed.
- Medical devices, financial trading services, satellite command and control systems are other examples of domains where the use of statecharts is prevalent.

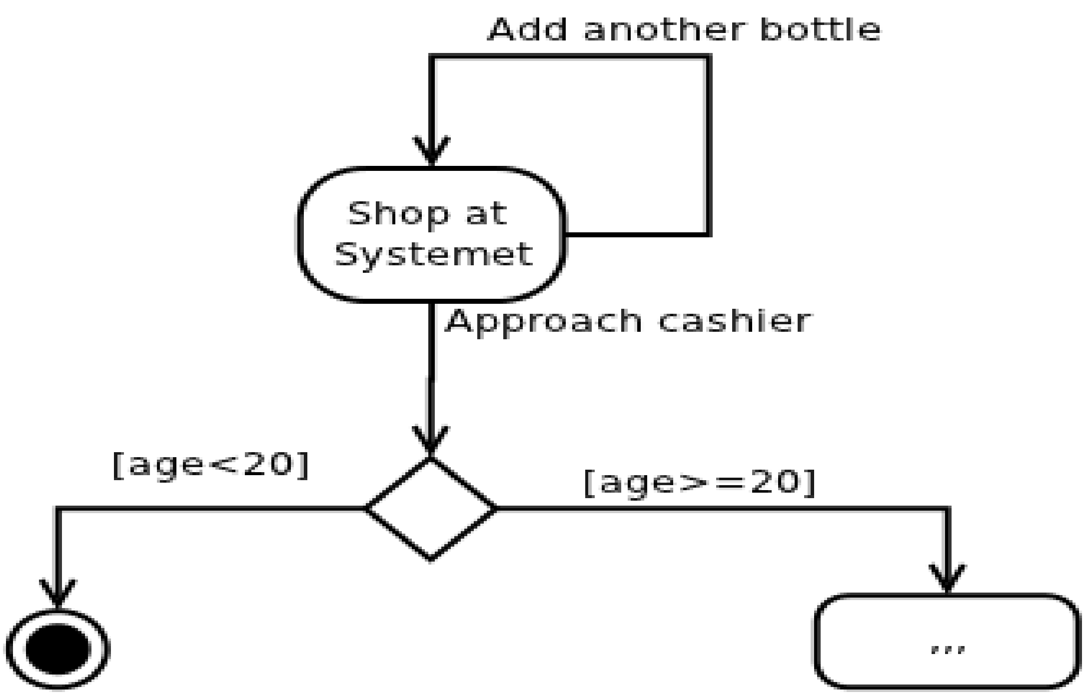
States and Transitions

- We have shown how certain aspects of a system can be modelled as *transitions* between *states*.
- The states are abstract concepts, and each state might concretely be represented by a number of variables with different values at run-time.
- Sometimes it makes sense to implement a “state variable” that captures the current state of the system.
- Checking “what state we are in” is as simple as looking at the state variable.

Triggers and Guards

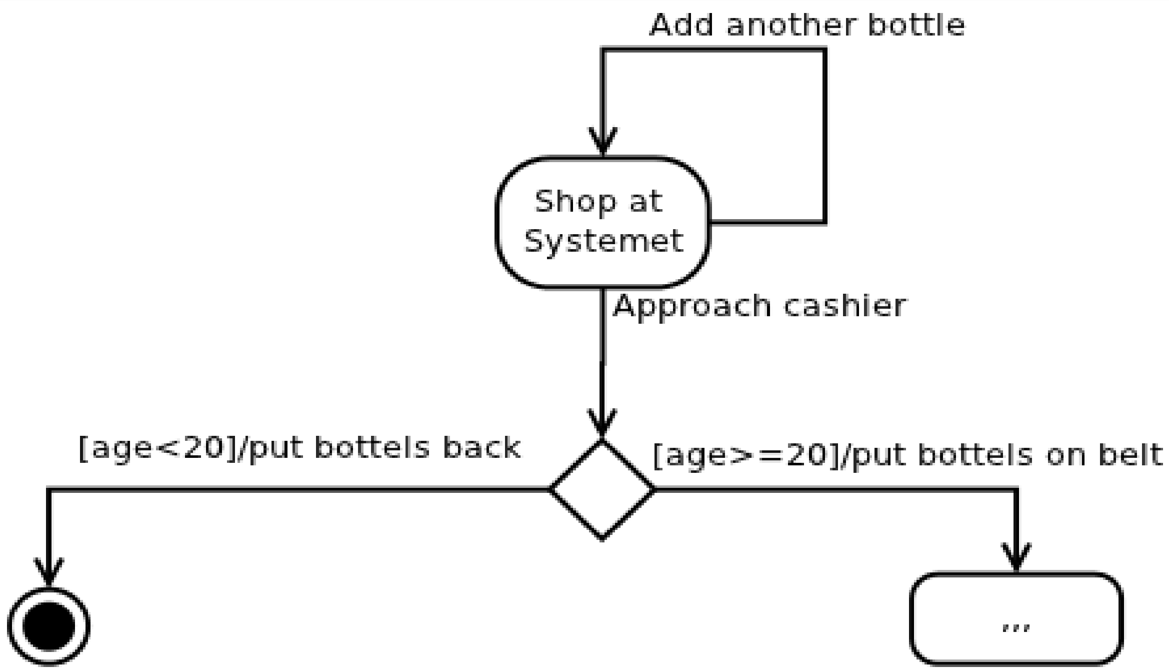
- A state transitions is labelled with the trigger that causes it.
- Sometimes a transition is guarded by a condition which must be true for the transition to happen.
- We use the diamond notion where the incoming transition is labelled in the usual way, and the outgoing transition is guarded.

Triggers and Guards



Actions

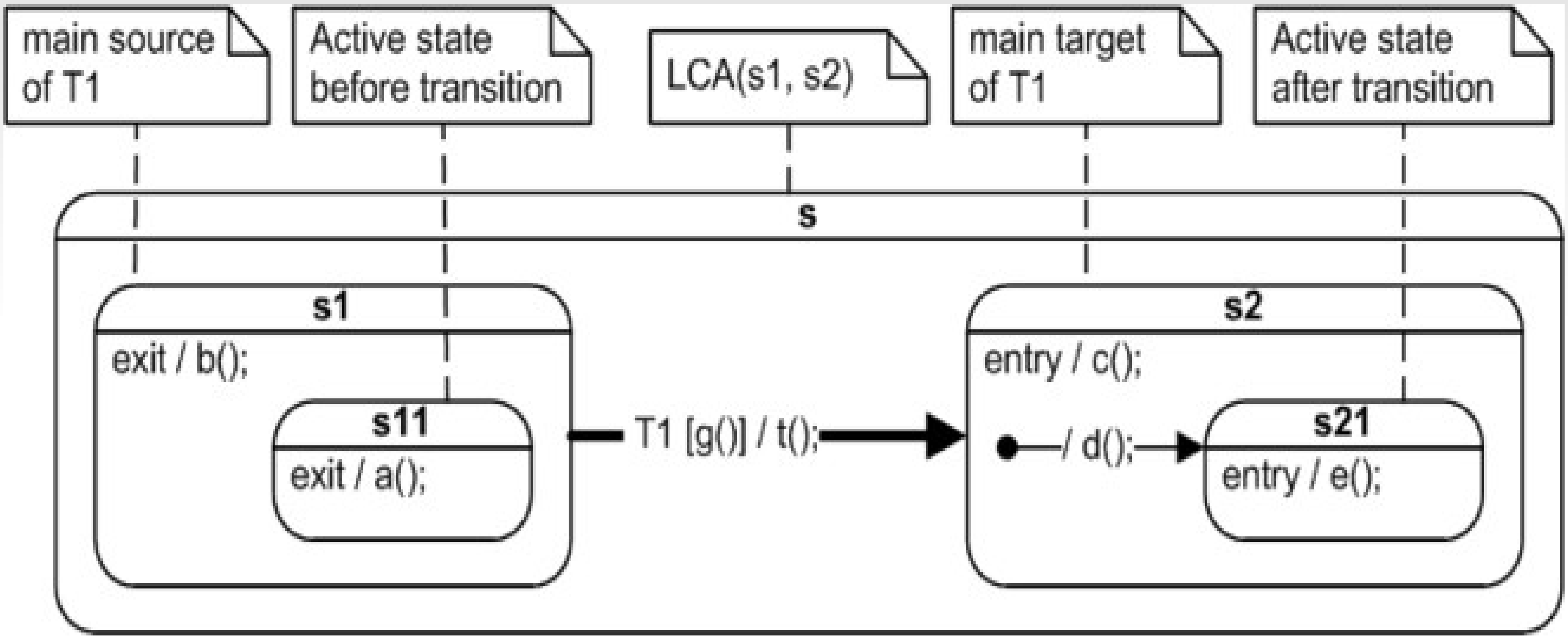
- *Entry* into a state, *exit* from a state and transition can have actions such as `updateDisplay()` (could also be in natural language).



Hierarchical State Machines

- In many cases it makes sense to look at the states of a single system from several levels of abstraction.
- The internal transitions among the substates can be modelled either inside the enclosing state or in a separate diagram.
- State transitions going from the enclosing state are considered coming out of all the nested sub states.

Ordering of Actions



Ordering of Actions, *Cont'd*

- If the guard g() evaluates to true, then the events happen in the following order: **a()**, **b()**, **t()**, **c()**, **d()**, **e()**.
  - All the exit conditions of the nested states inwards-out until we reach the outer-most state transitioned from (in this case **a()** and **b()**)
  - Then any actions on the transition (in this case **t()**)
  - Then all the entry actions followed by actions on start transitions outside-in of all the nested states (in this case **c()**, **d()** and **e()**).

Orthogonal Regions

- One aspect not covered by the previous examples is that of orthogonal regions in a state.
- Normally, with the exception of nested states, a system is in a single state only at any given point.
- However, with orthogonal regions, we can model systems as being in several states simultaneously, which is often essential for modelling concurrent behaviour.

Orthogonal Regions, *Cont'd*

- Orthogonal regions are drawn as nested states where the enclosing state is divided into several regions with a dashed line.
- The following statechart shows a web server consisting of a web service that:
  - either awaits requests or processes them, and
  - a log rotation service that moves big log files to some storage disk every 30 minutes

Orthogonal Regions, *Cont'd*