UML Hierarchical State Machine

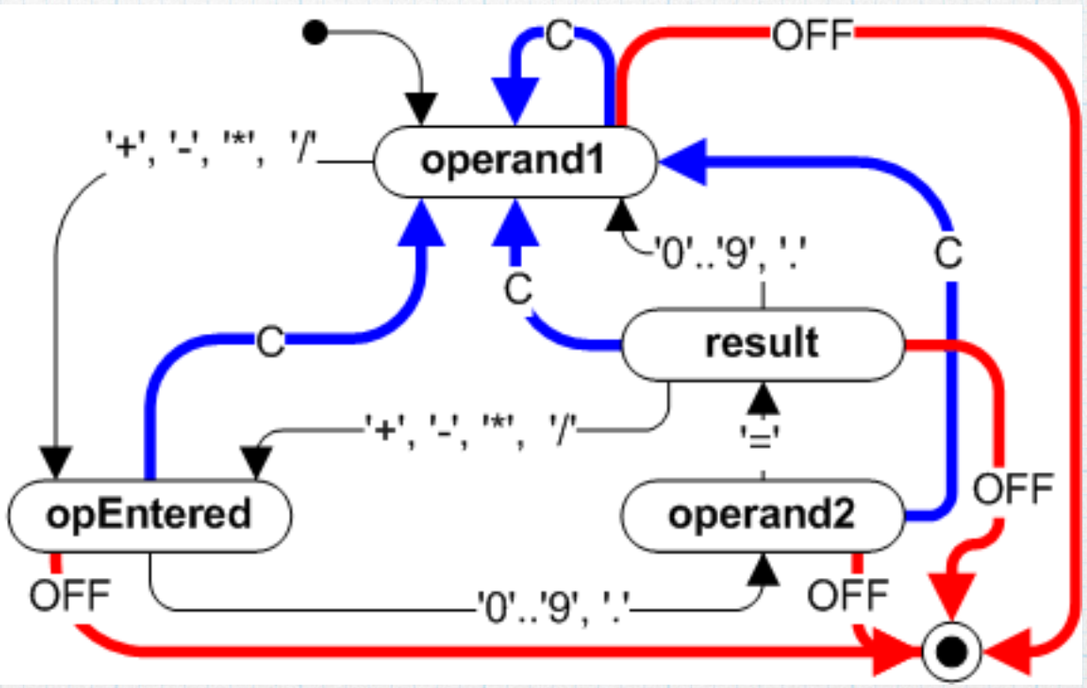
version 1.0

Victor Li
vli02@hotmail.com

A state machine

* A piece of software
* Composed with a set of states
* And a set of events
* Performs transition from one state to another and execute transition actions
* Based on events being received

Example: calculator



Why state machine

* Event arrival is unpredictable
* Action various for same event when system is in different situations
* Logic complexity increases when number of states and events increase

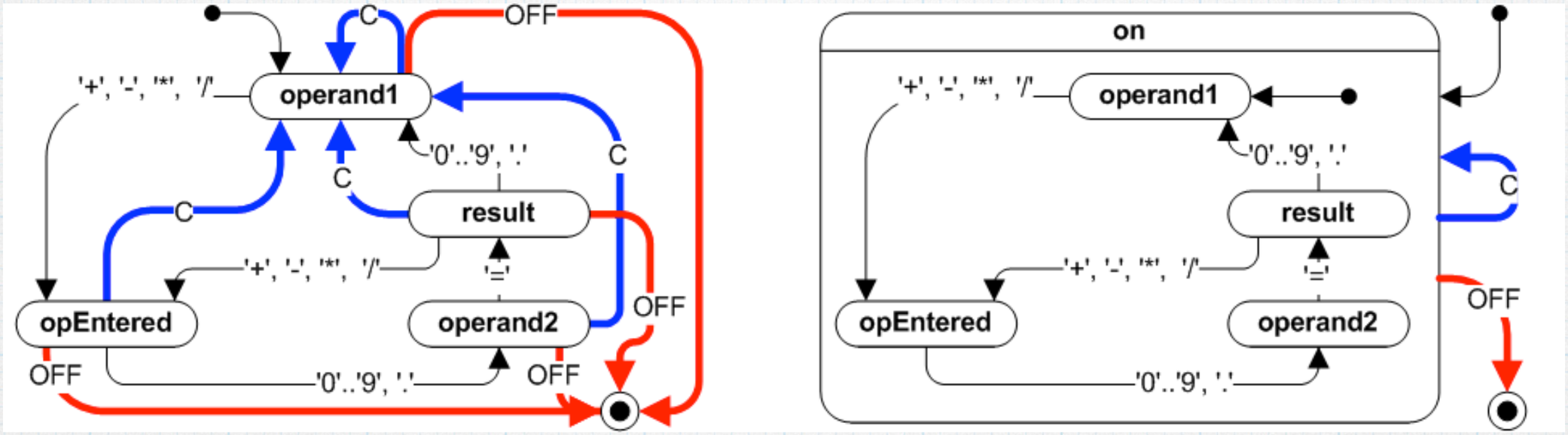
State machine design

* A state reflects a system situation
* An event is a request
* State transition rules
* Actions to execute when state transitions

Hierarchically nested states

* A composite state which reflects a common situation of a group of states
* Make it as a super state, and
* The group of states as its sub states
* Super-Sub can form multiple hierarchies

Example: calculator



* Super state "on"
* Sub states "operand1"...
* initial sub state "operand1"

Benefits

* Common event handling for all sub states
* Greatly reduce the number of transition rules, thus
* Simplify state transition diagram

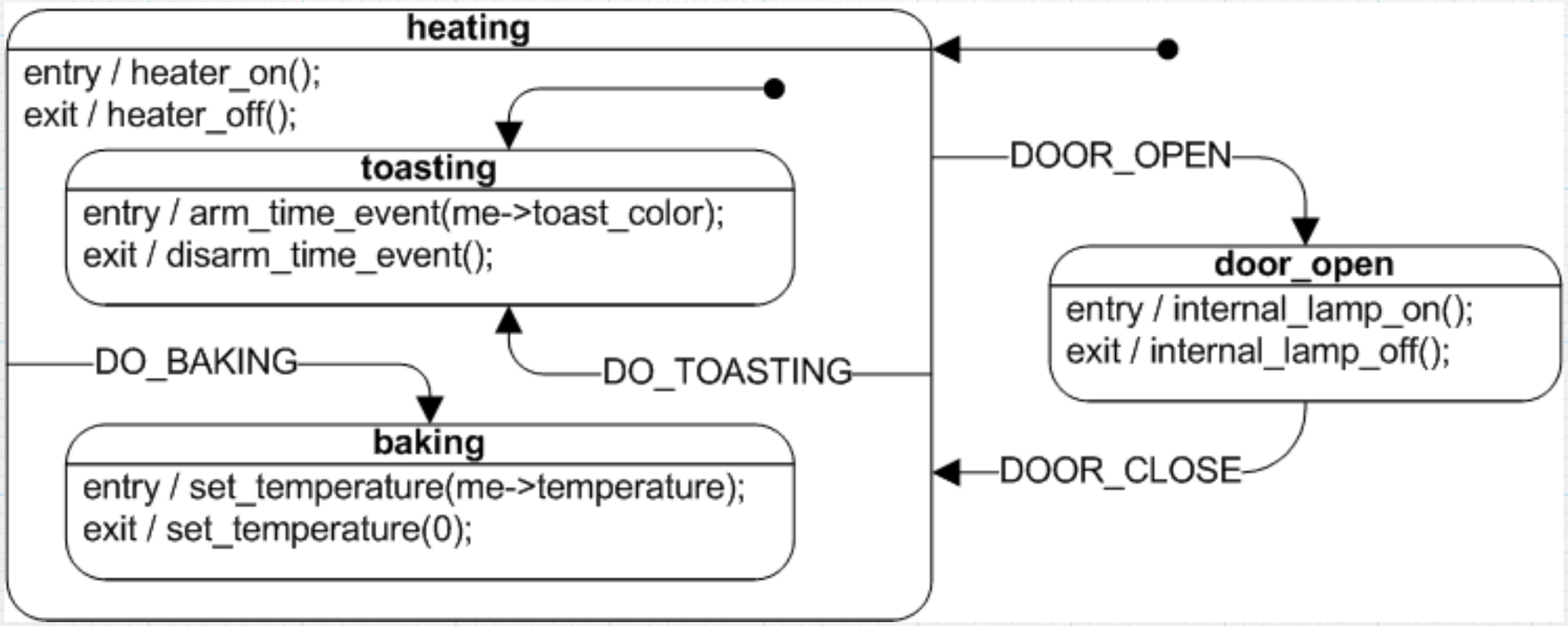
Entry/Exit actions

* Each state can have an optional entry action and an optional exit action

* Entry action is execute when transition enters the state

* Exit action is executed when transition exits the state

State notation



* a rounded rectangle labeled with state name
* entry action following 'entry /'
* exit action following 'exit /'

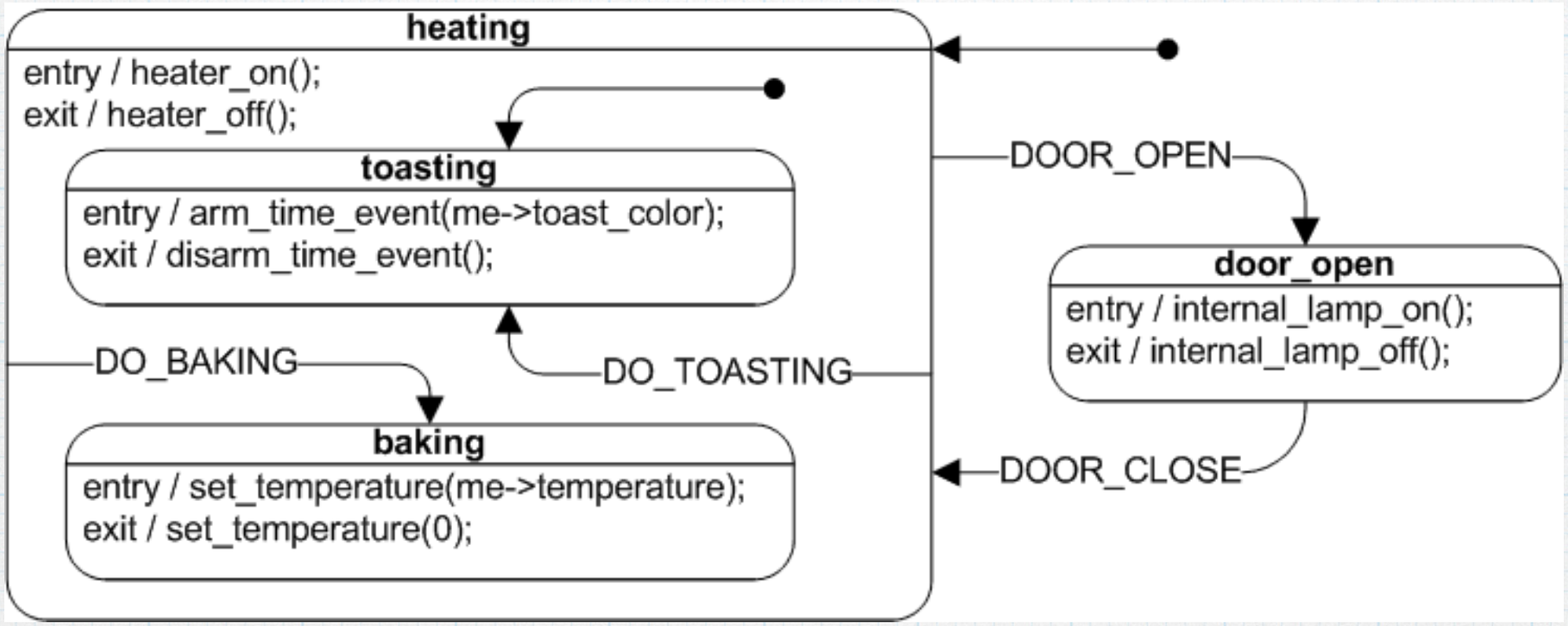
State transition

* A behavior of system reaction, when
* An event is received, and
* Taken by the state
* A transition can have an associated action
* Sub state takes precedence of handling the event, then super states if the event is not handled

Initial transition

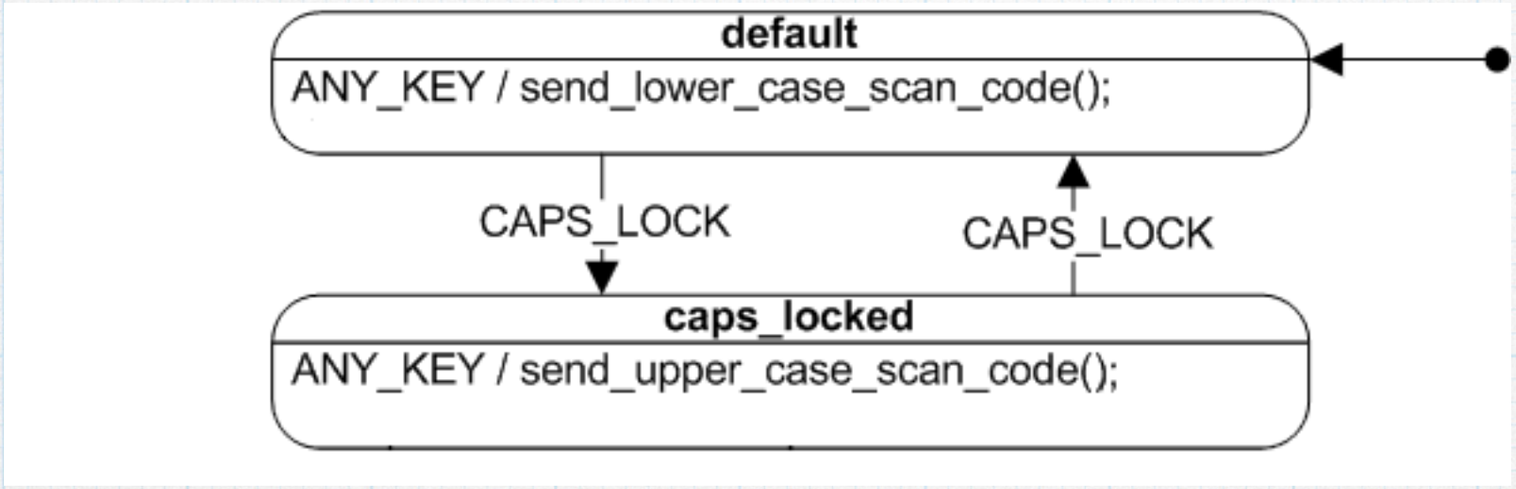
* Initial transition to start state
* Super state must have initial sub state
* If a transition destination is a super state, it always transition to its initial sub state once transition reaches to the super state
* A complete transition will always land onto a leaf state in the end

Initial transition



* originates from a solid circle, and
* points to destination state, with
* an optional transition action shown on the path as '/ action()'

Internal transition

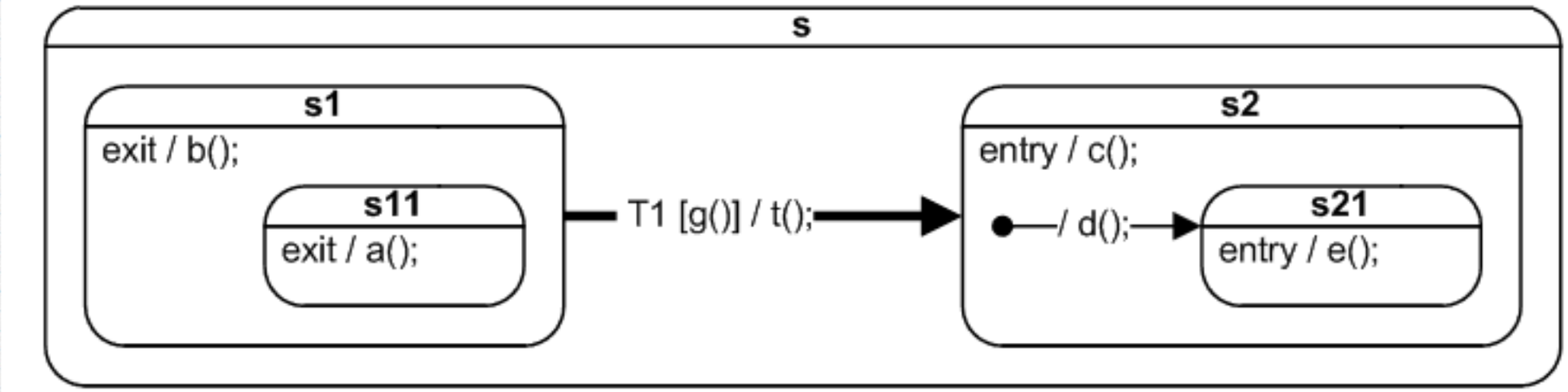


* A transition which has no state change
* Do not run exit and entry actions of the state
* Shown inside a state as 'event / action()'

External transition

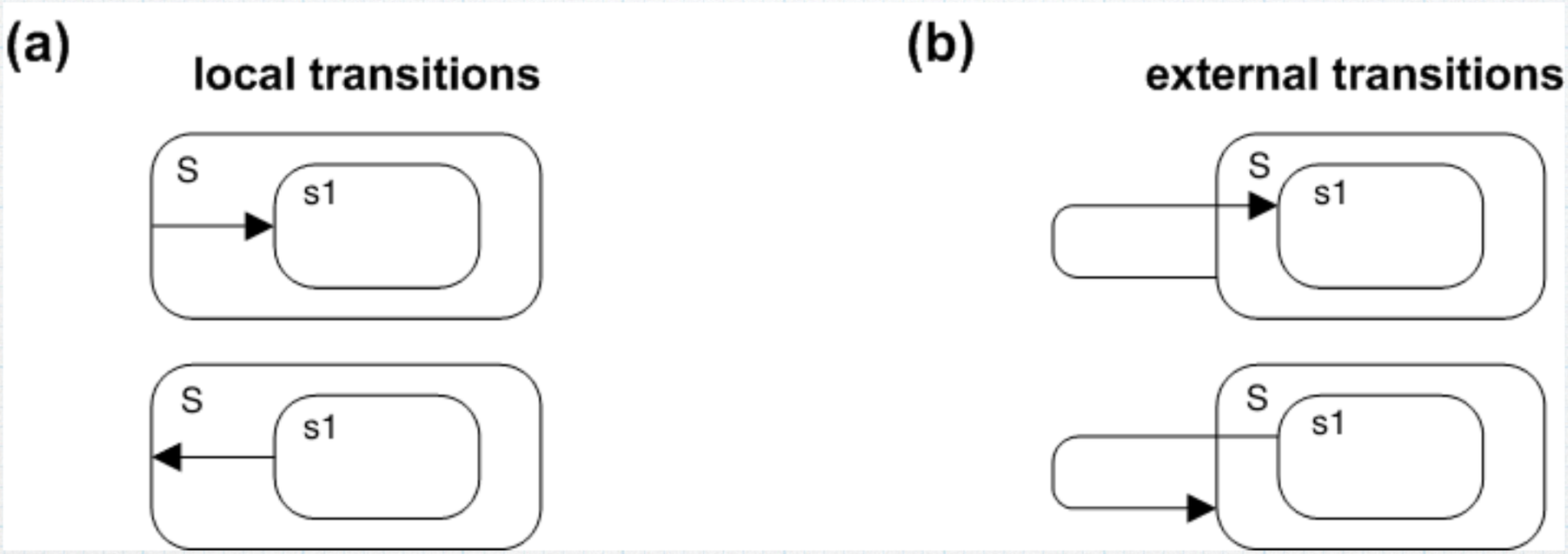
* A transition goes thru outside of originating state to destination state
* The destination state is usually a different state; or
* The originating state, i.e. Self transition
* Exit actions of source state and entry actions of destination state are executed

External transition



* Shown as an arrow with event and action as 'event / action()'
* Exit up to, but not including, the least common ancestor
* Then execute transition action and entry actions

Local transition



* A transition between a super state and one of its sub state
* Do not exit and re-entry the super state

Guard condition

* It is a boolean expression
* Evaluated when a state transition is attempted
* Enable the transition if evaluation result is true
* Disable the transition otherwise

Event deferral

* Handling of an event can be postponed on a state
* The postponed event is handled once the state changes to a different state
* Shown as a clause '[event list] / defer' inside a state

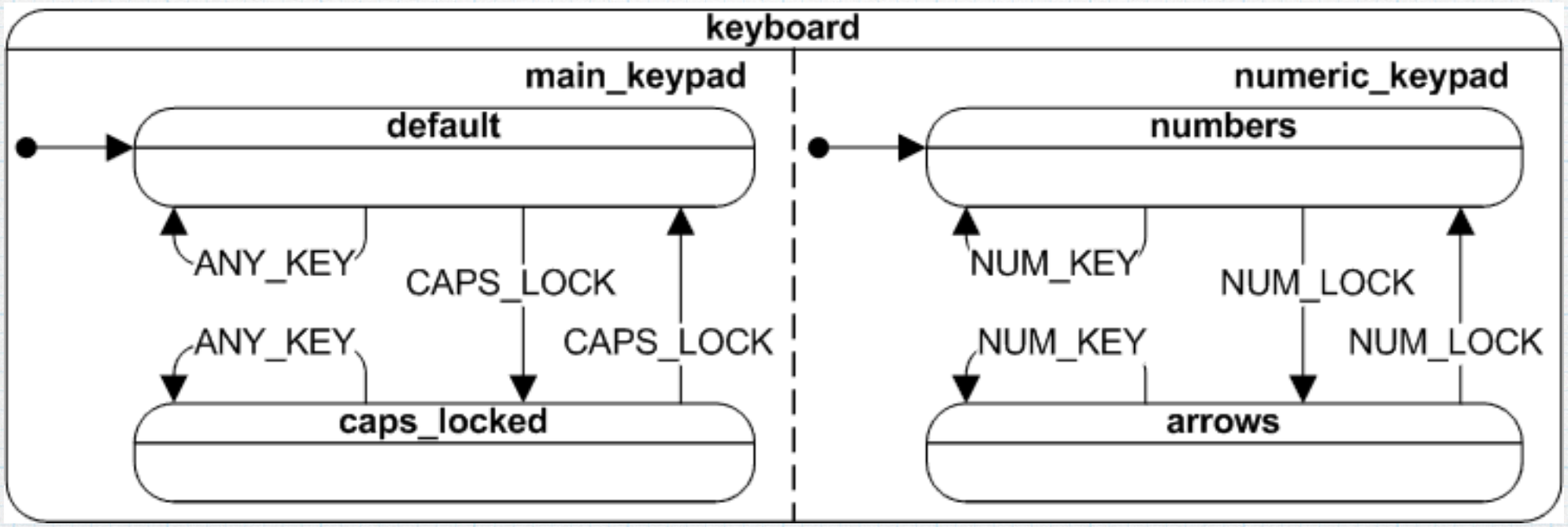
Run To Completion

* A transition has to finish before next event can be taken
* Transition guard condition is evaluated, if it satisfies
* Exit actions run first, from the innermost state to the outermost state
* Then transition action
* Entry actions run last, from the outmost state to the innermost state

Orthogonal Regions

* A super state has
* Two or more sets of sub states
* Each set of sub states is independent, and
* compatible with other set of sub states

Orthogonal Regions

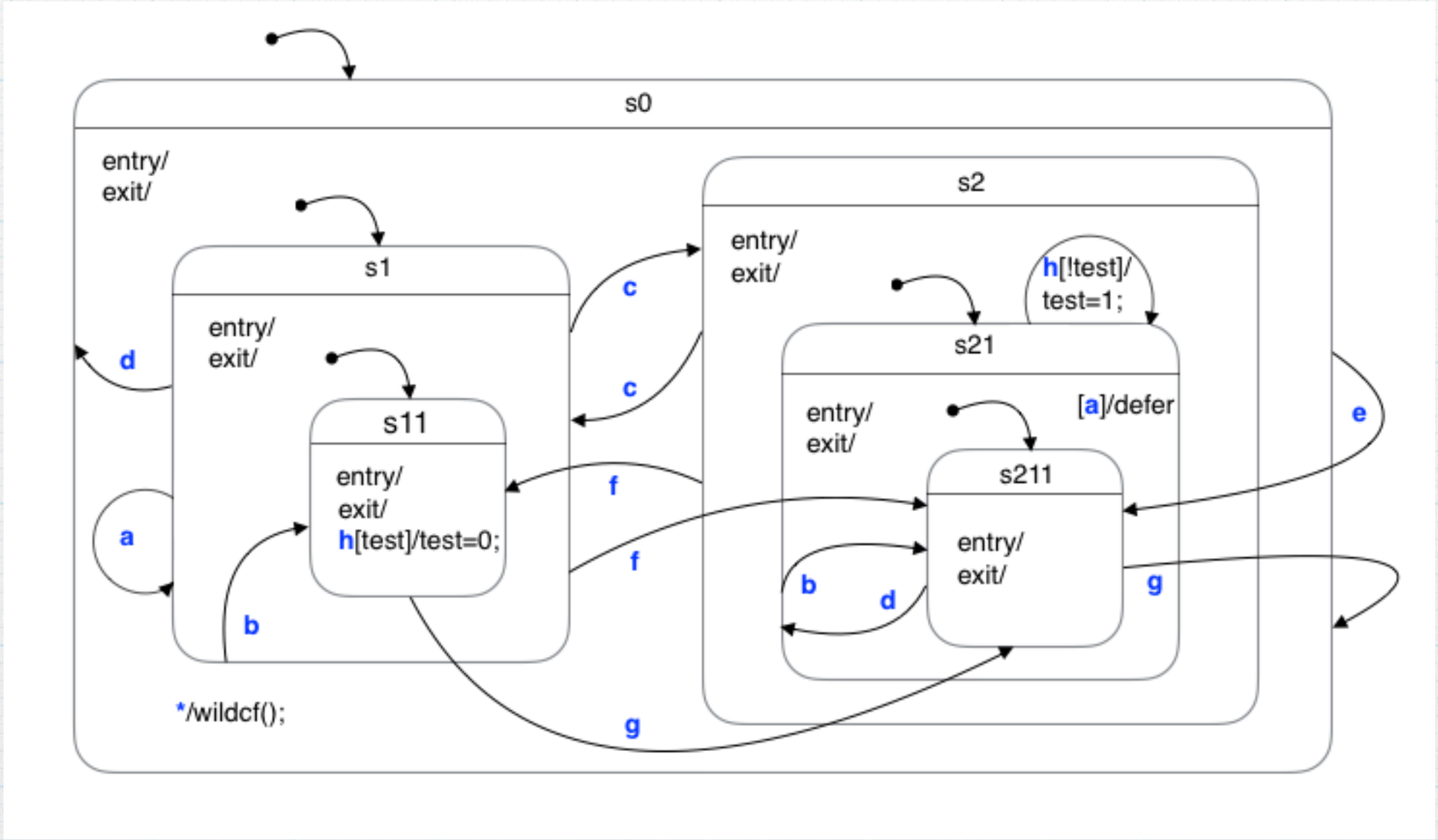


* Independent - transition happens in one region does not affect other regions
* Compatible - all regions are active simultaneously

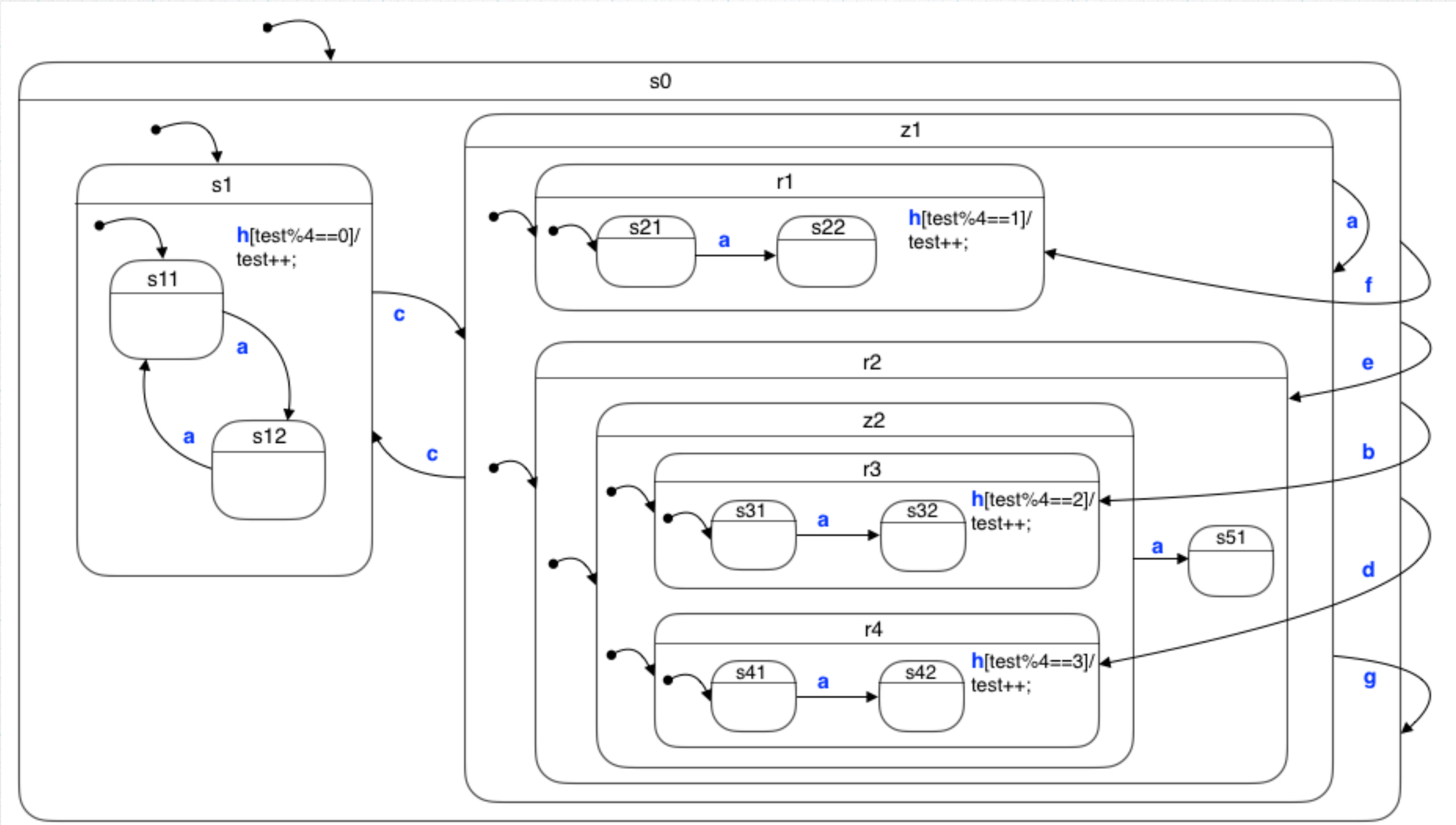
Benefits

* Reduce number of states from m * n... to m + n..., when
* system behavior fragmented into independent, concurrent active parts.
* Regions form and gone when transition enters to or exits from super state.

Example: keypress3



Example: regions



Resources

* UML State Machine @wikipedia
* [https://en.wikipedia.org/wiki/UML_state_machine]
* Intuitive Hierarchical State Machine
* Intuitive Hierarchical State Machine Programming Specification