UML State Diagrams

Fritz Bartlett
07-May-1999
A Simple State Machine

Initial State

State 1

Event Trigger
Action

State 2

Event Trigger
Action

State 3

Self Transition

Final State
Definitions

• State Machine

A state machine is a behavior which specifies the sequence of states an object visits during its lifetime in response to events, together with its responses to those events.
Definitions

• **State**

  A *state* is a condition during the life of an object during which it satisfies some condition, performs some activity, or waits for some external event.

• **Event**

  An *event* is the specification of a significant occurrence. For a state machine, an event is the occurrence of a stimulus that can trigger a state transition.
• Transition

A *transition* is a relationship between two states indicating that an object in the first state will, when a specified set of events and conditions are satisfied, perform certain actions and enter the second state. A transition has:

• Transition Components
  - a source state
  - an event trigger
  - an action
  - a target state
Definitions

• Self-Transition

A self-transition is a transition whose source and target states are the same.
Definitions

• **Action**

  An *action* is an executable, atomic (with reference to the state machine) computation. Actions may include operations, the creation or destruction of other objects, or the sending of signals to other objects (events).
A Simple State Machine

Initial State

State 1

Event Trigger
Action

State 2

Event Trigger
Action

State 3

Self Transition

Final State
Entry and Exit Actions

<table>
<thead>
<tr>
<th>State Name</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Entry</th>
<th>Entry Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit</td>
<td>Exit Action</td>
</tr>
</tbody>
</table>
Definitions

• Substates

♦ A substate is a state that is nested in another state
♦ A state that has substates is called a composite state
♦ A state that has no substates is called a simple state
♦ Substates may be nested to any level
• Transitions to a composite state
  ◆ If a transition is to a composite state, the nested state machine must have an initial state
  ◆ If a transition is to a substate, the substate is entered after any entry action for the enclosing composite state is executed followed by any entry action for the substate
Advanced Transitions

Transitions from a composite state

- If a transition is from a substate within the composite state, any exit action for the substate is executed followed by any exit action for the enclosing composite state.

- A transition from the composite state may occur from any of the substates and takes precedence over any of the transitions for the current substate.
Advanced State Machine

Sub-States

Transition to a composite state
Transition from a composite state
Transition from a substate

State Name

State 0
Transition to a substate
State 1
State 2
State 3

Entry | Entry Action
Exit  | Exit Action
### State Table

<table>
<thead>
<tr>
<th>State</th>
<th>FirstIndex</th>
<th>LastIndex</th>
<th>EntryAction</th>
<th>ExitAction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Act1</td>
<td>Act2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>None</td>
<td>Act3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

### Transition Table

<table>
<thead>
<tr>
<th>Trans</th>
<th>Events</th>
<th>Mask</th>
<th>Action</th>
<th>NewState</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>E0 &amp; E1</td>
<td>E0</td>
<td>Act0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>E1</td>
<td>E1</td>
<td>None</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>
MODULE stateMachine - Sequential State Machine Package Declarations

DESCRIPTION:
   The stateMachine.h header file contains definitions of external interfaces to the state machine package

#define ssmMask(event) (1<<event)

enum {SSM_TERMINAL_INDEX = (unsigned short int)-1};
enum {SSM_NULL_ACTION = NULL};
enum {SSM_MAX_EVENT = 31};

typedef void *SsmId_t;
typedef int (*SsmAction_t)(const SsmId_t stateId, void const *context,
                           const unsigned short int action);
typedef void (*SsmDisplay_t)(char *text);
typedef unsigned long int SsmEventSet_t;

/* State table element */
typedef struct
{
    unsigned short int firstIndex;  /* First index in transition table */
    unsigned short int lastIndex;   /* Last index in transition table */
    unsigned short int inAction;    /* Entry action index */
    unsigned short int outAction;   /* Exit action index */
} SsmState_t;

/* Transition table element */
typedef struct
{
    SsmEventSet_t events;          /* Event value */
    SsmEventSet_t mask;            /* Event mask */
    unsigned short int action;     /* Action index */
    unsigned short int newState;   /* New state */
} SsmTransition_t;

extern SsmId_t ssmCreate(const unsigned short int initState,
                           const unsigned short int termState,
                           const unsigned short int maxIndex,
                           const SsmEventSet_t initEvents,
                           const SsmAction_t actionFunct,
                           void const *context,
                           SsmState_t (*stateTable)[],
                           SsmTransition_t (*transTable)[];
extern int ssmDelete(const SsmId_t stateId);
extern int ssmEventClear(const SsmId_t stateId,
                          const unsigned int event);
extern int ssmEventSet(const SsmId_t stateId,
                        const unsigned int event);
extern unsigned int ssmExecute(const SsmId_t stateId);
extern unsigned short int ssmCurStateGet(const SsmId_t stateId);
• Problems with UML notation
  
  ◆ When more than one transition from a state is enabled there is no method for specifying precedence
  
  ◆ For nested states there is no method for specifying precedence of the enclosing or enclosed state
Example: High Voltage Channel