Translating SCXML Statecharts to iUML-B State-machines

Karla Morris  :  Sandia National Laboratories, CA, USA
Colin Snook  :  University of Southampton, UK
Motivation

- Event-B provides verification by formal proof...
- ... but notation is restricted to simplify verification.
- Engineers are used to a richer notation..
- .. they may find the restrictions difficult to accept.
- iUML-B State-machines help but still close to Event-B.
- Can Harel style state-chart semantics be reconciled with iUML-B?
- We investigate a translation from SCXML state-charts to iUML-B state-machines (and hence to Event-B).
SCXML

- State Chart XML:
  - State Machine Notation for Control Abstraction
- XML notation
- Harel Statecharts
- Executable (via simulator tools)
- Related to CCXML Call Control XML, event-based telephony
<?xml version="1.0"?>
<scxml xmlns="http://www.w3.org/2005/07/scxml"
version="1.0"
datamodel="ecmascript"
initial="off">
    <!-- trivial 5 second microwave oven example -->
    <datamodel>
        <data id="cook_time" expr="5"/>
        <data id="door_closed" expr="true"/>
        <data id="timer" expr="0"/>
    </datamodel>
    <state id="off">
        <!-- off state -->
        <transition event="turn.on" target="on"/>
    </state>
    <state id="on">
        <!-- on/pause state -->
        <initial>
            <transition target="idle"/>
        </initial>
        <transition event="turn.off" target="off"/>
        <transition cond="timer &gt;= cook_time" target="off"/>
        <state id="idle">
            <!-- default immediate transition if door is shut -->
            <transition cond="door_closed" target="cooking"/>
            <transition event="door.close" target="cooking">
                <assign location="door_closed" expr="true"/>
            </transition>
            <!-- start cooking -->
        </state>
        <state id="cooking">
            <!-- a 'time' event is seen once a second -->
            <transition event="door.open" target="idle">
                <assign location="door_closed" expr="false"/>
            </transition>
            <transition event="time">
                <assign location="timer" expr="timer + 1"/>
            </transition>
        </state>
    </state>
</scxml>
red = external trigger event
[black] = conditional, no trigger

Rodin Workshop 2016 – Linz, 23rd May
iUML-B Statemachines

EVENTS

e1 = WHEN <in S1> THEN <becomes S2> END

where, <in S1> and <becomes S2> depend on the data that represents state
iUML-B Statemachines

Guard: shield = closed

Guard: machine_onState ≠ ACTIVE

Rodin Workshop 2016 – Linz, 23rd May
Similarities

- Hierarchical nested state-charts

- Transitions with
  - Conditions / Guards
  - Actions

- States can have Entry and Exit Actions
  - (use with care in iUML-B)
Differences

• Event-B has..
  – Refinement
  – Invariants

• SCXML has..
  – External Trigger events
    • Hence transitions do not have a name/label
  – Sequential actions
  – Run to Completion – Big step/little step
SCXML Extensions

• XML tools allow new meta-model ‘namespaces’ to be introduced.
  – Existing SCXML tools will ignore them

• Needed in order to support:
  – Refinement levels (new attribute $<\text{iumlb:refinement} ...>$)
  – Invariants (new element $<\text{iumlb:invariant} ...>$)
  – Guards (new element $<\text{iumlb:guard} ...>$)
### SCXML Extension Attributes

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Meaning</th>
<th>Allowed Parents</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>string used as the name of an Event-B event elaborated by the generated i-UMl-B</td>
<td>scxml:transition</td>
</tr>
<tr>
<td>type</td>
<td>string used as the membership set for the Event-B variable generated from the parent data element</td>
<td>scxml:data</td>
</tr>
<tr>
<td>name</td>
<td>string used for the name or label of a generated iUML-B element</td>
<td>iuml:b:invariant, iuml:b:guard</td>
</tr>
<tr>
<td>predicate</td>
<td>string used for the predicate of a guard or invariant</td>
<td>iuml:b:invariant, iuml:b:guard</td>
</tr>
<tr>
<td>derived</td>
<td>boolean indicating that the guard is a theorem (default to false)</td>
<td>iuml:b:invariant, iuml:b:guard</td>
</tr>
</tbody>
</table>
Example extended SCXML
(extensions are the bits in red)

<datamodel iumlb:refinement="2">
  <data expr="false" id="Gate_In.Block" iumlb:type="BOOL" />
</datamodel>
<!-- Other model details -->
<state id="BLOCKED">
  <transition cond="[On_In.CardAccept==true]" target="UNBLOCKED">
    <iumlb:guard name="gd1" predicate="On_In.CardAccept==true" refinement="2"/>
    <assign expr="true" location="Gate_In.Block" iumlb:refinement="3"/>
  </transition>
  <onentry>
    <assign expr="true" location="Gate_In.Block"/>
    <assign expr="false" location="On_In.Reset"/>
  </onentry>
  <onexit>
    <assign expr="false" location="Gate_In.Block"/>
  </onexit>
  <iumlb:invariant predicate="Gate_In.Block == TRUE" name="GateCondition"/>
</state>
Initial translation supports..

- Data models
- Hierarchical nested statemachines
- Parallel Statemachines
- ‘When’ Transitions (label)
- Transition parameters, guards and actions
- Invariants
- Initial and Final states
- Refinement (superposition only)
Diagram of SCXML

Rodin Workshop 2016 – Linz, 23rd May
Example – generated iUML-B
Next steps

• Try modelling the run to completion semantics

• E.g. trigger events create a token,
  – A new token can only be consumed when no transitions are enabled

• Try enforcing transition run-to-completion sequences

• *Still omit sequencing of actions*
Enhance iUML-B to support triggers

- iUML-B Statemachines will own a collection of triggers.
  - Each trigger will generate an Event-B BOOL variable.
    - (Note simplification of SCXML, which permits several triggers of a kind to be queued).
  - Transitions may reference a trigger.
    - The reference will generate a guard, $<\text{trigger variable}> = \text{TRUE}$
    - And an action $<\text{trigger variable}> := \text{FALSE}$.
  - Transitions may own a collection of ‘Raise’ actions that reference an internal trigger.
    - This will generate an action $<\text{trigger variable}> := \text{TRUE}$.
  - Triggers may be designated as external.
    - An interface event will be generated to create a new trigger ($<\text{trigger variable}> := \text{TRUE}$)
    - when it has been consumed ($<\text{trigger variable}> = \text{FALSE}$) and
    - No transitions are enabled. (run to completion)

- A partial ‘run-to-completion’ semantics will be introduced by disabling all interface events while any external or internal transition is enabled.
External Trigger Event

Old trigger has been consumed

No transitions enabled

Raise new trigger

Rodin Workshop 2016 – Linz, 23rd May
Triggered transition

The trigger guard

Raise an internal trigger

Consume the external trigger
Conclusions

- Strong motivation from engineers

- Difficult to reconcile semantic differences
  - Run-to-completion, Sequential execution

- We adopt a compromise
  - Support what we can
    - Add extensions where necessary
  - Otherwise, restrict SCXML
Thank you

Questions?