Sandia National Laboratories

UNIVERSITY OF Southampton
School of Electronics
and Computer Science

Reconciling SCXML Statechart Representations and Event-B Lower Level Semantics

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

SCXML

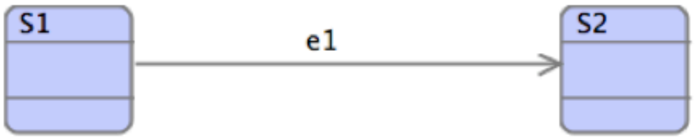
```
<?xml version="1.0"?>
<scxml
xmlns="http://www.w3.org/2005/07/scxml"
   version="1.0"
   datamodel="ecmascript"
   initial="off">

 <!--  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">
  <initial>
    <transition target="idle"/>
  </initial>
  <!-- on/pause state -->
  <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"/>
    <!-- start cooking -->
   </transition>
  </state>
  <state id="cooking">
   <transition event="door.open"  target="idle">
    <assign location="door_closed"  expr="false"/>
   </transition>
   <!-- a 'time' event is seen once a second  -->
   <transition event="time">
    <assign location="timer" expr="timer + 1"/>
   </transition>
  </state>
 </state>
</scxml>
```

iUML-B Statemachines

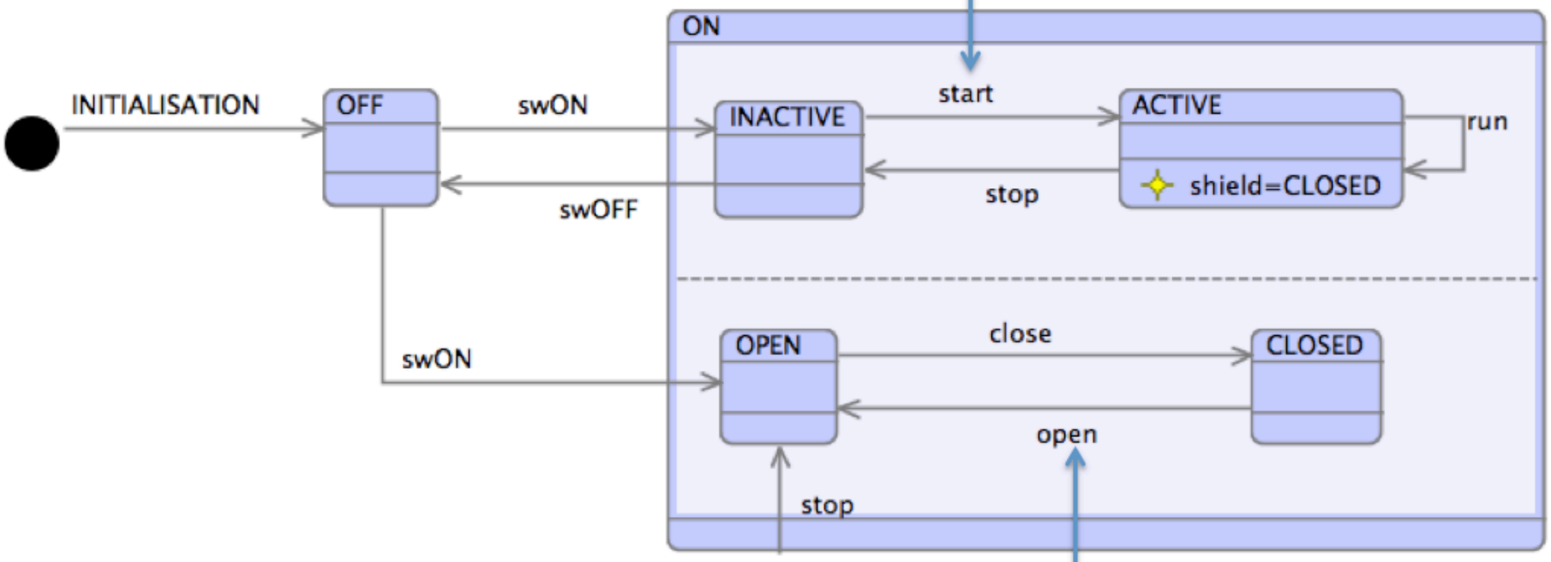


```
EVENTS
        e1 ≙ WHEN <in S1> THEN <becomes S2> END
```

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

iUML-B Statemachines

guard: shield = closed



guard: machine_onState ≠ ACTIVE

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 <iumlb:refinement …>)
  - Invariants (new element <iumlb:invariant …>)
  - Guards (new element <iumlb:guard …>)

SCXML Extension Attributes

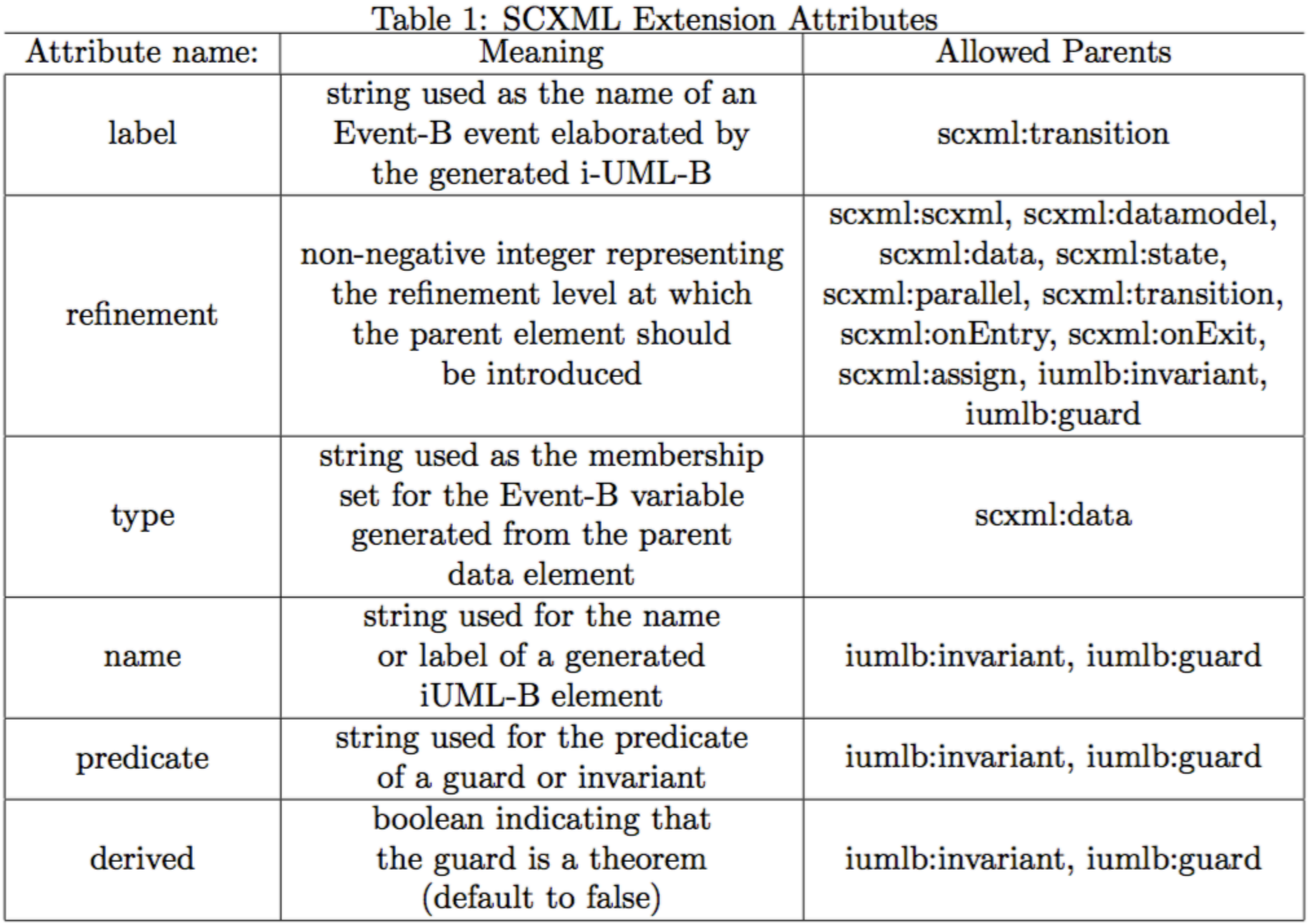


Table 1: SCXML Extension Attributes

Attribute name:	Meaning	Allowed Parents
label	string used as the name of an Event-B event elaborated by the generated i-UML-B	scxml:transition
refinement	non-negative integer representing the refinement level at which the parent element should be introduced	scxml:scxml, scxml:datamodel, scxml:data, scxml:state, scxml:parallel, scxml:transition, scxml:onEntry, scxml:onExit, scxml:assign, iumlb:invariant, iumlb:guard
type	string used as the membership set for the Event-B variable generated from the parent data element	scxml:data
name	string used for the name or label of a generated iUML-B element	iumlb:invariant, iumlb:guard
predicate	string used for the predicate of a guard or invariant	iumlb:invariant, iumlb:guard
derived	boolean indicating that the guard is a theorem (default to false)	iumlb:invariant, iumlb:guard

Example extended SCXML

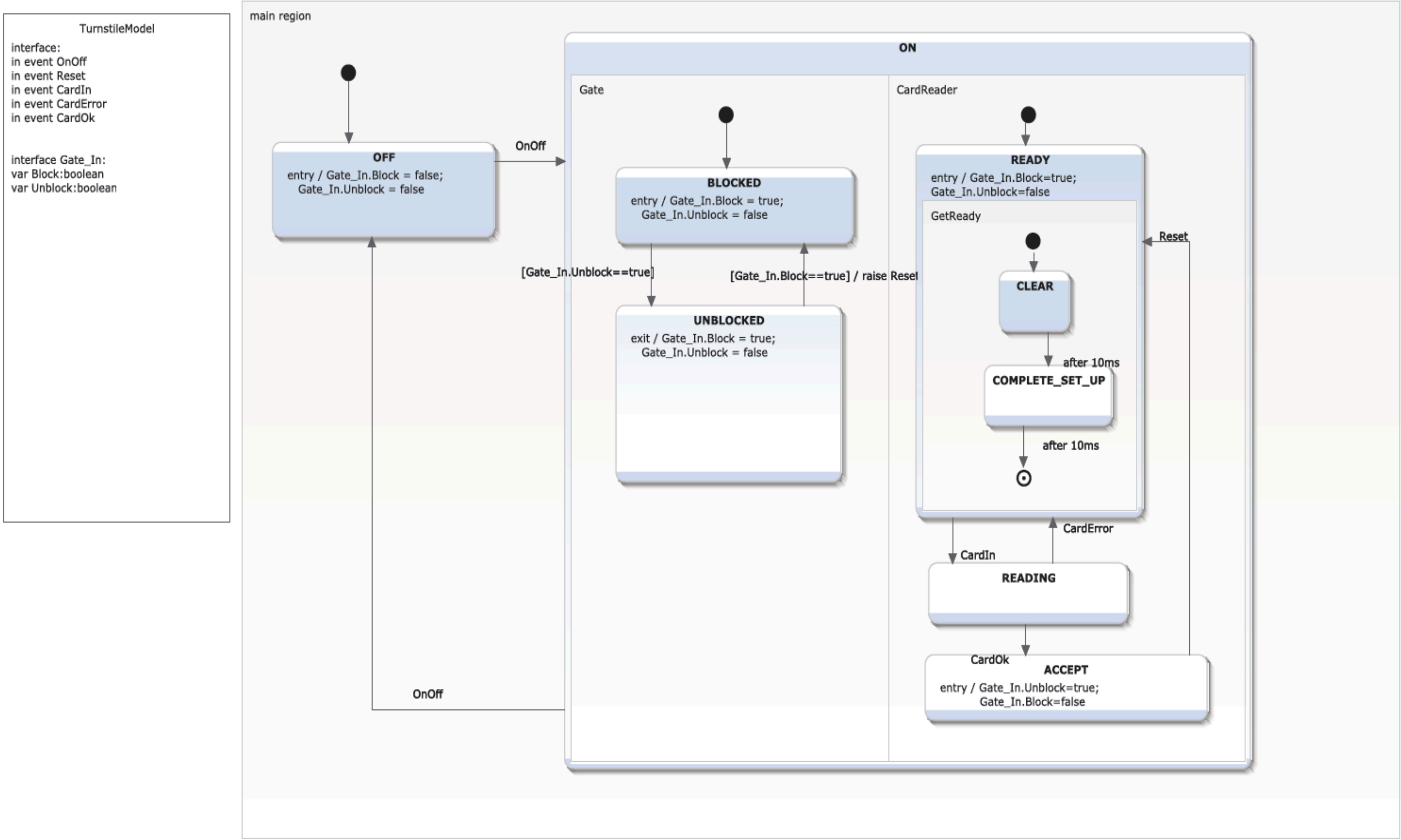
(extensions are captured in red)

```
<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
- Parrallel Statemachines
- ‘When’ Transitions (label)
- Transition parameters, guards and actions
- Invariants
- Initial and Final states
- Refinement (superposition only)

Diagram of SCXML

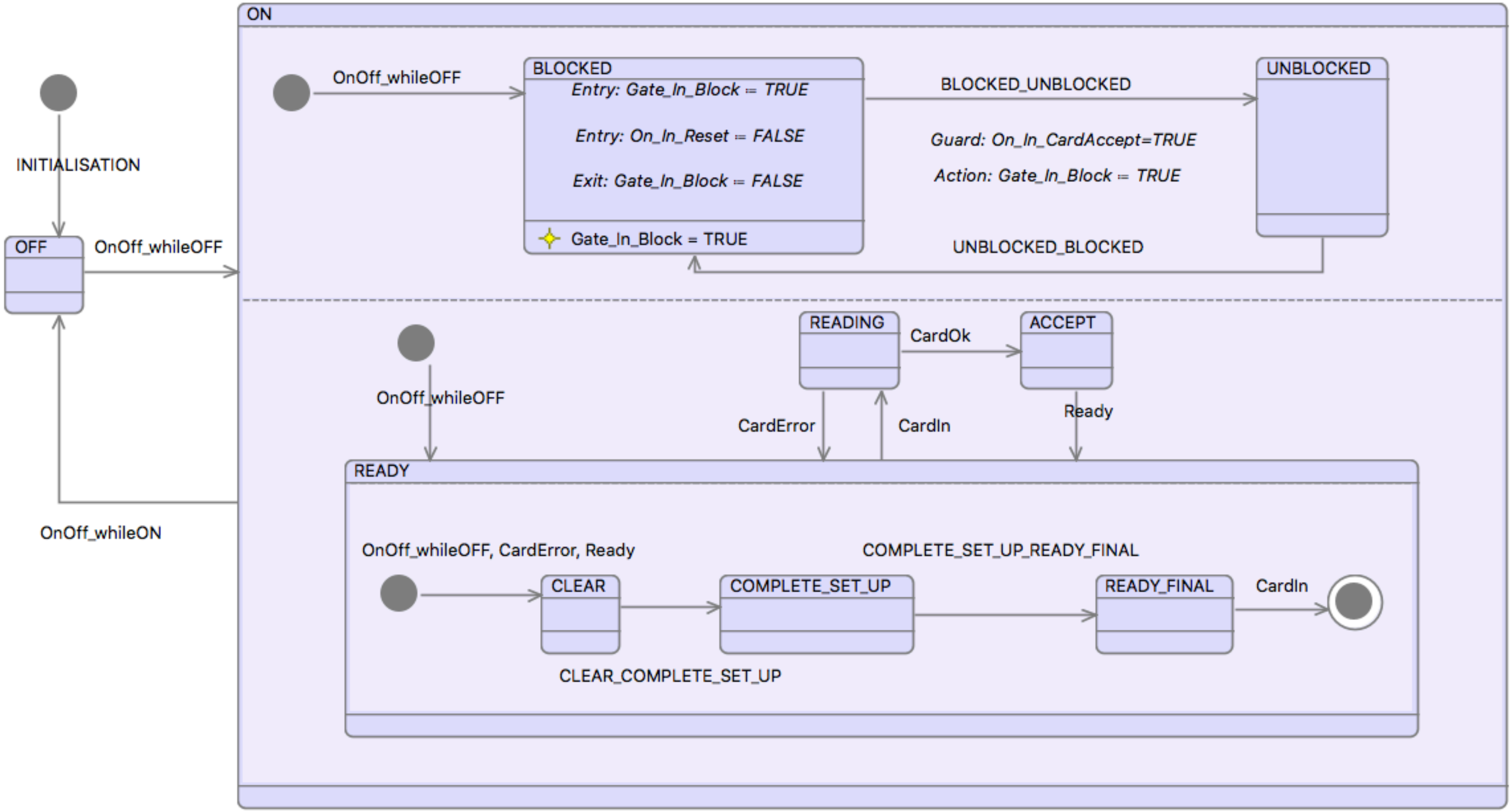


TurnstileModel

```
interface:
in event OnOff
in event Reset
in event CardIn
in event CardError
in event CardOk

interface Gate_In:
var Block:boolean
var Unblock:boolean
```

Example – generated iUML-B



Next steps

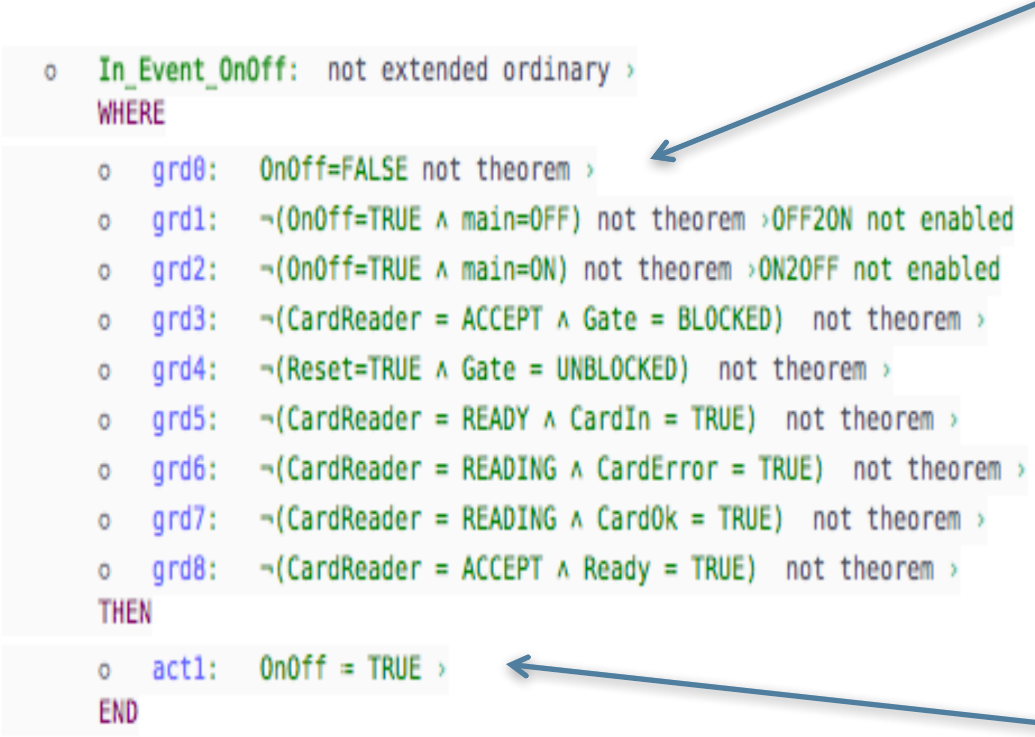
- Try modelling the run to completion semantics
- E.g. trigger events create a token,
  - A new token can only be consumed when no transitions are enabled
- Try enforcing transition run-to-completion sequences
- *Still omit sequencing of actions*

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, <*trigger variable*> = *TRUE*
    - And an action <*trigger variable*> := *FALSE.*
  - Transitions may own a collection of 'Raise' actions that reference an internal trigger.
    - This will generate an action <*trigger variable*> := *TRUE.*
  - Transitions may be designated as external.
    - An interface event will be generated to create a new trigger *(<trigger variable> := TRUE)*
    - when it has been consumed *(<trigger variable> = 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



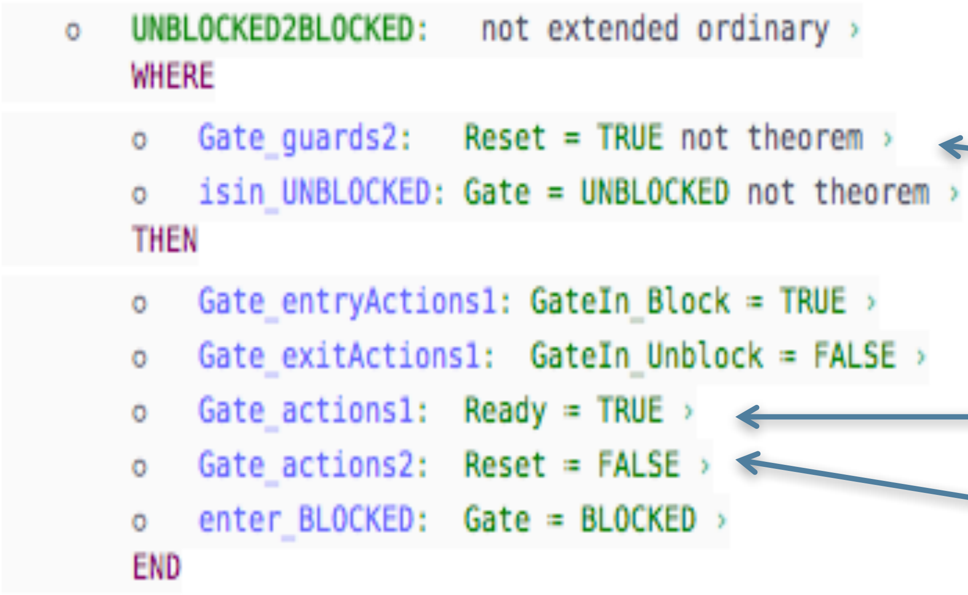
```
o  In_Event_OnOff:  not extended ordinary ›
   WHERE
      o  grd0:   OnOff=FALSE not theorem ›
      o  grd1:   ¬(OnOff=TRUE ∧ main=OFF) not theorem ›OFF2ON not enabled
      o  grd2:   ¬(OnOff=TRUE ∧ main=ON) not theorem ›ON2OFF not enabled
      o  grd3:   ¬(CardReader = ACCEPT ∧ Gate = BLOCKED)  not theorem ›
      o  grd4:   ¬(Reset=TRUE ∧ Gate = UNBLOCKED)  not theorem ›
      o  grd5:   ¬(CardReader = READY ∧ CardIn = TRUE)  not theorem ›
      o  grd6:   ¬(CardReader = READING ∧ CardError = TRUE)  not theorem ›
      o  grd7:   ¬(CardReader = READING ∧ CardOk = TRUE)  not theorem ›
      o  grd8:   ¬(CardReader = ACCEPT ∧ Ready = TRUE)  not theorem ›
   THEN
      o  act1:   OnOff ≔ TRUE ›
   END
```

Old trigger has been consumed

No transitions enabled

Raise new trigger

Triggered transition



```
o  UNBLOCKED2BLOCKED:   not extended ordinary ›
   WHERE
      o  Gate_guards2:   Reset = TRUE not theorem ›
      o  isin_UNBLOCKED: Gate = UNBLOCKED not theorem ›
   THEN
      o  Gate_entryActions1: GateIn_Block = TRUE ›
      o  Gate_exitActions1:  GateIn_Unblock = FALSE ›
      o  Gate_actions1:  Ready = TRUE ›
      o  Gate_actions2:  Reset = FALSE ›
      o  enter_BLOCKED:  Gate = BLOCKED ›
   END
```

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

Sandia
National
Laboratories

UNIVERSITY OF
Southampton
School of Electronics
and Computer Science

Thank you

Questions?