Refinement of SCXML state-charts via translation to Event-B

C. Snook[1], K.Morris[2], M.Butler[1], and R.Armstrong[2]

[1] University of Southampton, Southampton, United Kingdom
cfs@ecs.soton.ac.uk
[2] Sandia National Laboratories, Livermore, California, U.S.A.
knmorri@sandia.gov

Abstract. State-chart modelling notations, such as SCXML, with so-called 'run to completion' semantics and simulation tools for validation, are popular with engineers for designing machines. However, they do not support refinement and they lack formal verification methods and tools. Properties concerning the synchronisation between different parts of a machine may be difficult to verify for all scenarios. Event-B, on the other hand, is based on refinement from an initial abstraction and is designed to make formal verification by automatic theorem provers feasible, obviating the need for instantiation and testing. We would like to combine the best of both approaches by incorporating a notion of refinement, similar to that of Event-B, into SCXML and leveraging Event-B's tool support for proof. We describe the some pitfalls in translating 'run to completion' models to Event-B refinements, suggest a solution and propose extensions to the SCXML syntax to describe refinements. We illustrate the approach using our prototype translation tools and show by example, how a synchronisation property between parallel state-charts can be automatically proven at an incomplete refinement level by translation into Event-B.

Keywords: SCXML, State-charts, Event-B, iUML-B, refinement

1 Introduction

This is the introduction...

2 Background

2.1 SCXML

SCXML is a modelling language based on Harel state-charts with facilities for adding data elements that are manipulated by transition actions and used in conditions for their firing. SCXML follows the usual 'run to completion' semantics of such state-chart languages, where trigger events may be needed to enable transitions. Trigger events are queued when they are raised and then one is dequeued and consumed by firing all the transitions that it enables, followed by

any (un-triggered) transitions that then become enabled due to the change of state caused by the initial transition firing. This is repeated until no transitions are enabled and then the next trigger is de-queued and consumed. There are two kinds of triggers: internal triggers are raised by transitions and external triggers are raised by the environment (spontaneously as far as our model is concerned). An external trigger may only be consumed when the internal trigger queue has been emptied.

```
while running:
  while run2completion = false
    if untriggered_enabled
      execute(untriggered())
    elseif IQ /= {}
      execute(internal(IQ.dequeue))
    else
      run2completion = true
    endif
  endwhile
  if EQ /= {}
    execute(EQ.dequeue)
    run2completion = false
  endif
endwhile
```

Listing 1: Pseudocode for 'run to completion'

2.2 Event-B

Colin: we can copy from a previous paper

Overview of Event-B... semantics, refinement, proof obligations, tools..

2.3 iUML-B State-machines

Colin: we can copy from a previous paper

Overview of iUML-B State-machines... semantics from Event-B, translation...

2.4 SecBot example

Colin: introduce the secbot example showing how we would like to develop it in refinements

Secbot example in refinements..

3 Discussion

This is the discussion... Compare the behaviour of a visually similar statechart (SecBot?) in iUML-B and SCXML Based on their Event-B translations Using the theorem prover to test whether they are equivalent (they are not) Using the model checker to compare traces Using LTL to show that certain temporal properties are not achieved by the iUML-B version.

Early attempts 1) Simple next step based on negated guards negation is weakened 2) Engine improves the R2C semantics but still suffers from the negated guards problem (refinement of the user model kinda works simulations at the user level).

Colin: IF POSSIBLE? use the secbot example to illustrate the problem of guard stengthening on refinement

The solution: 3) Transition combinations approach works because there is always one event to completion. Invariants as well as simulations. a. Explosion mitigated by mutual exclusions same trigger different state-chart regions.

What can we do with it? Why is RTC needed? Refinement in RTC

4 Tooling

This is the section about our plug-in..

Colin: Things to cover: EMF meta-model, translation tool, anything else?

5 Example

The SecBot example : pics from slides

Note why it would not be a refinement

6 Related Work

This is the related work...

7 Conclusion

This is the conclusion...

Colin: maybe consider the note below (from our previous plans for a paper)
Note: Things to highlight with the choice of example 1. Look at a system that is better model with SCXML run to completion semantics than iUML-B semantics 2. Look at how you can check for violations of refinement in a SCXML model construction 3. Look at the sort of invariant properties you can verify about a SCXML model

All data supporting this study are openly available from the University of Southampton repository at http://doi.org/10.????/SOTON/D0???

References