Statecharts Code Synthesis

Reehan Shaikh

MSDL Summer Presentations
School of Computer Science
McGill University

August 27th, 2009

ubuntu

References

- COMP – 304, School of Computer Science, McGill University
- Harel, D. and Naamad, A. The STATEMATE semantics of statecharts. ACM Trans. Softw. Eng. Methodol. 5, 4 (Oct. 1996), 293-333.
- Harel, D. and Kugler, H. The Rhapsody semantics of statecharts (or, on the executable core of the UML). Lecture Notes in Computer Science, vol. 3147. Springer. pp. 325-354.

Outline

- Introduction
- Requirements of the DigitalWatch
- Solution to the DigitalWatch
- Code synthesis
  - Class diagram
  - Statecharts
  - DEVS
  - Demo
- Conclusion + Future work

Introduction

- What: Goals are:
  - To synthesize code from a statechart
  - Use this code to model a behaviourally equivalent DEVS system
- Why: allows for truly multi-formalism modelling since DEVS can be used as a semantic domain for DAEs, Motif, etc...
- How: Syntax check and compile (to nice, readable, executable code)

Requirements

- We should know these like the back of our hands by now, after taking COMP 304/522/763.
- Just as a recap (for visitors):
  - http://msdl.cs.mcgill.ca/people/hv/teaching/MS/assignments/assignment3
- Given a set of requirements, an API and some pre-defined events, create a statechart that meets those requirements

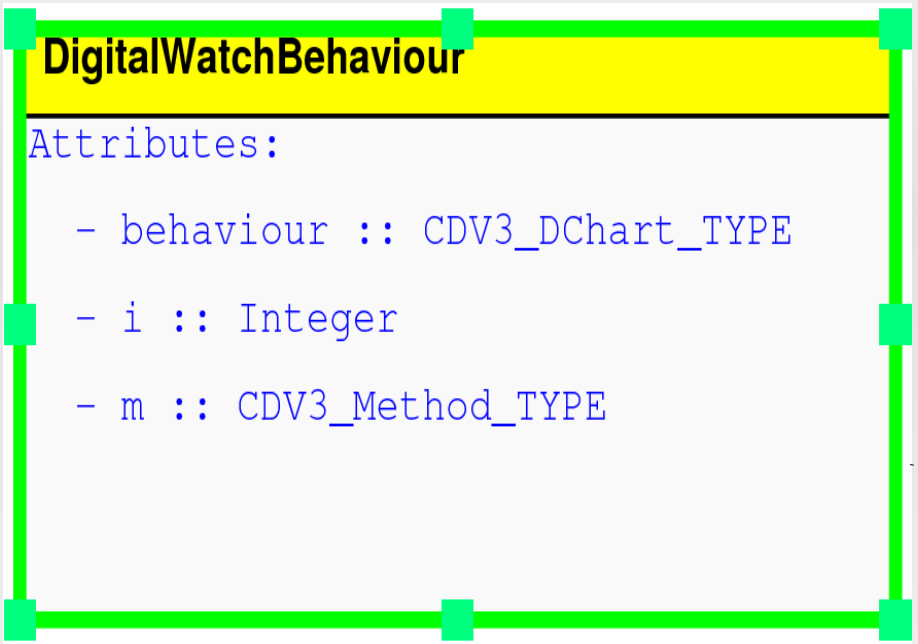
Class diagrams

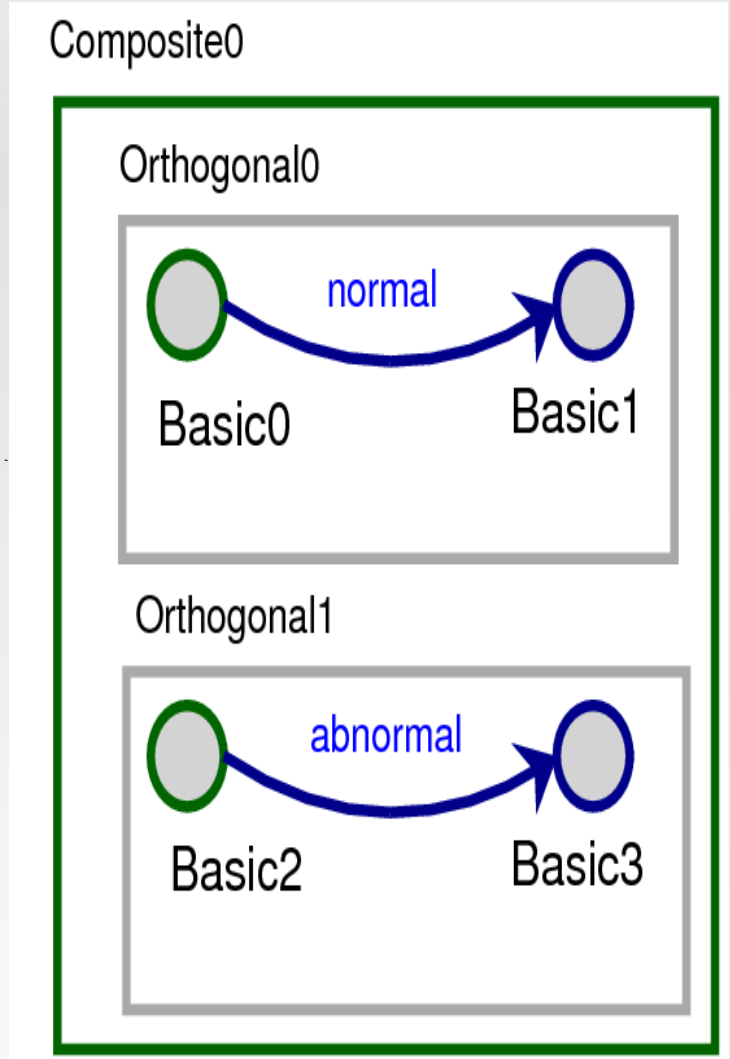
- Given a class diagram describing the system structure, where each class may have an associated statechart describing its behaviour, we must analyze and compile this model
- First, check validity of class diagram as a whole (at least one class, unique names, single default class)
- Second, process each association and inheritance
- If there were no errors, process each class. If the class has no errors and has a statechart, process the statechart

Statecharts

- Verify that there are no empty components
- Verify that each component doesn't contain two or more children with identical names
- Each component must contain a single default state
- Verify correct statechart semantics
- Calculate transition data (LCA, enter/exit actions)
- Synthesize code

CD + SC example





MSDL

ubuntu

CD + SC example

- So, what will the synthesize code look like?
- class DigitalWatchBehaviour:

```
  Basic0 = 0
  Basic1 = 1
  …
  def init(self,controller,loopMax)
    "initialize statechart variables..."
    i = 0 "user defined variable"
  def enterActionBasic0(self)
    "enter action for Basic0"
```

CD + SC example

- def exitActionBasic1(self)
  "exit action for Basic1"

  …

  def m(self,paramters)
  "body of user defined method"
  def event(self,event,time,*args)
  "add event to object's event queue"
  def getEarliestEvent(self)
  "return absolute time"

  ...

CD + SC example

- And now the core behaviour:

```
def transition(self,event,parameters)
  if currentState == Basic0
    if event == "normal"
      self.exitActionBasic0( )
      print "trigger normal"
      self.enterActionBasic1( )
  if currentState == Basic2
    if event == "abnormal"
      self.exitActionBasic2( )
      print "trigger abnormal"
      self.enterActionBasic3( )
```

CD + SC example

- Two helper functions
  - microstep to calculate current events to be processed and continuously call transition until stable, unchanged state
  - step to calculate AFTER events and continously call microstep
- Now, how would an application run, suppose there were many classes and statecharts?

DEVS Controller

- We need a controller to control the various activities. We will discuss the DEVS controller in particular.
- 4 functions to be filled
  - External transition
  - Internal transition
  - Output function
  - Time advance

DEVS Controller

- Each controller needs to keep track of all object instances that entail a statechart
- External transition
  - Interrupts are given to a DEVS model via ports
  - An external event is packaged as the event "portName:event" for the statechart and each statechart's step function is called
- Internal transition
  - Each statechart's step function is called

DEVS Controller

- Output function
  - If a statechart puts an output event, of the form "portName:event", on the controllers output queue, it is taken care of when the output function is called
- Time advance
  - Get each statechart's earliest event time
  - The earliest time of those is returned
  - Otherwise INFINITY is returned, meaning statecharts are finished processing or awaiting external interrupt

Demo + Conclusion

- So, we synthesized code for a statechart
- Essentially, we used DEVS as a particular controller, but other controllers are available
  - Real-time or simulated time
  - Time-sliced or event-scheduled
- Future work
  - Add a mechanisn to generate Tkinter events
  - Finish bug testing for the above controllers

!!!Thank you!!!

???Questions???