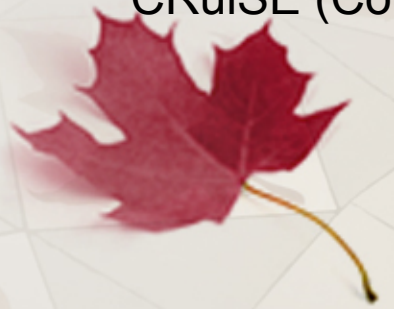
War Stories and Advances in Model-Directed Tracing

Timothy C. Lethbridge
Miguel Garzon
Hamoud Aljamaan

CRuiSE (Complexity Reduction in Software Engineering) Research Group
University of Ottawa

Dec 9, 2011



uOttawa

http://www.site.uottawa.ca/~tcl

www.uOttawa.ca

Summary of work accomplished since mid-year 2011 -1

Injection of trace directives into code generation for UML attributes and basic state machines

- I will demonstrate shortly
- Only a subset of the intended functionality so far

Improvements to generic plugin capability for different tracing tools

- Not yet working with LTTNG, but will be soon hopefully
- Generic plugin outputs to stdout

Summary of work accomplished since mid-year 2011 -2

Code generation for state machines improved

- Needed to overcome problems faced with Papyrus

Incremental reverse engineering (Mario very interested in this)

- Detect constructs in Java and convert to model

Key people:

- Hamoud Aljamaan – MOTL
- Miguel Garzon – Incremental reverse engineering
- Andrew Forward – Infrastructure
- Omar Badreddin – PhD nearing completion (state machines)
- Sultan Eid – Masters (C++ generation/instrumentation)

Papyrus

Not a nice tool for working with state machines

- Awkward to edit and no 'help'
  - Here's a quick demo …

Code generation insufficient for our needs

- But this is a pre-requisite for injecting tracing directives

Versions of papyrus still have backward compatibility issues

- Previous example models and code won't work

We're still waiting to integrate fully into it

- Hope to do so eventually

State machines as models

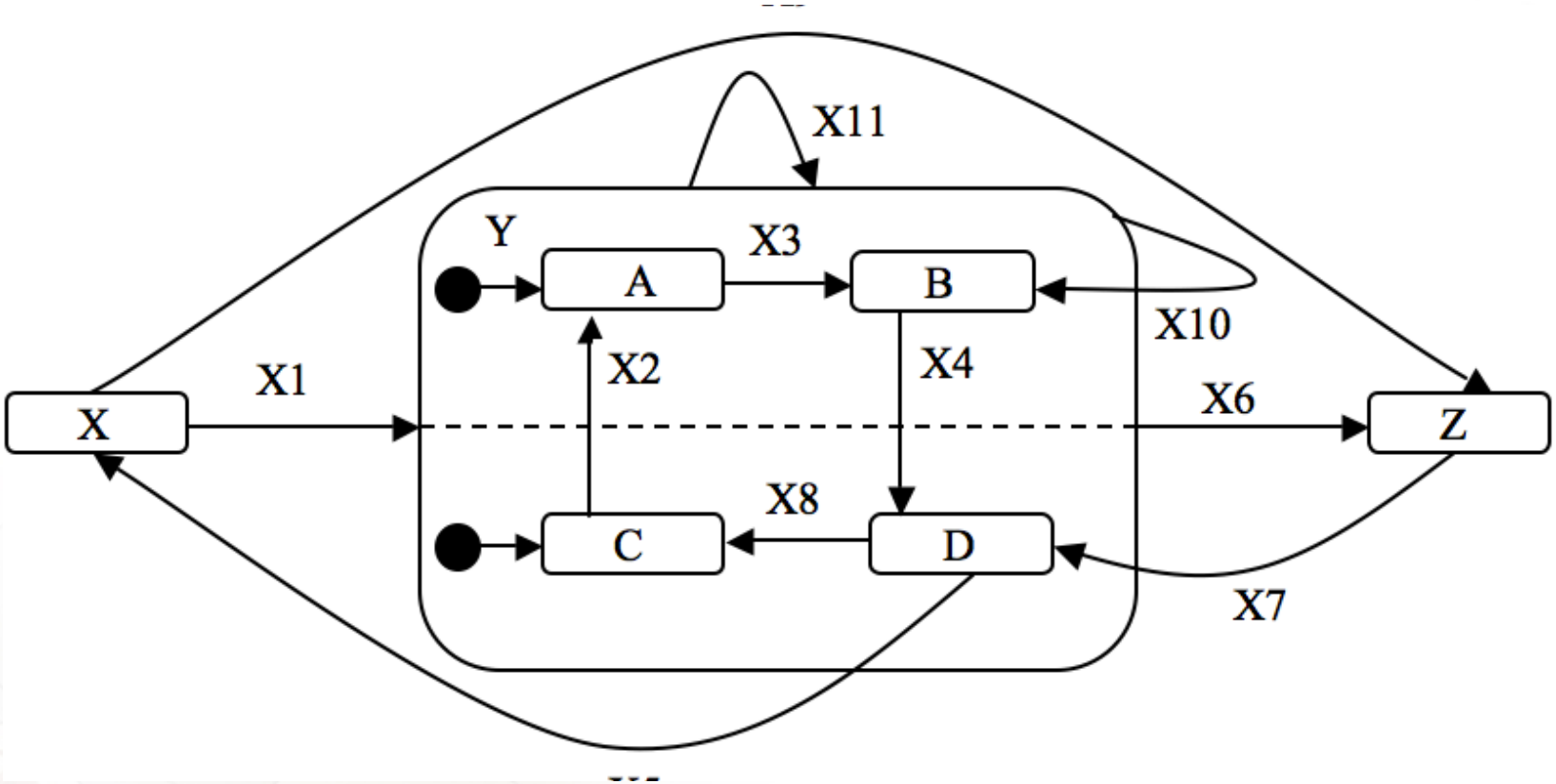
Every variable can conceptually be considered a state machine

- Every possible value of the variable is a state
- System behaviour determined by the set of states

To better model systems, we are interested in detecting and describing patterns of behaviour that can be described by small sets of states

- Finite state machines
- Events and guards govern state change

State machine challenge 1: Accounting for all transition types

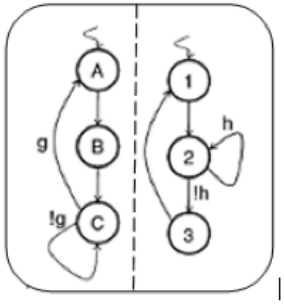


State machine challenge 2: Recursion of nesting and multiple regions

- Accounting for
  - Correctly going to the right start state(s) when transitioning into a state machine with nesting
  - Transitioning from an arbitrary state to an arbitrary state
    - ‘backing’ out of just the right number of levels
      - » With correct exit actions performed
    - Entering just the right number of new states
      - » With correct entry actions performed
- We have not found an open-source code generator that gets this fully right
  - Bugs in our own one only just ironed out

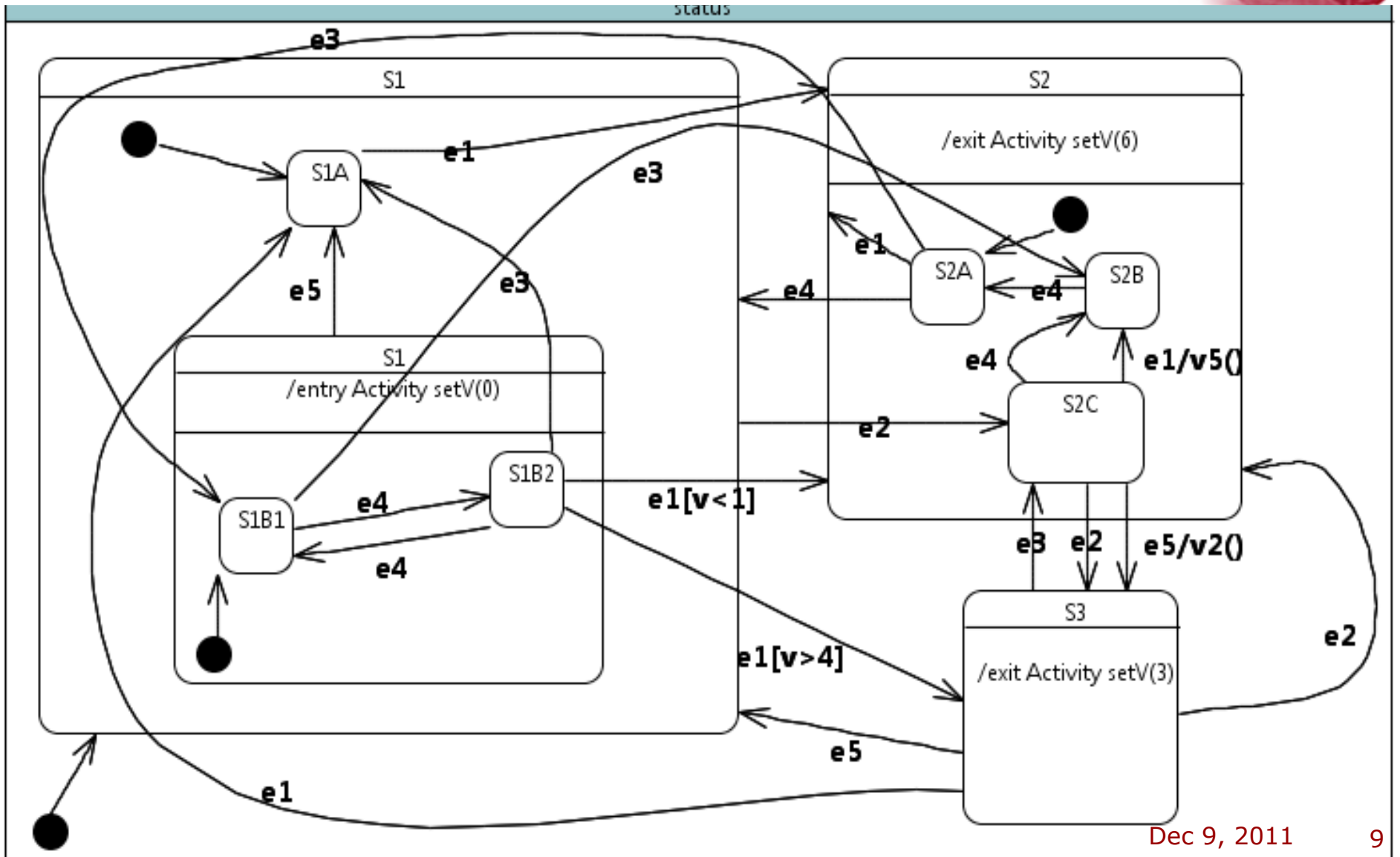
State machine challenge 3: The 'Flattening problem'

- The 'set of real' states, and hence generated code, from a model can grow
  - With 2 concurrent regions that have 3 states each, there would be 9 'real states'
  - With 5 concurrent regions with 5 states each
  - 3125



- Our solution: Generate multiple separate state machines and use a 'null state' when one is not active
  - The previous example would have
    - X-Y-Z, Null-A-B, Null-C-D
  - The 5x5 example would have 25 states
  - Events have to be designed to cause simultaneous transitions in multiple state machines

An example ‘status’ state machine we will use for demonstration



Trace directives in state machines

Trace all state machines at all levels

- trace status
  - By default shows events, entry, exit, actions

Trace just one submachine (recursing to all levels)

- trace S1
- trace S1B1

Trace just one submachine (just showing one level)

- trace S1 level(1)

Trace an event (regardless of state machine)

- trace e1

Trace an event in a state machine

- trace e1 in S2A

Trace directives in state machines 2

Trace only entry into a state

- trace entry S2

Control what to record when a directive is executed

- trace S1 record v

Combining the above with other aspects of MOTL

- trace status after v<3

Trace cases and other MOTL elements

Demonstrations

All demonstrated from the command line

cd ~/tmp/papyrus/newsmtest

bbedit sm.model gentrace.cmd trace.motl

ur sm.model gentrace.cmd trace.motl

To open the generated code in Java

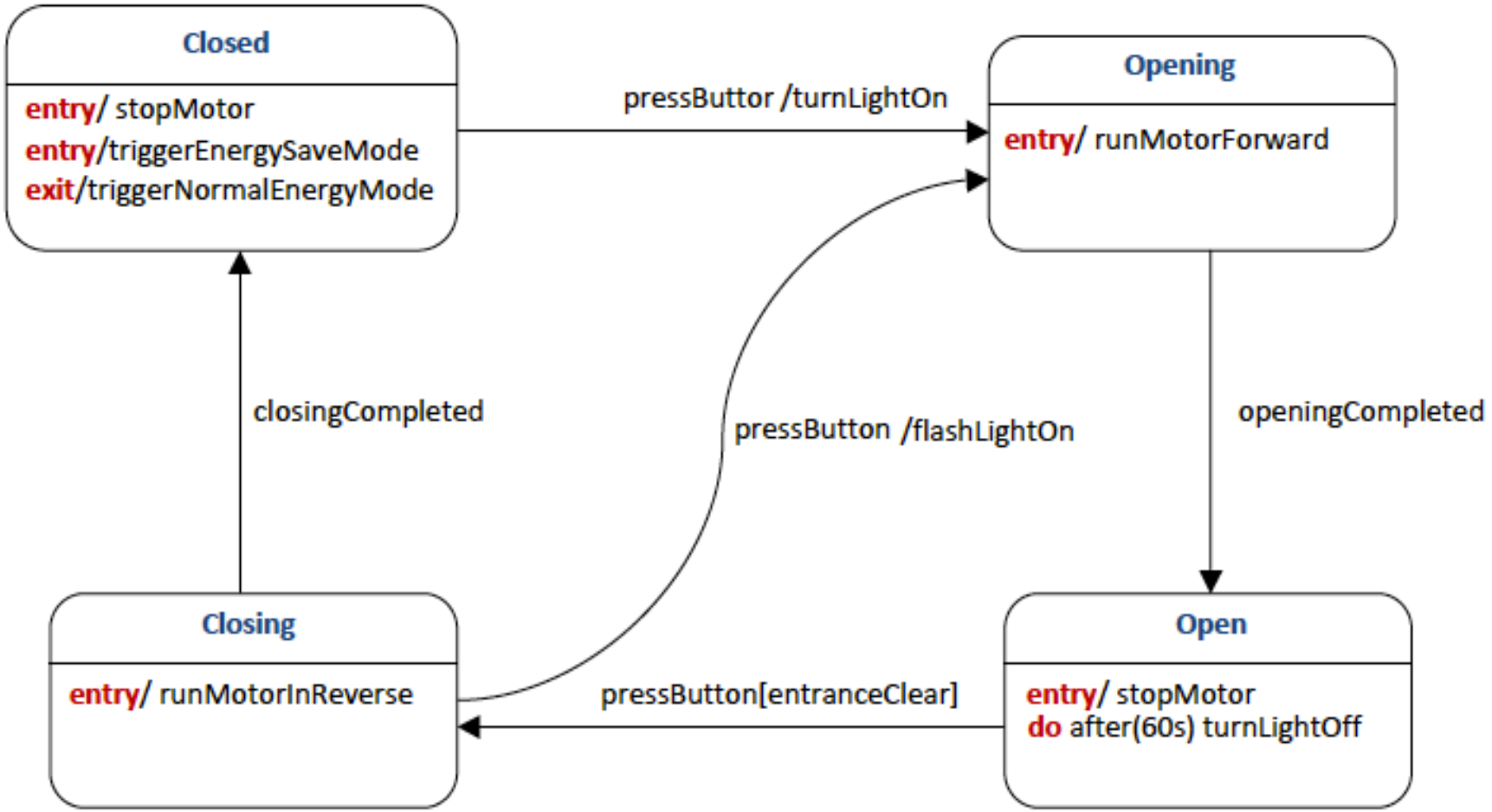
mate test/StateMachineTest.java

Another example with diagram on next page

cd ../garage; mate trace.motl

ur garage.model gentrace.cmd trace.motl

Another example



Planned work 1
for the next few months

Trace generation

- Complete the code generation for the MOTL language
  - Limited-time directives
    - trace x for n
      - » Start a counter at the first trace and stop tracing after n hits
    - trace x during t
      - » Record the ms time counter at the first trace and stop tracing when t ms has passed
    - trace x period t
      - » Record the ms time counter at the first trace; then trace again only at the next match after t ms has passed

Planned work 2

Complete the C++ generation

- Using the same templates as Java
- The student assigned this task lacked expertise so it is taking longer than hoped

Finish the LTTNG trace generation plugin

Papyrus Integration

Planned work 2

Gather traces and display paths in the model

Experiment with real systems

Experiment with the benefits for real users

Incremental reverse engineering source code to UML models