

AFFIRM

Quarterly Meeting

November 7, 2016

Years 1 & 2

- Determining what abstractions are appropriate and necessary in the ADSL
- Exploration of the case studies, primarily OM(1), WBS, and BRAIN
- Experiments toward translating system specifications in an architectural DSL into transition system models.
- Converging on a suitable modeling framework (calendar automata, clock, and fault model)

Year 3 Plans

- Build a prototype SAL/Sally backend for the ADSL:
 - translation of expression language and message passing semantics
 - configurable hybrid fault model
 - generation of framework specific lemmas (e.g. calendar lemmas)
 - specification of properties
 - generation of observers and abstract state machines
- Specify our case studies in terms of the prototype ADSL and translator
 - OM(1)
 - WBS
 - Multi-level system: BRAIN, TTE, ...
- ...

Year 3 Plans

- Build a prototype SAL/Sally backend for the ADSL:
 - ☑ translation of expression language and message passing semantics
 - ☑ configurable hybrid fault model
 - generation of framework specific lemmas (e.g. calendar lemmas)
 - specification of properties
 - generation of observers and abstract state machines
- Specify our case studies in terms of the prototype ADSL and translator
 - ☑ OM(1)
 - WBS
 - Multi-level system: BRAIN, TTE, ...
- ...

ADSL Workbench Prototype

We've been working towards a prototype workbench with the following features:

- Architectural DSL
 - Haskell eDSL
 - simple, intuitive syntax
 - well-defined semantics (e.g. in terms of Petri net semantics)
 - Flexible and expressive for representing distributed, fault-tolerant systems
- C code generation
 - simple and fairly generic code generation strategy
 - hard real-time embedded systems
 - POSIX systems
- SAL/Sally model generation
 - Fault model: configurable at translation time
 - Safety properties: synchronous observers and abstractions can be specified at a high level
- AADL generation

DEMO

```
vagrant@contrib-jessie:~$ sally -v 1 --engine kind --kind-max 1 A3.mcmt  
[2016-11-08.06:31:33] Processing A3.mcmt  
[2016-11-08.06:31:33] K-Induction: checking initialization 0  
[2016-11-08.06:31:33] K-Induction: got unsat  
[2016-11-08.06:31:33] K-Induction: checking consecution 0  
[2016-11-08.06:31:33] K-Induction: got sat  
[2016-11-08.06:31:33] K-Induction: checking initialization 1  
[2016-11-08.06:31:33] K-Induction: got unsat  
[2016-11-08.06:31:33] K-Induction: checking consecution 1  
[2016-11-08.06:31:33] K-Induction: got unsat  
valid
```