DASH: Modelling and Analysis of Declarative State-Based Transition Systems

Work in Progress (*)

Jose Serna, **Nancy A. Day**, and Shahram Esmaeilsabzali
With past contributions by Sabria Farheen

Waterloo Formal Methods Lab
David R. Cheriton School of Computer Science
University of Waterloo

Future of Alloy Workshop
Apr 2018
http://129.97.7.33:8080/dash/

(*) In International Workshop on Model-Driven Requirements Engineering (MoDRE) @ IEEE International Requirements Engineering (RE),Sep 2017

Modelling Transition Systems of Reactive Systems

- Modelling the control-oriented aspects of a system can be done naturally in hierarchical and concurrent control states (*i.e.*, the Statecharts family of languages).
  - Control states means a state with a name.
- Control-oriented modelling languages lack abstractions for modelling data.
- We need both! An integrated model that describes both control and data aspects of the system at an abstract level.
- Goal: a natural extension to Alloy to create these integrated models.

Modelling Transition Systems

- A transition system (TS) has: snapshots (variables with values) and transitions.

Modelling Transition Systems

- A transition system (TS) has: snapshots (variables with values) and transitions.
- Control states are a means of sequencing transitions by factoring/grouping snapshots that share the same set of possible future behaviours.
- Typically, the names of control states are user-observable elements of the model (*i.e.*, red, yellow, green of a traffic light) so are natural for modellers.

Modelling Transition Systems

- A transition system (TS) has: snapshots (variables with values) and transitions.
- Control states are a means of sequencing transitions by factoring/grouping snapshots that share the same set of possible future behaviours.
- Typically, the names of control states are user-observable elements of the model (*i.e.*, red, yellow, green of a traffic light) so are natural for modellers.
- Hierarchical states provide more model decomposition and a natural way to express priority.
- Concurrent states provide a separation of concerns for aspects of the model that can happen at the "same" time.

Modelling Transition Systems

- A transition system (TS) has: snapshots (variables with values) and transitions.
- Control states are a means of sequencing transitions by factoring/grouping snapshots that share the same set of possible future behaviours.
- Typically, the names of control states are user-observable elements of the model (*i.e.*, red, yellow, green of a traffic light) so are natural for modellers.
- Hierarchical states provide more model decomposition and a natural way to express priority.
- Concurrent states provide a separation of concerns for aspects of the model that can happen at the "same" time.
- Events are occurences of user actions; or internal actions to model cascading effects between concurrent regions.

```
1  abstract sig ValvePos {}
2  one sig OpenPos, ClosedPos extends ValvePos {}
3  abstract sig Room {}
4
5  state HeatingSystem {
6      valve: Room -> ValvePos
7      desiredTemp: Room -> Int
8      ...
9      event deactivate {}
10     init {
11         all r: Room | r.valve = ClosedPos
12     }
13     conc state Furnace {     ...    }
14     conc state Controller {
15         default state Off { }
16         state On {
17             trans t3 {
18                 on heatSwitchOff
19                 goto Off
20                 do r.valve' = ClosedPos
21                 send deactivate
22             }
23         ...
24     }
25     ...
26 }
```

Alloy

control state

snapshot variables

create an event

initial snapshot constraint

Alloy

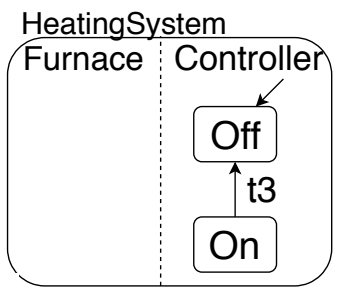
control state

transition

event

Alloy (w/ prime)

send an event



Syntax of Core Dash

Everything in Alloy plus:

```
1   [conc|default] state <name> { ...   }
2
3   trans <name> {
4       from <src_state>
5       on <trigger_event>
6       when <alloy>
7       goto <dest_state>
8       do <alloy>
9       send <generated_event>
10  }
11
12  event <name> {}
13
14  condition <name> {}
```

on/when = precondition

postcondition

Dash Features

- Transitions understood within context (source state, *etc.*)
  - Concise
  - Permits factoring by state, event, and condition
  - If no natural control states, trans are loops on enclosing state

Dash Features

- Transitions understood within context (source state, *etc.*)
  - Concise
  - Permits factoring by state, event, and condition
  - If no natural control states, trans are loops on enclosing state
- Global access to variables, but namespaces by state

Dash Features

- Transitions understood within context (source state, *etc.*)
  - Concise
  - Permits factoring by state, event, and condition
  - If no natural control states, trans are loops on enclosing state
- Global access to variables, but namespaces by state
- Transition comprehension (patterns)

```
trans to_idle {
  from * on hang_up goto idle
}
```

Dash Features

- Transitions understood within context (source state, *etc.*)
  - Concise
  - Permits factoring by state, event, and condition
  - If no natural control states, trans are loops on enclosing state
- Global access to variables, but namespaces by state
- Transition comprehension (patterns)

```
trans to_idle {
  from * on hang_up goto idle
}
```

- Add-ons:

```
addon (do incErrorCounter)
    to (from * goto Error)
addon (do incErrorCounter) to t19
```

Dash Features

- Transitions understood within context (source state, *etc.*)
  - Concise
  - Permits factoring by state, event, and condition
  - If no natural control states, trans are loops on enclosing state
- Global access to variables, but namespaces by state
- Transition comprehension (patterns)

```
trans to_idle {
  from * on hang_up goto idle
}
```

- Add-ons:

```
addon (do incErrorCounter)
    to (from * goto Error)
addon (do incErrorCounter) to t19
```

- Transition templates (reusable)

Semantics of Dash

- A "big" step (model's response to env) can be multiple transitions in different concurrent states.
- Philosophy for Semantic Decisions: Atomicity of transitions; internal sequencing of transitions in concurrent states should be rare.

Semantics of Dash

- A "big" step (model's response to env) can be multiple transitions in different concurrent states.
- Philosophy for Semantic Decisions: Atomicity of transitions; internal sequencing of transitions in concurrent states should be rare.
- Using the framework of Esmaeilsabzali:
  - **Concurrency:** only one concurrent state can take a transition in a big step (avoids race conditions and makes model's behaviour more understandable).
  - **Big-step Maximality:** only one transition per concurrent region in a big step (avoids potentially infinite big-steps).
  - **Event Lifeline:** events last the entire big step.
  - **Variable Lifeline:** variables change their values in small steps.
  - **Priority:** Outer state over inner state.

Semantics: Frame Problem

- Mismatch:
  - In control-oriented modelling languages, any unchanged variable keeps its value in the next state (like a program).
  - In declarative modelling languages, a variable must be explicitly constrained.

Semantics: Frame Problem

- Mismatch:
  - In control-oriented modelling languages, any unchanged variable keeps its value in the next state (like a program).
  - In declarative modelling languages, a variable must be explicitly constrained.
- Our solution:
  - If variable a is NOT designated environmental and is not mentioned in the action of the taken transition, it keeps its value from the previous snapshot.
  - To override this semantics, action can be that variable a must be within its range of values ($0 \leq a' \leq 100$).

Tool Chain: xText

- Grammar for Dash (which includes Alloy)
- "Smart" editor automatically created
- Transformation rules to:
  1. Transform to Core Dash (state hierarchy plus fully detailed transitions)
  2. Transform Core Dash to Alloy
     - Transition names are present in Alloy so it's easier to relate a counterexample to the model.

Analyzing Transition Systems in Alloy

BMC (Bounded Model Checking)
Scoped TCMC (Transitive-Closure-based Model Checking for CTLFC)
Significance axioms to ensure the TS is "big enough"

Table: Deducing Complete Model Checking Results

		Scoped TCMC		BMC using `ordering`	
Property		Pass	Fail	Pass	Fail
Safety		Ambiguous	Real Bug	Ambiguous	Real Bug
Finite Liveness	w/o dead-loop	Ambiguous	Real Bug	Real Pass	Ambiguous
	w/ dead-loop	Real Pass	Ambiguous		
Infinite Liveness		Ambiguous	Real Bug	*Cannot Express*	
Existential		Real Pass	Ambiguous	*Cannot Express*	

Summary

- Dash is an extension of Alloy to create an integrated model that describes both control and data aspects of the system at an abstract level.
- Dash provides explicit syntax for:
  - Transitions, preconditions, postconditions
  - Hierarchical and concurrent control states
  - Events to allows cascading effect between concurrent regions
  - Semantics chosen to match combination of declarative and control-oriented modelling paradigms and address the frame problem.
  - Syntactic sugar for conciseness (transition comprehension, add-ons, factoring by events, conditions, transition templates).
- Dash is fully integrated with Alloy for everything else (expressions, data abstractions, functions, and predicates).