A Comprehensive Study of Declarative Modelling Languages

B, Event-B, Alloy, Dash, TLA$^+$, PlusCal, AsmetaL

Amin Bandali

June 30, 2020

UNIVERSITY OF WATERLOO
FACULTY OF MATHEMATICS
DAVID R. CHERITON SCHOOL OF COMPUTER SCIENCE

Formal Specifications

Architects draw detailed plans before a brick is laid or a nail is hammered. Programmers and software engineers don't.

Can this be why houses seldom collapse and programs often crash?

To designers of complex systems, the need for ***formal specifications*** *should be as obvious as the need for blueprints of a skyscraper.*

But few software developers write specifications because they have little time to learn how on the job, and they are unlikely to have learned in school.

— Leslie Lamport, Turing Award Winner, 2013

Declarative Behavioural Modelling

Declarative behavioural modelling is a powerful modelling paradigm that enables users to model system functionality abstractly and formally.

An *abstract model* is a concise and compact representation of the key characteristics of a system, and enables the stakeholders to reason about the correctness of the system in the early stages of development.

Declarative Modelling Languages

- are used to write behavioural formal specifications of systems
- using a state-machine-oriented / transition system approach
- in essence model a Kripke structure

Examples of declarative modelling languages:

- Alloy
- TLA$^+$
- VDM
- Z

Declarative Models

- describe the transitions declaratively using constraints, rather than imperative calculations and/or statements;
- include user-defined and -axiomatized units of data, which can represent rich datatypes such as lists and trees;
- have a formal mathematical and logical foundation, usually first-order logic (FOL) and/or set theory; and
- allow writing models without specifying the size of sets (the scopes); the scopes may need to be specified for analysis.

Use of Declarative Models

- Zave's use of Alloy and Spin to find specification-level bugs in the specification of the Chord network protocol;
- Amazon's use of TLA$^+$ has helped find subtle bugs in complex real-world systems and prevent the bugs from reaching production; and
- Huynh *et al.*'s use of B for formalizing a new healthcare access control model with conflict resolution for overriding patient consent as to who can access their Electronic Health Records (EHR) under strictly defined scenarios by regional laws of Québec and Canada to protect the patient's life.

Research Question

Given that there are a great many declarative modelling languages to choose from, *how does one make a choice of which language to use*?

Thesis Contributions

The contributions of this thesis are

- a set of criteria to compare declarative modelling languages;
- comparison of selected declarative modelling languages (B, Event-B, Alloy, Dash, TLA$^+$, PlusCal, and AsmetaL) based on these criteria; and
- recommendations for the choice of modelling language based on the characteristics of the transition system under description.

Introduction & Motivation

Methodology

Comparison Criteria

- Control Modelling
- Data Modelling
- Modularity

Contributions

Methodology

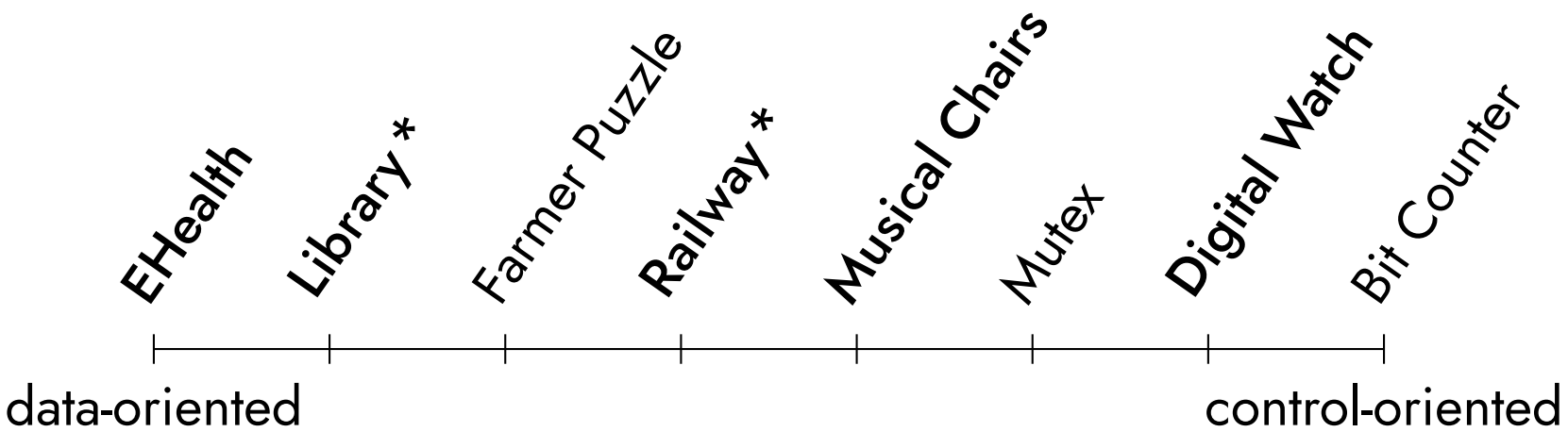
1. select 6 relatively small examples across the data- vs. control-oriented spectrum;
   - *control-oriented*: has complex conditions for when a transition is relevant that are naturally expressed using modes, control states, or concurrency; and
   - *data-oriented*: has complex constructions of data;
2. model the examples in the 3 languages B, Dash, and TLA$^+$;
3. describe the differences and similarities across the languages while modelling the examples, forming the initial comparison criteria for comparing the languages; and
4. publish our results.

Methodology (cont'd)

5. expand set of comparison criteria to include other interesting characteristics of languages;
6. expand our set of chosen languages to include AsmetaL, Alloy, Event-B, and PlusCal in addition to B, Dash, and TLA$^+$;
7. model existing examples in new languages, ensuring proper setup of tool support for all of the languages;

Methodology (cont'd)

8. expand our set of examples with 2 new larger systems, Library and Railway (marked with an asterisk), increasing our model count from 18 relatively small examples to 35 including the larger ones; our final case studies are those typeset in **bold**; and



Methodology (cont'd)

Table 1: Order of modelling case studies across languages

Case study \ Language	B	Event-B	Alloy	Dash	TLA$^+$	PlusCal	AsmetaL
EHealth	1	2	3	1	1	4	5
Musical Chairs	1	4	E	1	1	3	2
Digital Watch	1	2	5	1	1	4	3
Library	E	1	E	3	5	2	4
Railway	1	7	3	5	6	4	2

Legend: E indicates Existing models, *i.e.* those that we had no influence on. The numbers in each row indicate the order of languages the case study was done in.

9. note the differences and similarities across the languages with respect to our comparison criteria while modelling the examples.

Introduction & Motivation

Methodology

Comparison Criteria

- Control Modelling
- Data Modelling
- Modularity

Contributions

Comparison Criteria

We classify our comparison criteria into 3 main categories:

- **control modelling** (structuring transition systems),
- **data modelling** (data descriptions in transition systems), and
- **modularity** aspects of modelling.

Introduction & Motivation

Methodology

Comparison Criteria

- Control Modelling
- Data Modelling
- Modularity

Contributions

Control Modelling

concerned with control aspects and structure of transition systems

Control Modelling

concerned with control aspects and structure of transition systems

a transition system TS is a tuple (S, TR, I), where

- S is a set of snapshots,
- $TR \subseteq S \times S$ is a transition relation, and
- $I \subseteq S$: is a set of initial snapshots.

Control Modelling Criteria

- snapshot variables
- initialization
- **transition relation**
- **control state hierarchy**
- invariants
- inconsistency
- **frame problem**

Control Modelling — Transition Relation

- completely explicit: Alloy
- mostly explicit: TLA$^+$ and AsmetaL
- implicit: B, Event-B, Dash, and PlusCal

Control Modelling — Control State Hierarchy

- **labelled control state**: is a distinguished set of variables with a finite set of values, that are used to control when a transition can be taken
- languages with labelled control states can have **control state hierarchy** and concurrency
- control state hierarchies are a powerful tool for representing the states or modes of a control-oriented system
- in our set of languages, unique feature of Dash

Control Modelling — Frame Problem

refers to the issue of how snapshot variables that are not explicitly constrained in a transition may or may not change from one snapshot to the next

Control Modelling — Frame Problem

- Alloy: all unconstrained variables may change
- B, Event-B, and PlusCal: all unconstrained variables remain unchanged
- Dash and AsmetaL: monitored (environmental) variables may change from one snapshot to the next, controlled variables not constrained in a transition remain unchanged by it
- TLA$^+$: requires all transitions to constrain all variables, either by constraints on primed and unprimed names of variables or by marking them with the `UNCHANGED` keyword

Introduction & Motivation

Methodology

Comparison Criteria

- Control Modelling
- Data Modelling
- Modularity

Contributions

Data Modelling

concerned with the description of the data aspects of models

Data Modelling Criteria

- primitives & subtypes
- **constructors**
- built-ins
- expressions
- events
- constants
- well-formedness & **typechecking**
- scopes

Data Modelling — Constructors

are operators that create *composite units of data* from primitives or other composite data units

examples: functions, relations, and records

constructors may include *multiplicities*, which impose constraints limiting the values in the composite data being constructed

Data Modelling — Constructors

examples:

- B and Event-B have *arrow* constructors for creating functions; *e.g.*
  - $\nrightarrow$ and $\rightarrow$ for partial and total functions
  - $\twoheadrightarrow$ and $\twoheadrightarrow$ for partial and total surjective functions
- Alloy and Dash have *multiplicity keywords* such as `lone`, `one`, and `some` that can be used to create various kinds of functions; *e.g.*
  - `->lone` and `->one` for partial and total functions
  - `some->lone` and `some->one` for partial and total surjective functions

Data Modelling — Typechecking

is the process of checking whether the use of data in the formulas conforms to the constraints expressed in the *type signatures*

type signatures are syntax denoting the kind of object contained in a snapshot variable, constant, or quantified variable as either a primitive or a composite unit of data

Data Modelling — Typechecking

- TLA$^+$ and PlusCal have no type signatures, and typing constraints are stated and checked along with other invariants
- in Alloy and Dash typechecking consists of checking that no relation has been given different arities and that no expression can be shown to be redundant or contain a redundant sub-expression using solely the declarations
- B, Event-B, and AsmetaL have type signatures and typechecking that help statically catch errors like assigning a value from a set to a variable with a type signature declaring a different/incompatible set, and applying a function to arguments that do not match its type signature

Introduction & Motivation

Methodology

Comparison Criteria

- Control Modelling
- Data Modelling
- Modularity

Contributions

Modularity

concerned with the constructs of each language for writing modular descriptions of transition systems

Modularity Criteria

- decomposition into **subtransition relations**
- namespaces of subtransition relations
- data decomposition into multiple files
- file import
- file export
- file parameterization
- file namespaces
- syntax overloading

Modularity — Subtransition Relations

- a subtransition system is a full description of a transition system
- subtransition systems are composed to create the single top-level transition relation implicitly or explicitly
- B, Alloy, TLA$^+$, and AsmetaL support decomposition into subtransition systems

Introduction & Motivation

Methodology

Comparison Criteria

- Control Modelling
- Data Modelling
- Modularity

Contributions

Recommendations

- Alloy, TLA$^+$, and AsmetaL allow fine control over the **transition relation** of the transition system, due to their explicit representation for the transition relation
- Dash is a great choice for modelling control-oriented systems, such as the digital watch case study, where relevance of **transitions** and their being enabled can be captured using potentially hierarchical **control states**, **events**, and concurrent regions
- Alloy models may suffer from **inconsistency** due to under-specification of behaviours, relating to the **frame problem**, as we saw in the EHealth and Library case studies; TLA$^+$ addresses this using the `UNCHANGED` keyword

Recommendations (cont'd)

- Dash and AsmetaL allow a clear distinction between the system and its surrounding environment by distinguishing between **controlled** and **monitored** variables
- B and Event-B's extensive set of arrow **constructors** for various relational and functional units of data allow terse and concise descriptions of highly-constrained composite units of data in highly data-oriented systems, such as the Library case study, both in **type signatures** and in formulas, similar to Alloy and Dash's **multiplicities**

Recommendations (cont'd)

- B and Event-B are similar languages with the same roots, but they differ in B's support for several kinds of relationships between machines, B's sequence and tree **built-ins** and record **constructor**, Event-B's additional arrows for constructing specific kinds of relations, and Event-B's support for declaring **subtypes** and partitioning a set into multiple disjoint subsets;
- PlusCal enjoys the power and expressiveness of TLA$^+$'s **expressions**, wrapped in a syntax geared more towards modelling multi-process concurrent and parallel algorithms, with additional **well-formedness** safeguards making certain classes of bugs unrepresentable in a valid model

Recommendations (cont'd)

- with respect to modularity, Event-B, Dash, and PlusCal allow **data decomposition** of a model across multiple files, while B, Alloy, TLA$^+$, and AsmetaL additionally allow **control decomposition** of a model into subtransition systems across multiple files.

Thesis Contributions

The contributions of this thesis are

- a set of criteria to compare declarative modelling languages;
- comparison of selected declarative modelling languages (B, Event-B, Alloy, Dash, TLA$^+$, PlusCal, and AsmetaL) based on these criteria; and
- recommendations for the choice of modelling language based on the characteristics of the transition system under description.

Thanks!

Lines of Code for Case Studies

Table 2: Lines of code for each case study across the languages

Case study \ Language	B	Event-B	Alloy	Dash	TLA$^+$	PlusCal	AsmetaL
EHealth	62	111	135	95	120	101	87
Digital Watch	135	210	295	112	197	160	142
Musical Chairs	68	84	130	65	101	106	97
Library Management	180	164	317	120	146	151	207
Railway	82	86	387	280	79	84	78