Introduction to Z

Bart Massey

January 7, 2016

Values

What is a value?

- Mathematical "value" is fundamental
- Primitive things like numbers, symbols
- Composite things like sets
- Values can be *named*.
- Naming is not the same as assignment!

Constraints

- A *constraint* partially specifies values
- e.g $a \in \mathbb{N}, a > 5$ tells us something about a, but not all
- $a = 5$ is also a constraint, so naming is constraining
- We particularly care about constraints between inputs, outputs, and states

Types

You've worked with types your whole career. But what is a type, anyway?
A type is a collection of values. It denotes a membership constraint on a value. So $a \in \mathbb{N}$ and $a : \mathbb{N}$ are equivalent statements.

Finite State Machines

A *state* associates values with a time (sort of).

A finite state machine has a finite set of states, with well-specified *transitions* between them. It usually has a *start state* and some *accepting states*.

Parameters

We separate the system under study from the “real world”.
External inputs and outputs drive state machines, and outputs are conditioned on inputs.

Sets

A set is a collection of items. (Items can be anything.)
A set is an unordered collection
A set has no duplicate elements

- Platonic ideals of things
- Convenient in a surprising number of places
- Can use "property functions" to deal with duplication

May be "typed": All elements of the set are the same "kind"

Set Descriptions

Set displays: $\{0, 1, 2, 3\}$
Set constructors: $\{x : \mathbb{N} \mid x < 4\}$
Informal descriptions with dots: $\{0 \,..\, 3\}$
Construction using set operations:

- Union:

$$A \cup B = \{e \mid e \in A \vee e \in B\}$$

- Intersection:

$$A \cap B = \{e \mid e \in A \wedge e \in B\}$$

Views Of Z

- Formalized mathematical notation for
  - automated typechecking
  - automated reasoning
  - easy reading
- Precise description for
  - checking consistency
  - checking completeness
  - organizing model

Z Notation

Z consists of names, values, and constraints organized into *paragraphs*. By convention, all-caps names are types, names ending in '?' are inputs, names ending in '!' are outputs, and names ending with a single-quote are “after-states”.

Z Paragraphs

- *Paragraph* is Z basic unit:
  - *Declarations* give interface + types
  - *Constraints* give relation between vars
- Constraint part may be omitted

Z Top-Level Paragraphs

Some parts of Z description are global, e.g.

- Set existence

$$[PLAYER]$$

- Free types

$$OBJ ::= rock \mid scissors \mid paper$$

- Constraints

$$\# PLAYER = 2$$

Z Schema Definitions

A *schema* defines and constrains *state*, e.g.

- Definition

$$\begin{array}{|l}
\textit{Referee} \\
\hline
\textit{referee} : \textit{OBJ} \times \textit{OBJ} \rightarrow \textit{VAL} \\
\hline
\end{array}$$

- Definition with constraints

$$\begin{array}{|l}
\textit{Beats} \\
\hline
\textit{beats} : \mathbb{P}(\textit{OBJ} \times \textit{OBJ}) \\
\hline
\textit{beats} = \{(\textit{rock}, \textit{scissors}), \\
\qquad\quad (\textit{scissors}, \textit{paper}), \\
\qquad\quad (\textit{paper}, \textit{rock})\} \\
\hline
\end{array}$$

Z and State

A Z schema describes a state. It is essentially a node in a state machine.

A Z schema can also describe a state transition: an edge in a state machine. The "before" and "after" (un-primed and primed) values are constrained with respect to each other.

Z Is Not Stand-alone

Every Z paragraph should be surrounded by English. This is nice, because it makes it possible for mere mortals to understand the Z. It is also *necessary* to provide a connection between the Z and reality.