Z Specification of WhenBus

Arjun, Shashikanth, Vinay, Harsha and Hemanth Robbi

Dept. of Computer Science and Engineering
IIT Madras
email : WhenBus.software@gmail.com

Abstract

This paper presents the Z language specification of the WhenBus Application.

Introduction

This paper presents the technical details of the WhenBus application. The various Z-Schemas of components involved in the application are listed.

In order to provide a complete and consistent definition of the objects and the operations which act on them, there is a need to develop a formal model. Formalism also allows proofs of desired properties of the system. A more rigorous definition can be provided by using a formal language. We have given a formal specification WhenBus system using Z. This is a general purpose specification language, using mathematics to specify the system, and schemas to structure and modularize the specification. A Z specification consists of a mix of formal mathematics and informal English text. The formal part provides the precise, unambiguous definition of the system. The informal text acts as a commentary on the formal part. It can be consulted to find out what aspects of the real world are being described.

A Z specification describes the state of the system, and various operations that can change the state. It uses schema boxes to display this information.

Using Z-schemas we will be able to describe both static and dynamic aspects of the application. The static aspects include:

- The states a particular object can occupy
- The various constraints that are to be met, irrespective of the dynamics of the system.

The dynamic aspects include:

- The various methods the application supports
- The IO specification

Definitions

BUS_NUMBERS : Is a set of all the bus numbers
MASTER_BUSES_SET : Set of all MASTER_BUS objects
QUESTIONS : Is a set of all identifiers of various questions formulated to get user feedback
STOP_MAP : name $\mapsto$ stop_id
QUESTION_MAP : $\mathbb{N}\mapsto$ QUESTION
BUS_NUMBER_MAP : bus_number $\mapsto$ MASTER_BUS
FPU : FeedBack Providing User [see SRS for details]
COMMUTER : User not providing a Feedback
USER_STATE : FPU | COMMUTER

$\mathbb{P}$ ***in the following specification indicates a list.***

Z-Specification

$$
\begin{array}{|l}
\hline
USER \\
type : USER\_STATE \\
info : USER\_INFO \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
USER\_INFO \\
is\_qualitative : \{0, 1\} \\
bus\_number : STRING \\
bus\_id : \mathbb{N} \\
question\_id : \mathbb{N} \\
response : \mathbb{N} \\
location : LOCATION \\
\hline
question\_id \in QUESTIONS \\
response \in QUESTION\_MAP\ \ (question\_id).responses \\
bus\_number \in BUS\_NUMBERS \\
bus\_id \in \\
\{b.id \mid b \in BUS\_NUMBER\_MAP(bus\_number).buses\} \\
\hline
\end{array}
$$

LOCATION

$latitude : \mathbb{R}$
$longitude : \mathbb{R}$

$latitude \in [-90.0, 90.0]$
$longitude \in [-180.0, 180.0]$

QUESTION

$id : \mathbb{N}$
$question : STRING$
$responses\_string : \mathbb{P}\, STRING$
$responses : \mathbb{P}\, \mathbb{N}$
$response\_map : STRING \mapsto \mathbb{N}$

MASTER_BUS

$bus\_number : STRING$
$route : \mathbb{P}\, BUS\_STOP$
$buses : \mathbb{P}\, BUS$
$source : BUS\_STOP$
$destination : BUS\_STOP$

$source =$ **begin(**$route$**)**
$destination =$ **end(**$route$**)**

MASTER_BUS is the parent of BUS schema. The BUS refers to each individual bus with the same bus_number

BUS

$id : \mathbb{N}$
$timing : BUS\_STOP \mapsto TIME$
$reverse : \{0, 1\}$
$location : LOCATION$
$update\_time : TIME\_STAMP$
$velocity : \mathbb{R}$

---

if$(reverse)$ {
$timing : monotonically\ \ decreasing$
}
else {
$timing : monotonically\ \ increasing$
}
$The\ \ ordering\ \ is\ \ based\ \ on\ \ the\ \ route\ \ in\ \ the\ \ parent\ \ MASTER\_BUS$

BUS_STOP

$id : \mathbb{N}$
$name : STRING$
$location : LOCATION$

Get_Bus_Info

$\Xi MASTER\_BUS$
$bus\_number? : STRING$
$bus\_stop? : BUS\_STOP$
$expected\_arrival! : TIME$

$bus\_number \in BUS\_NUMBERS$
$expected\_arrival! = \min b.timing(bus\_stop)$
$for\ \ b \in MASTER\_BUS.buses$
$\wedge\ b.timing(bus\_stop) \geq CURRENT\_TIME$

Get_Bus_Location

$\Xi MASTER\_BUS$
$bus\_number? : STRING$
$bus\_stop? : BUS\_STOP$
$bus\_location! : LOCATION$

$bus\_location! = b.location$
$where\ \ b = argmin\ \ c.timing(bus\_stop)$
$for\ \ c \in MASTER\_BUS.buses$
$\wedge\ c.timing(bus\_stop) \geq CURRENT\_TIME$

Send_Feedback

ΔBUS
$user\_info? : USER\_INFO$

$bus = b$
$b \in BUS\_NUMBER\_MAP(user\_info?.bus\_number).buses$
$where\ \ b.id = user\_info?.bus\_id$
if$(user\_info?.is\_qualitative)$ {
$calculate\ \ \delta\ \ the\ \ time - shift\ \ from\ \ user\_info?.response$
}
else {
$bus'.location = user\_info?.location$
$calculate\ \ \delta\ \ the\ \ time - shift\ \ from\ \ user\_info?.location$
}
$bus'.timing[i] = bus.timing[i] + \delta$
$\forall\, i >= nearest\_bus\_stop(user\_info?.location)$
$bus'.update\_time = CURRENT\_TIME$

Textual Description

USER

The user has two possible types FPU | COMMUTER. When the user is willing to provide anonymous feedback to the application for real-time update the user type is FPU, else it is COMMUTER. The user also is associated with the object USER_INFO.

USER_INFO

This object helps in providing the necessary information for real-time update of the bus timing of the bus the user is currently requesting or is travelling in one. As mentioned, in the SRS the feedback is either qualitative or quantitative. See SRS for more details, described in the feedback protocol section.

LOCATION

This object is used to standardize how geo-location is represented, it uses the Decimal degrees format.

QUESTION

The QUESTION object formally defines how the application stores the questions which are used for obtaining qualitative feedback from the user. The list of these questions is fixed and is associated with a formula to help update the timing information in the database.

MASTER_BUS

This entity represents a bus of a specific number, and contains a list of BUS objects which have the same bus_number and follow the same route but with different timings.

BUS

This represents individual buses, which help in tracking it and hence provide location of the bus and the schedule that particular bus follows.

BUS_STOP

This entity is used to store the various bus stops in the city, and associates a location attribute to each stop.

Get_Bus_Info

This method helps the user to get the expected arrival time of a bus, which the user specifies by the bus_number and the bus_stop the user is in.

Get_Bus_Location

This method similar to the Get_Bus_Info provides the most likely position of the next bus in the MAP interface.

Send_Feedback

This method is what makes the application a crowd-source application. With the help of the user_info object from the FPUs, the application updates the information it has regarding the bus requested or travelled in by the FPU.

Conclusion

The formal specification of various schemas and interface the WhenBus Application offers is specified using the Z-language. This offers a clear view of what the software can do by capturing the important features of the Application.

Most schemas presented in the paper is what will be used for storing data in almost the same format. Clearly, Formally stating what the system does helps in designing the entire software in a modular fashion.

References

Z- Language Introduction
www.cs.nott.ac.uk/ pszrq/files/4FSPnotation.pdf