Abdul Hafeez `CS14B031` Harshith Reddy `CS14B011`
P Sai Mohith `CS14B047` Sachin Sridhar `CS14B057`
Y Deekshit Reddy `CS12B030`

February 6, 2017

Real Office Z Schema

Here we model an Event Scheduler system for CSE Department Office in Z. Scheduling a number of events in multiple venues on a daily basis is a painstaking task for anyone.The purpose of this system is to make event scheduling system automated for the CSE Department Office.
Here is an excerpt from the informal description:

> Suppose the office staff would like to schedule an event then they give the required parameters as input. The system then checks for availability of different resources for the given parameters and allocates resources. All information about an event is also stored for future reference in a database.

We begin the Z model by introducing five basic sets that hold everything of interest in this universe, namely events, venues, times, strings(errors,reports and output notifications from a query):

$$[\mathbf{EVENTS}, \mathbf{VENUE}, \mathbf{TIME}, \mathbf{CAPACITY}, \mathbf{STRING}]$$

The set of state variables are as follows:

> **eventname**: This is a set of all event names which are recorded in the database at a particular time.
> **eventvenue**: This is a function which when applied to certain events gives back the venues associated with the respective events.

evnentime: This is a function which when applied to certain events gives back the time interval associated with the respective event.The interval is described by start time and end time.
venueproj: This is a function which when applied to certain venues gives back whether a projector is present or not present in the respective venue(s).
venuecap: This is a function which when applied to certain venues gives back the maximum capacity associated with each venue.
name: This is an input variable which takes the event name from user.
venue: This is an input variable which takes the venue required for an event from user.
time: This is an input variable which takes the time interval(start time and end time) required for an event from user.
capacity: This is an input variable given by the user to specify the capacity requirement for the event.
projector: This is an input variable where user specifies projector requirement.
refreshment_req: This is an input variable where user specifies refreshment requirements.
stationary: This is an input variable where user specifies stationary requirements.
error: output variable which contains the error messages for various input parameters(event name,venue,time,projector requirement,event capacity) not being satisfied while adding an event.
report: output variable which contains the status of the cancel event operations .
info: output variable which contains the info of a particular event when queried for.
error1:output variable which is used to give a message when a particular event user has given to find does not exist.
events: output variable which stores the list of events for which notifications have to given at a particular time.

The following schema is a state space schema defining all the static variables and functions associated with them.

REALOFFICE

$$
\begin{array}{l}
\mathbf{eventname} : \mathbf{EVENTS} \\
\mathbf{eventvenue} : \mathbf{EVENTS} \nrightarrow \mathbf{VENUE} \\
\mathbf{eventtime} : \mathbf{EVENTS} \nrightarrow \mathbf{TIME} \\
\mathbf{venueproj} : \mathbf{VENUE} \nrightarrow \{\mathbf{present}, \mathbf{notpresent}\} \\
\mathbf{venuecap} : \mathbf{VENUE} \nrightarrow \mathbf{CAPACITY} \\
\hline
\mathbf{eventname} = \operatorname{dom} \mathbf{eventvenue} \\
\mathbf{eventname} = \operatorname{dom} \mathbf{eventtime} \\
\operatorname{ran} \mathbf{eventvenue} = \operatorname{dom} \mathbf{venueproj} \\
\operatorname{ran} \mathbf{eventvenue} = \operatorname{dom} \mathbf{venuecap}
\end{array}
$$

We then define various function schema for atomic operations like adding an event, cancelling an event, and notifications. The modify event operation can be achieved by calling CANCELEVENT and then ADDEVENT with the new parameters.

The Add Event schema takes inputs name, venue, time, projector requirement, refreshment requirement, stationary requirements .It does checks on the validity of these input arguments. If the arguments are valid, it updates the list of events.

ADDEVENT

$\Delta\mathbf{REALOFFICE}$
$\mathbf{name?} : \mathbf{EVENTS}$
$\mathbf{venue?} : \mathbf{VENUE}$
$\mathbf{time?} : \mathbf{TIME}$
$\mathbf{projector?} : \{\mathbf{required}, \mathbf{notrequired}\}$
$\mathbf{capacity?} : \mathbf{CAPACITY}$
$\mathbf{refreshment\_req?} : \{\mathbf{present}, \mathbf{notpresent}\}$
$\mathbf{stationary\_req?} : \{\mathbf{present}, \mathbf{notpresent}\}$
$\mathbf{error!} : \mathbf{STRING}$

$(\mathbf{name?} \notin \mathbf{eventname} \wedge$
$\mathbf{venue?} \in \operatorname{ran} \mathbf{eventvenue} \wedge$
$\phi = \{\mathbf{x} : \mathbf{EVENTS} \mid \mathbf{eventvenue(x)} = \mathbf{venue?} \wedge \mathbf{eventtime(x)} = \mathbf{time?}\} \wedge$
$\mathbf{capacity} <= \mathbf{venuecap(venue)} \wedge$
$\mathbf{venueproj(venue?)} = \mathbf{present} \wedge$
$\mathbf{eventname}' = \mathbf{eventname} \bigcup \{\mathbf{name?} \nrightarrow \mathbf{venue?}\} \bigcup \{\mathbf{name?} \nrightarrow \mathbf{time?}\})$
$\vee$
$(\mathbf{name?} = \mathbf{eventname} \wedge$
$\mathbf{error!} = \mathbf{EventAlreadyExists})$
$\vee$
$(\mathbf{venue?} \notin \operatorname{dom} \mathbf{eventvenue} \wedge$
$\mathbf{error!} = \mathbf{VenueDoesNotExist})$
$\vee$
$(\{\mathbf{x} : \mathbf{EVENTS} \mid \mathbf{eventvenue(x)} = \mathbf{venue?} \wedge$
$\mathbf{eventtime(x)} = \mathbf{time?}\} \neq \phi \wedge$
$\mathbf{error!} = \mathbf{EventClashesWithAnotherEvent})$
$\vee$
$(\mathbf{projector?} = \mathbf{required} \wedge$
$\mathbf{venueproj(venue?)} = \mathbf{notpresent} \wedge$
$\mathbf{error!} = \mathbf{ProjectorNotPresent})$
$\vee$
$(\mathbf{capacity} <= \mathbf{venuecap(venue)} \wedge$
$\mathbf{error!} = \mathbf{VenueCapacityInsufficient})$

The Cancel Event schema checks if the input event name exists and removes it if it does.Also when an event is removed all the corresponding data that is stored for that event is deallocated.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textbf{CANCELEVENT}} \\
\hline
\Delta\textbf{REALOFFICE} \\
\textbf{name}? : \textbf{EVENTS} \\
\textbf{report}! : \textbf{STRING} \\
\hline
(\textbf{name}? \notin \textbf{eventname} \land \\
\textbf{report}! = \textbf{EventDoesNotExist}) \\
\lor \\
(\textbf{name}? \in \textbf{eventname} \land \\
\textbf{eventname}' = \textbf{eventname} \setminus \{\textbf{event}\} \land \\
\textbf{report}! = \textbf{EventCancelled}) \\
\hline
\end{array}
$$

The Find Event schema does a query for a given input event name and returns the info associated with it ,gives a report if event does not exist.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textbf{FINDEVENT}} \\
\hline
\Xi\textbf{REALOFFICE} \\
\textbf{name}? : \textbf{EVENTS} \\
\textbf{info}! : \textbf{STRING} \\
\hline
(\textbf{name}? \in \textbf{eventname} \land \\
\textbf{info}! = \text{"}\textbf{Name} : \text{"} + \textbf{name} + \\
\text{"}\textbf{Venue} : \text{"} + \textbf{eventvenue}(\textbf{name}) + \\
\text{"}\textbf{Time} : \text{"} + \textbf{eventtime}(\textbf{name}) + \\
\text{"}\textbf{ProjectorAtVenue} : \text{"} + \textbf{venueproj}(\textbf{name})) \\
\lor \\
(\textbf{name}? \notin \textbf{eventname} \land \\
\textbf{report}! = \textbf{EventDoesNotExist}) \\
\hline
\end{array}
$$

The Notify schema notifies the user if there is an event happening at time *duration* from now. The value of the variable *duration* can be statically set.The notify function time and duration it could be adjusted in two ways,generate an alert at the start if every day and generate alert ’x’ minutes/hours before the start of an event.

$$
\begin{array}{|l}
\textbf{NOTIFY} \\
\hline
\Xi\mathbf{REALOFFICE} \\
\mathbf{time}? : \mathbf{TIME} \\
\mathbf{events}! : \mathbf{P}(\mathbf{EVENTS}) \\
\hline
\mathbf{events}! = \{\mathbf{e} : \mathbf{EVENTS} \mid \mathbf{eventtime}(\mathbf{e}) = \mathbf{time}? + \mathbf{duration}\} \\
\hline
\end{array}
$$