

Robert McIntyre P149064
Formal Methods
SA1 Site Safety System
Gordon Dickers

Task One

A system was created using Z notation to describe the schema. This was written within Tex Studio. Tex studio is an opensource LaTeX editor. This uses the zed-csp package which allows schemas to be used.[1] Task one involved in creating a solution to a critical system.

1 Introduction

This system shows how the staff interact with the security systems when they enter and exit buildings. There are several different permissions for entering buildings.

This table shows which areas can be occupied by which personnel.

Table 1: Range/Domain table

	Reactor_Core	Control_Room	Admin_building	On_Site	Off_Site
Tech		x	x	x	x
Science	x	x		x	x
Management		x	x	x	x
Offices_Staff			x	x	x

2 Types

It has been identified that the system should manage which people can access certain areas of the site. [STAFF] is the set of all the individual staff that are working in the company. The notation for the types of buildings that people are allowed in is as follows:

Work_area ::= Reactor_Core | Control_Room | Admin_Building | On_Site | Off_Site

Security_level ::= Tech | Science | Management | Office_Staff

There are also messages as shown below:

Messages ::= staff_entered | staff_exited | staff_added | staff_removed | staff_not_present
| staff_in_area | staff_not_in_area | access_violation

3 Original State of System

<i>System</i> $Person : \mathbb{P} STAFF$ $areIn : Security_Level \Leftrightarrow STAFF$ $location : Work_Area \Leftrightarrow STAFF$
--

4 Initialisation

<i>initSystem</i> <i>System'</i> $personPosition = \emptyset$ $person = \emptyset$ $WorkArea = \{ ControlRoom, AdminBuilding, ReactorCore \}$ $SiteAccess = \{ On_Site, Off_Site \}$ $canAccess = \{ Tech \mapsto Control_Room, Tech \mapsto Admin_Building,$ $Tech \mapsto On_Site, Tech \mapsto Off_Site,$ $Science \mapsto Reactor_Core, Science \mapsto Control_Room,$ $Science \mapsto On_Site, Science \mapsto Off_Site,$ $Management \mapsto Control_Room, Management \mapsto Admin_Building,$ $Management \mapsto On_Site, Management \mapsto Off_Site,$ $Office_Staff \mapsto Admin_Building,$ $Office_Staff \mapsto On_Site, Office_Staff \mapsto Off_Site, \}$
--

5 Change of State Operations

- i) Add Staff
- ii) Remove Staff
- iii) Change Area
- iv) Leave Area

- v) Is Staff On Site
- vi) Is Staff in Area

5.1 Add Staff

When staff are added there needs to be validation to check if the person already exists in the database. There needs to be an error message showing the user that the data is invalid.

The schema below changes the system by using $\Delta(Delta)$ The pre-condition is that the person is not in the database.

<i>Addingstaff</i>	_____
$\Delta System$	
$s? : STAFF$	
$s? \neq known$	
$System' = System \cup \{s?\}$	
$Message! : Staff_added$	

Add staff Error Completes the all of the adding staff.

<i>AddStaffError</i>	_____
$\exists STAFF$	
$p? : person$	
$res! : RESPONSE$	
$p? \in system \wedge res! = StaffnotinSystem$	
$res! : Staff_Not_Added$	

5.2 Remove Staff

<i>Removing_Staff</i>	_____
$\Delta System$	
$rs? : Staff$	
$res! : RESPONSE$	
$rs? \in system$	
$rs? \notin StaffInBuildingSystem = System \setminus \{rs?\}$	
$res! = Staff_Removed$	

RemoveStaffError _____

Ξ STAFF

rse? : STAFF

res! : RESPONSE

$rse? \notin person \wedge res! = Personnotstaff$

$\vee$

$rse? \in person \wedge rse?buildingpersonPosition \wedge res! = Staffinthearea$

res! : Staff_Not_Found

5.3 Change Area

<i>ChangeArea</i>
$\Delta System$
$wa? : WORKAREA$
$s? : STAFF$
$d? : DEPARTMENT$
$res! : RESPONSE$
$s? \in canAccess \cap s? \notin wa?$
$workArea' = wa? \cup \{s?\}$
$res! = Staff_Entered$

<i>ChangeAreaError</i>
$res! : RESPONSE$
$res! = Access_Violation$

5.4 LeaveArea

<i>LeaveArea</i>
$\Delta System$
$wa? : WORKAREA$
$s? : STAFF$
$d? : DEPARTMENT$
$s? \in wa?$
$WorkArea' = WorkArea \setminus \{s?\}$
$res! = Staff_exited$

5.5 Is Staff on Site

<i>IsStaffonSite</i>
$s? : STAFF$
$b? : BUILDING$
$d? : DEPARTMENT$
$s? \in person$
$s? \in b?person' = person$
$personPosition' = personPosition \cup \{s? \mapsto d?\}$

5.6 Is Staff in Area

$ \begin{array}{l} \textit{IsStaffInArea} \\ \hline \Delta \textit{Systems?} : \textit{STAFF} \\ b? : \textit{BUILDING} \textit{res!} : \textit{RESPONSE} \\ \hline s? \in \textit{member} \\ \wedge \\ s? \in b? \\ \textit{res!} = \textit{staffInArea} \\ \textit{person}' = \textit{person} \\ \hline \end{array} $

5.7 Error Operations

Messages ::= staffEntered | staffExited | staffAdded | staffRemoved | staffNotPresent | staffInArea | staffNotInArea | accessViolation

Task Two

The SA1 site safety system described in task one can be classified within formal methods. There are several ways that formal methods that can be described. Two articles have been looked at which explain formal specification and formal verification. The first one is Hauxthausen[2] which splits the classification into three types formal specification , formal specification with semi formal verification and formal specification with formal verification. The second is Rushby[3] which defines the extent of formality of the specification. These articles have been used to decide which classification the SA1 site safety system fits into.

From the first article[2] the best fitting classification is formal specification with semi verification. The SA1 safety site uses formal notation with a language that has a precise syntax. The language is Z notation. The Z notation takes a system and describes the state of the system using mathematical symbols. The Z notation has its own syntax in which every line has a mathematical meaning. The SA1 safety system has many schemas to show how the different parts of the system work. There are mathematical symbols to represent a real world actions. The specification simply tells us the result not the method of staff entering and exiting the building. There are some verifications to check if the schemas are correct, for example when adding new staff.

In order to progress to the next level, formal specification and formal verification, formal proofs need to be constructed this can be done by model checking or theorem proving. Computer based tools can be used to aid the process. There are different types of tools that can be used to construct the proof depending on the complexity needed for the SA1 Site Safety System. The model checking is sufficient as this is faster and easier to use.[2]

Rushby divides the formality of the specification into four different levels. This goes from level 0 which is no formal method to level 3 which is a formal specification language that uses a comprehensive support for all environments. The SA1 site safety system is on a level 2 which is a formalised specification language with some forms of automation.[3] The SA1 site safety system is a level 2 because it uses a specification language that has a syntax checker and prettyprinters. It also has checking tools that detect different types of faults. There are examples of propositional logic and first-order logic in the SA1 Site Safety System for example the remove staff error which shows there being a logical statement to determine the record to be removed from a set of data is a first-order logic. In order to progress up a level there needs to be higher-order logic which is more complex than first-order logic. Higher-order logic has individual and relational variables which are defined.[4] To change levels there needs to be either a proof checker or a theorem prover to check the logic in the system. A proof checker is a program that checks the proof entered by the user. A theorem prover is a program that does this automatically without user intervention.

References

- [1] D. E. Lightfoot, Formal specification using Z. Palgrave, 2001.
- [2] A. E. Haxthausen, An Introduction to Formal Methods for the Development of Safety-critical Applications, pp. 132, 2010.
- [3] J. Rushby, Formal methods and the certification of critical systems, Sri-Tr Csl-93-7, no. December, 1993.
- [4] J. B. Almeida, M. J. Frade, J. S. Pinto, and S. Melo de Sousa, Rigorous Software Development. 2011.