International Journal of the Physical Sciences Vol. 6(31), pp. 7262 - 7270, 30 November, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.1246
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

Extending promotion to operate controller based on train's operation

Sher Afzal Khan[1]*, Nazir Ahmad Zafar[2] and Farooq Ahmad[1]

[1]Faculty of Information Technology, University of Central Punjab Lahore, Pakistan.
[2]Department of Computer Science, College of Computer Science and Information Technology, King Faisal University, Al Hassa, Saudi Arabia.

Accepted 26 September, 2011

Previous work described by the authors was the use of promotion to link the global state (controller) with the local state (train system). In which the operation of a train makes changes in controller, based on the approach of promotion. However, the model did not apply to any of the critical interlocking components. This paper extends to provide safety along the critical components, that is, when train moves along the straight track or passing through a crossing or level crossing. For the safety of a train a safe distance (moving block) is associated with every train to avoid collision. Similarly, to define the safe boundaries of crossing and level crossing, circular blocks are defined which has two states, green when it is free and red when it is occupied. The syntax and type of the formal model is checked by using the Z/EVES tool.

Key words: Formal methods, Z-notation, promotion, moving block railway interlocking, safety properties.

INTRODUCTION

The train's weight in thousands of tons, great length, iron track, distributed environment and presence of track components, makes the moving block interlocking a very complex system for its modeling. It is a safety, monetary and environmental critical system, because its failure may cause the loss of human life, severe injuries, loss of money and environmental issues. To ensure the safety, quality and reliability of a distributed railway interlocking system, it requires formal modeling and step by step design processes. According to the European committee of electro technical standardization (European committee, 2001), the use of formal methods in the development of safety-critical and computer-controlled systems for railways applications is recommended. To avoid collision and other conflicts, there is a mechanism that transforms the message from a signalman of the rail network to the rail team, that is, to either pass or stop a train is called the signaling system. The interlocking is an arrangement of signals that prevents trains from collision along railway components: crossings, level crossings and switches.

The signaling and track arrangements are collectively called interlocking system. The purpose of railway interlocking system is to prevent trains from collisions and derailing, while at the same time allowing normal train's movement. There are two main existing technologies of railway interlocking systems: fixed block and moving block interlocking. Moving block is getting more importance in the railway industry, because of some disadvantages in the fixed block interlocking system (Khan and Zafar, 2011).

There exists much work on modeling of the interlocking system. The work (Akillilar and Koppe, 2008; Calame, 2007; Daliri et al., 2011, Guo et al., 2007; Hei et al., 2007; Lim, 2008; Khan et al., 2011; Yavuz, 2011) and a list (Bjørner, 1998) of 299 publications, addressing various issues on this topic, proves its importance.

Most of the publications are on the traditional fixed block, whereas this work is on the moving block interlocking which reflects a different approach. The work of Simpson (1999) is the start of moving block concept using Z-notation, communicating sequential processes (CSP) and failures-divergence refinement (FDR2). However, the author does not address the safety properties of the interlocking system. Further, Zafar (2006, 2009)

*Corresponding author. E-mail: sherafzal@ucp.edu.pk.

extends the existing work towards the safety properties of moving block railway interlocking system (MBRIS). For example, the work of Zafar (2006) describes a specification of the safety properties based on the model in graph theory. Moreover, Zafar (2009) specifies the critical components: tracks, switches, crossings and level crossings of moving block interlocking system by using an integration of graph theory and Z-notation. However, his specifications are split in many schemas which create difficulties in understanding. Also he defines only the safety properties with no any operation to perform action to change the state in case of collision or derailing. This paper extends our work (Khan and Zafar, 2007) in which we used the approach of promotion to specify the control of moving block interlocking system. Promotion was used to link the train system with the corresponding control system that is associated in every sector using Z-notation. However, the safety along critical components was not specified. Here, we link control system with train through promotion to provide safety along the linear movement, crossing and level crossing. Promotion allows us to links certain changes in control state based on changes in the state of train system, which is passing through crossing, level crossing or moving straight. In our specification, we consider control system as global operation and train system as local operation and the schema ‘promote’ makes changes in global state based on local state.

FORMAL METHODS

The advancements of the computing, control and communication technologies in the 21st century have increased the complexity of hardware and software systems, such as computing critical systems, automated systems, networked or embedded systems, etc. Moreover, the functionality and scale of such systems have also been grown. The likelihood of errors and flaws in the systems becomes greater and design of ultimate bug-free systems becomes more difficult, because of the increase of complexity. Design of such a complex system can be possible with the help of advanced mathematical methodologies and step by step design process. Consequently, the issue of system's complexity attracts the attention of researchers towards effective methods of modeling and verification of systems. One of the best ways of achieving this goal is by using the formal methods. Formal methods are languages which are dependent on the use of mathematical procedures and notations to describe and analyze the properties of software systems (Wing, 1999). These descriptions and notations are usually from the area of discrete mathematics, including set theory, predicate logic and graph theory. These mathematical notations reduce inconsistencies, ambiguities and incompleteness from the software system. The procedure by Liu and Adams (1995) of software development based on formal methods is described in Figure 1. Software life cycle model based on `refinement' is the result of a requirement analysis which is usually expressed in everyday language. `Specification 1' describes the formal requirements which are transformed from the informal language. Further, the development from `specification '1' to `specification n' describes the process of design. The two fundamental principles that arise in the development of a software system are the validation and verification. Validation describes whether the generated system accomplishes the requirements, whereas verification ensures whether the created software meets the requirements founded in the previous stage. The purpose of this approach is to exhibit the maturity of a system from informal requirements to coding. The benefit of this is to recognize faults and misunderstanding in the initial stage of the software design life cycle. These errors can be removed easily from the software system, consequently producing a high class system with cost reduction.

SPECIFICATION OF THE MOVING BLOCK INTERLOCKING SYSTEM

Here, we use Z-notation to describe the safety system of the moving block interlocking.

Fundamentals of the systems

In the moving block interlocking system, every train has a moving block. However, crossing and level crossings are enclosed by the static circular blocks. In our specification, every sector is controlled by a computerized control system. The schema ‘promote’ alert the controller based on trains operations in the sector. When a train approaches to a network component, state of the controller is changed. A train can enter into a component when the state of the component is clear. A state of the component is occupied when the intersection of a component region with a moving block of a train is non empty. On the other hand, state of a component is clear when there is no moving block in its region. For example, train T1 passes through crossing and level crossing as shown in the Figure 2. If the moving block represented by B1 intersects the circular region of railway crossing, control alerts the train about the state of the crossing. If the state is cleared, then the train is allowed to pass the crossing. If it is occupied then train is required to stop before the point of intersection. For a safety along a level crossing the barrier must be closed for traffic when a moving block intersects the level crossing region.

The specification of the system uses *[Trains, ID],* where *Trains* is the set of all trains and ‘*ID’* is the set of identity numbers associated with trains in the system. The paper

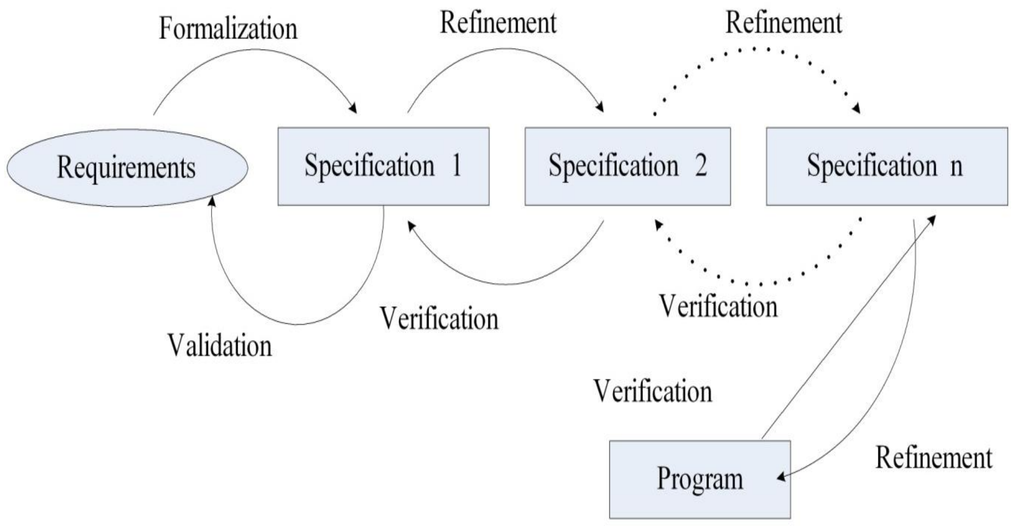


Figure 1. The process of software development using formal methods.

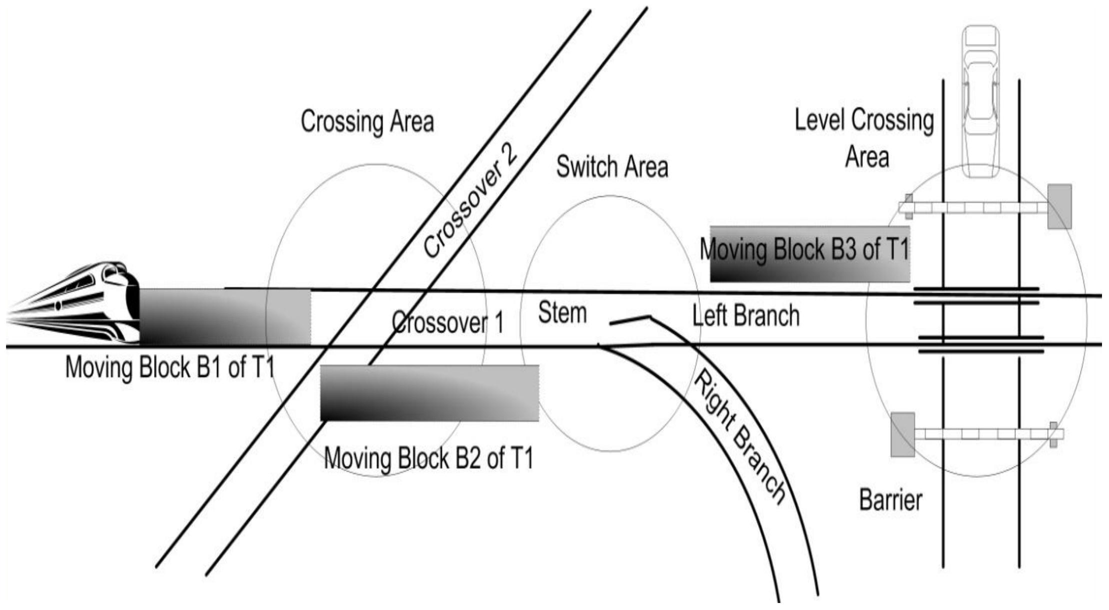


Figure 2. Sector of moving block interlocking system.

defines *[Speed, movingBlocks]* where the *Speed* is a set of type natural numbers that shows the speed of a train.

The *movingBlocks* is the set of track segments which depends on the speed of a train, that is, moving block is of maximum length if the speed is maximum and of minimum length if the speed is minimum. The segment is the set of ordered pair of points. Where point is the identification spot on a railway track and the set of all points on the railway track can be represented by the set *[Points]*. Another given type is the set *[Sectors]*, the set of segments defined as sector of type (Points × Points). The set of all crossing is denoted by the set *[Crossings].* The identification numbers associated with every crossing is representing by the set *[IDCrossing]*. Every sector is controlled by a computer based control system. In this paper, the schema *Promote* is used to link the schema *TrainSystem* with the schema *ControlRoom*. The schema *Promote* continuously updates the schema control room about trains operations in the corresponding sector.

Specification of the system

Here, we specify the train system, control system, static system, promotion and operations to avoid collisions along the linear track, crossing and level crossing using Z-notation.

Train system

The specification of schema *TrainSystem* requires the following sets: [*Points, ID, Sectors, Trains, movingBlocks*]. The specification is defined the injective functions *trainIdentifier* from the set *Trains to the set ID.* This shows the unique identification of a train in the system.

TrainSystem

$trainIdentifier: Trains \rightarrowtail ID$
$trainSpeed: ID \rightarrow \text{Speed}$
$blockSize: \text{Speed} \rightarrow movingBlocks$
$location: movingBlocks \rightarrow \mathbb{P}\ (Points \times Points)$
$distribuation: \mathbb{P}\ (Points \times Points) \rightarrow \mathbb{P}\ Sectors$
$control: \mathbb{P}\ Sectors \rightarrow ControlRoom$

The function *trainSpeed* is defined from the set *ID* to the set *Speed*; it gives the corresponding speed of any train moving in a sector under the control of a control system. The *blockSize* is a function from the set *Speed* to the set *movingBlocks*. It describes the size of moving block associated with a train depending on its speed. The function *location* represents the division of a moving block into the identified set of segments. This shows the identification of a location of a train in the sector. The distribution is a function between the set of segments and the set of sectors. This provides information about sectors having occupied segments. Finally, *control* is the function from the set of sectors to the schema *ControlRoom*. This continuously informs the *ControlRoom* about a status of a train in a related sector.

Control system

The control system is represented by the state schema *ControlRoom*. The variables, *sectorStatus* and *CircularBlockStatus* describe the status of sector and of circular blocks, respectively. The variable *sectorOccupiedby* is of type *P (Trains × ID).* This provides the identification of train which occupied the corresponding sector. The variable *segmentsOccupied* is of type *P (Points × Points)*. This shows the set of all segments occupied by a particular train in the corresponding sector. The *segmentOccupiedby* is declared as of type *(Trains × ID)*, and it provides an identification of a train that occupied the set of segments. The declaration *traininRedZone* represents the set of segments occupied by a train. *RedZoneduetoTrain* is the identification of train that occupied the red zone. *DirectionofRedZone* represents the direction of a red zone.

ControlRoom

$sectorStatus,\ LevalCrossingStatus, CrossingStatus: Reports$
$CircularBlockStatus: Reports$
$segmentsOccupied:\ \mathbb{P}\ (Points \times Points)$
$sectorOccupiedby: \mathbb{P}\ (Trains \times ID)$
$segmentOccupiedby: Trains \times ID$
$traininRedZone: \mathbb{P}\ (Points \times Points)$
$RedZoneduetotrain: Trains \times ID$
$directionofRedZone: Directions$

Static system

This schema *CrossingSystem* includes *Levcrossidentifier,* an injective function that provides a unique identification of crossing. The function *circularblck* provides the static circular block associated with every crossing. The *radius* of the circular block is the set of segments required for any train to stop before reaching the point of crossing.

CrossingSystem

$levcrossidentifier: Crossings \rightarrowtail IDCrossing$
$circularblock: IDCrossing \rightarrow CircularBlocks$
$circularSegments: CircularBlocks \rightarrow \mathbb{P}\ (Points \times Points)$

The schema *LevelCrossingSystem* includes *Levcrossidentifier*, an injective function from the set of level crossings to the set *IDLevel*. This provides a unique identification of a level crossing. The function *staticblock* is associated with every level crossing used for a safety of the level crossing. Every static block is defined to be a circular of radius equal to the maximum braking distance of a train.

LevelCrossingSystem

$levcrossidentifier: LevelCrossings \rightarrowtail IDLevel$
$staticblock: IDLevel \rightarrow StaticBlocks$
$coveredSegments: StaticBlocks \rightarrow \mathbb{P}\ (Points \times Points)$.

The schema *SectorSystem* defines the relation denoted by *signal* used to define any signal with a train moving in the sector. The *timeinSector* is the relation defined between the set *(Sectors × movingBlocks)* and the set *Time*. This provides *time* of a moving block in a sector.

SectorSystem

$signal: ID \leftrightarrow Reports$
$timeinSector: Sectors \times movingBlocks \leftrightarrow \text{Time}$
$direction: movingBlocks \leftrightarrow Directions$
$GateSignal: IDLevel \rightarrow Reports$

The direction is the relation that is defined between the moving block associated with a train in a sector, and the set *Direction* provides the direction of any train moving in a sector.

Promotion

Promotion allows us to compose and factor the specifications. It has also been called framing, because it is evocative of placing a frame around part of a specification: only what is inside the frame may change; what is outside must remain unaffected as in Spivey (1992). The *train?, id?* and *speed?* are the input for the system that is associated with a train entered into a related sector having the *movingblock?* The question mark '?' with a variable represents the input to the system. The *directionofTrain?* is of type *Directions* shows the direction of the incoming train in sector. To inform the interlocking system continuously about the occupied sectors, segments, speed and an identification of a train, we use the input variables: *sectorsOccupied?, segmentsOccupied?, speed?*, *train?* and *id?* All these functionality are operated continuously by the schema signature of schema *Promote*. The schema *Promote* expresses the relationship between state schemas *TrainSystem* and *ControlRoom*. The schema *Promote* checks the identification of each sector and then link the schema *TrainSystem* with schema *ControlRoom* by the relation. The *sectorOccupied?* is the set of sectors occupied by the moving block of train that depends upon train's speed, direction and segment occupied.

```
Promote
ΔTrainSystem
ΔControlRoom
id?: ID
train?: Trains
speed?: ℤ
movingblock?: movingBlocks
directionofTrain?: Directions
trainLocation?: ℙ Sectors
segmentsOccupied?: ℙ (Points × Points)
sectorsOccupied?: ℙ Sectors
──────────
(train?, id?) ∈ trainIdentifier
(id?, speed?) ∈ trainSpeed
(speed?, movingblock?) ∈ blockSize
(movingblock?, segmentsOccupied?) ∈ location
(segmentsOccupied?, sectorsOccupied?) ∈ distribuation
θ ControlRoom = control sectorsOccupied?
control′ = control ⊕ {(sectorsOccupied? ↦ θ ControlRoom)}
```

θControlRoom=control(sectorsOccupied?)

Avoiding collision along the linear motion

The schema *ReceiveAndSendSignals* is the state schema that needs the input variables *train?*, *mb?, speed?* and *segoccupied?* And it represents train, moving block, speed and the segments occupied by the train, respectively.

```
ReceiveAndSendSignals
train?: Trains, mb?: movingBlocks
movingBlockdueTo?: ID × Trains
speed?, maxSpeed, minSpeed: ℤ
segOccupied?: ℙ (Points × Points)
```

Clear status of the system along the straight motion

The schema *ClearStatus* includes the schemas Ξ*ReceiveAndSendSignals* and Ξ*TrainSystem*. The notation Ξ with schema provides all the functionality to the state schema with no change in its state. It also includes the schema *ControlRoom* with notation Δ which means that schema could be utilized whenever the system required describing an operation that may change the state of the schema *ControlRoom*.

```
ClearStatus
ΞReceiveAndSendSignals
ΞTrainSystem, ΔControlRoom
movingblock: movingBlocks, sectorStatus!
Free:ReportsContinueGreenZone: Reports
──────────
(mb?, segOccupied?) ∈ location
(∀movingblock: movingBlocks • location movingblock ∩
            location mb? = ∅)
⇒ sectorStatus! = Free
    ∧ signal′ (trainIdentifier train?) =
                    ContinueGreenZone
sectorOccupiedby′ = sectorOccupiedby
segmentOccupiedby′ = segmentOccupiedby
traininRedZone′ = traininRedZone
RedZoneduetotrain′ = RedZoneduetotrain
directionofRedZone′ = directionofRedZone
LevalCrossingStatus′ = LevalCrossingStatus
```

The predicate part of the schema exhibits that whenever the intersection of a moving block of a train with the moving block of any other train is empty, it delivers a signal of green zone to the train. The remaining binding of the schema *ControlRoom* remains unchanged.

Providing safety to trains facing the red zone

The operation schema *LinearSafetyRed* includes the schema Ξ*SectorSystem,* Ξ*TrainSystem* and ∆*ReceiveAndSendSignals*. Its executions depend on the intersection of moving block mb? of a train with the moving block of any other train along a linear movement. If the intersection is nonempty along the same direction, it changes the state of schema *ControlRoom,* which informs both trains about the occupied status of the sector. For safety requirements, it signals the behind train to stop and in front to increase its speed.

LinearSafetyRed

$\Delta ReceiveAndSendSignals$
$\Xi SectorSystem, \Xi TrainSystem, \Delta ControlRoom$
$timeinsect?: Time\ , sector?: Sectors$
$movingblock: movingBlocks, speedofmb?: Speed$
$directionofRedZone: Directions$
$sectorStatus!, SignalforTraint!: Reports$
$StoptrainRedZone, IncreasetheSpeed, Occupied: Reports$

$(mb?, segOccupied?) \in location$
$\exists t: Trains$
$\quad \bullet\ location\ (blockSize\ (trainSpeed\ (trainIdentifier\ t))) \cap$
$\qquad location\ mb? \neq \emptyset$
$\quad \wedge\ timeinsect?$
$\qquad > timeinSector\ (sector?,$
$\qquad\quad (blockSize\ (trainSpeed\ (trainIdentifier\ t))))$
$\quad \wedge\ speedofmb? < trainSpeed\ (trainIdentifier\ t)$
$\Rightarrow sectorStatus' = Occupied$
$\quad \wedge\ segmentOccupiedby' = (t, trainIdentifier\ t)$
$\quad \wedge\ signal'\ (trainIdentifier\ t) = StoptrainRedZone$
$\quad \wedge\ signal'\ (trainIdentifier\ train?) = IncreasetheSpeed$
$\quad \wedge\ SignalforTraint! = StoptrainRedZone$
$\quad \wedge\ directionofRedZone'$
$\quad = direction\ (blockSize\ (trainSpeed\ (trainIdentifier\ t)))$

The safety of a train along the linear motion is specified by the following relation.

$$\Delta TotalLinearSafety \ \widehat{=}\ LinearSafety \wedge LinearSafetyRed$$

Avoiding collision along railway crossing

The *CrossingIdentification* is the state schema with input variables *idcrossing?* and *cirblock?* of type *IDCrossing* and *CircularBlocks*, respectively. The *cirblock?* Denotes the static circular block which enclosed the crossing, it has a radius of length equal to the maximum braking distance of a train.

CrossingIdentification

$\Delta ReceiveAndSendSignals$
$\Xi CrossingSystem, \Xi TrainSystem$
$\Delta ControlRoom$
$idcrossing?: IDCrossing, cirblock?: CircularBlocks$

Safety along crossing

The schema *SafetyOnCrossing* includes *CrossingIdentification*, the variable *mb?* The specification reveals, if the intersection of any moving block with a circular block is non empty, then the circular block becomes a red zone for any other train in the system and the control delivers the signals of red zone to the incoming trains. In latter case, if the intersection of the circular block with a moving block is empty, then the crossing is a green zone for the incoming train.

SafetyOnCrossing

$\Delta CrossingIdentification$
$mb?: movingBlocks$
$CircularBlockStatus!, RedZoneForEveryTrain: Reports$
$Signalformb, StopTillTheCrossingClear: Reports$

$(\exists movingblock: movingBlocks \bullet\ location\ movingblock \cap$
$\qquad circularSegments\ cirblock? \neq \emptyset)$
$\wedge\ circularSegments\ cirblock? \cap location\ mb? \neq \emptyset$
$\Rightarrow CircularBlockStatus' = RedZoneForEveryTrain$
$Signalformb = StopTillTheCrossingClear$
$sectorStatus' = sectorStatus$
$sectorOccupiedby' = sectorOccupiedby$
$segmentOccupiedby' = segmentOccupiedby$
$traininRedZone' = traininRedZone$
$RedZoneduetotrain' = RedZoneduetotrain$
$directionofRedZone' = directionofRedZone$

CrossingGreenZone

$\Delta CrossingIdentification$
$mb?: movingBlocks$
$CircularBlockStatus!, Clear: Reports$
$Signalformb, ContinueGreenSignal: Reports$

$\exists movingblock: movingBlocks \bullet$
$location\ movingblock \cap circularSegments\ cirblock? = \emptyset$
$\quad \wedge\ circularSegments\ cirblock? \cap location\ mb? \neq \emptyset$
$\quad \Rightarrow CircularBlockStatus! = Clear$
$Signalformb = ContinueGreenSignal$
$sectorStatus' = sectorStatus$
$sectorOccupiedby' = sectorOccupiedby$
$segmentOccupiedby' = segmentOccupiedby$
$traininRedZone' = traininRedZone$
$RedZoneduetotrain' = RedZoneduetotrain$
$directionofRedZone' = directionofRedZone$

The total safety along the crossing is given by the schema *TotalCrossingSafety* in the following relation:

$$(\Delta TotalCrossingSafety \mathrel{\widehat{=}} SafetyOnCrossing \land CrossingGreenZone)$$

Safety along the level crossing

The schema *SafetyatLevelCrossingGreen* includes schemas *ControlRoom, SectorSystem, TrainSystem, ReceiveAndSendSignals* and *LevelCrossingSystem*. This schema is activated when the intersection of the static block of level crossing with a moving block is empty. The predicate part generates the output reports of sector and level crossing status and commands of open the barrier and green the signal for the road traffic. The schema does not change the state of schema *ControlRoom.*

$$
\begin{array}{l}
\underline{\quad SafetyatLevelCrossingGreen \quad} \\
\Delta ReceiveAndSendSignals \\
\Xi LevelCrossingSystem \\
\Xi SectorSystem \\
\Xi TrainSystem \\
\Delta ControlRoom \\
\quad StaticBlocks \\
sectorStatus!: Reports \\
GreenSignalForTraffic, LevalCrossingStatus!: Reports \\
GateSignal!, OpenGate: Reports \\
\hline
(level?, idlevel?) \in levcrossidentifier \\
(idlevel?, stblock?) \in staticblock \\
(\forall movingblock: movingBlocks \bullet\ location\ movingblock \cap \\
\quad coveredSegments\ stblock? = \emptyset) \\
\Rightarrow LevalCrossingStatus! = GreenSignalForTraffic \\
\quad \land GateSignal'\ idlevel? = OpenGate \\
sectorStatus' = sectorStatus \\
sectorOccupiedby' = sectorOccupiedby \\
segmentOccupiedby' = segmentOccupiedby \\
traininRedZone' = traininRedZone \\
RedZoneduetotrain' = RedZoneduetotrain \\
directionofRedZone' = directionofRedZone \\
LevalCrossingStatus' = LevalCrossingStatus \\
\hline
\end{array}
$$

$$
\begin{array}{l}
\underline{\quad SafetyLevelCrossingRed \quad} \\
\Delta SafetyatLevelCrossingGreen \\
RedSignalForTraffic, LevalCrossingStatus!: Reports \\
SignalGateClosed: Reports \\
SignalGateOpen, delay, timeInterval: Reports \\
\hline
(level?, idlevel?) \in levcrossidentifier \\
(idlevel?, stblock?) \in staticblock \\
(\exists openblock: openBlocks \bullet\ location\ openblock \cap \\
coveredSegments\ stblock? \neq \emptyset) \\
\land GateSignal\ idlevel? = SignalGateOpen \\
\Rightarrow LevalCrossingStatus! = RedSignalForTraffic \\
\quad \land delay = timeInterval \\
\quad \land GateSignal'\ idlevel? = SignalGateClosed \\
sectorStatus' = sectorStatus \\
sectorOccupiedby' = sectorOccupiedby \\
segmentOccupiedby' = segmentOccupiedby \\
traininRedZone' = traininRedZone \\
RedZoneduetotrain' = RedZoneduetotrain \\
directionofRedZone' = directionofRedZone \\
\hline
\end{array}
$$

Whereas, the schema *SafetyLevelCrossingRed* is activated when the train's moving block intersects the static block of a level crossing. It reads the signal to a road traffic and closed the barrier after a defined fixed interval.

The safety of a train along the level crossing is specified by the following relation.

$$(\Delta TotaLevelCrossingSafety \mathrel{\widehat{=}} SafetyLevelCrossingRed \land SaftylevelCrossingGreen)$$

The total safety of a train is given by the specification:

$$(\Delta TotalSafety \mathrel{\widehat{=}} \Delta TotalLinearSafety \land \Delta TotalLevelCrossingSafety \land \Delta TotalCrossingSafety)$$

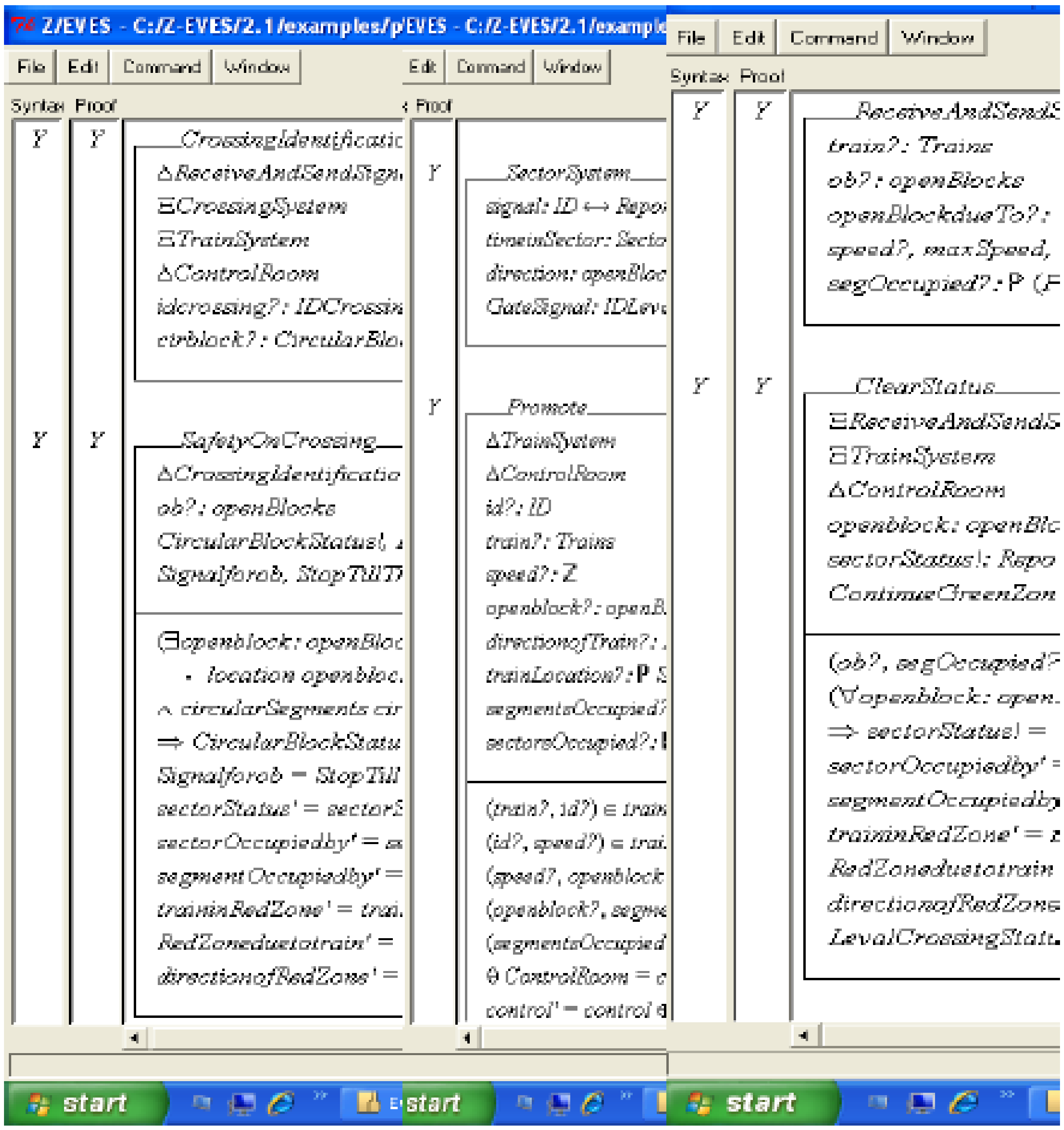
Specification analysis

Z is one of the formal methods, which is widely used in system development. Our experiences of applying this formal approach in modeling the system are the as follows:

1. This specification has supported us to know the complete understanding of the system due to step by step mathematics. Also this helps us to remove ambiguities and inconsistencies in the definitions.
2. We were able to do systematic testing of syntax and proof of the specification, using Z-EVES which is a mathematical toolkit developed by ORA Canada used for checking syntax and proof of the specification (Meisels and Saaltink, 1997). The use of Z-EVES ensures the correct syntax. The verified snapshots of our specification are given in Figure 3, which provide the accuracy of syntax and proof of the specification flow given in this paper.

Conclusion

In this specification, computer based control system was introduced using the approach of promotion, for preventing collision and allowing normal trains movement. This paper developed a system which provides safety to train system along the straight movement, crossing and level crossing. Two benefits were achieved: first, this research showed the successful use of formal methods for designing an industrial complex system; the second and the most important benefit shows that the complexity of the system was reduced and it was easy to see the control sequence of important properties required for implementation phase. The specification has used an approach of promotion which makes changes in control system based on the movement of a train in the train system. It has shown


Z/EVES - C:/Z-EVES/2.1/examples/p/EVES - C:/Z-EVES/2.1/example
File
Edit
Command
Window
Syntax
Proof
start

Y Y CrossingIdentificatio
ΔReceiveAndSendSign
ΞCrossingSystem
ΞTrainSystem
ΔControlRoom
idcrossing? : IDCrossin
cirblock? : CircularBlo

Y Y SafetyOnCrossing
ΔCrossingIdentificatio
ob? : openBlocks
CircularBlockStatus!,
Signalforob, StopTillT
(∃openblock : openBloc
• location openbloc
∧ circularSegments cir
⇒ CircularBlockStatu
Signalforob = StopTill
sectorStatus' = sectorS
sectorOccupiedby' = se
segmentOccupiedby' =
traininRedZone' = trai
RedZoneduetotrain' =
directionofRedZone' =

Y SectorSystem
signal: ID ↔ Repo
timeinSector: Secto
direction: openBloc
GateSignal: IDLeve

Y Promote
ΔTrainSystem
ΔControlRoom
id? : ID
train? : Trains
speed? : ℤ
openblock? : openB
directionofTrain? :
trainLocation? : ℙ S
segmentsOccupied?
sectorsOccupied? : ℙ
(train?, id?) ∈ train
(id?, speed?) ∈ trai
(speed?, openblock
(openblock?, segme
(segmentsOccupied
θ ControlRoom = c
control' = control ⊕

Y Y ReceiveAndSendS
train? : Trains
ob? : openBlocks
openBlockdueTo? :
speed?, maxSpeed,
segOccupied? : ℙ (F

Y Y ClearStatus
ΞReceiveAndSendS
ΞTrainSystem
ΔControlRoom
openblock : openBlc
sectorStatus! : Repo
ContinueGreenZon
(ob?, segOccupied?
(∀openblock : open
⇒ sectorStatus! =
sectorOccupiedby' =
segmentOccupiedby
traininRedZone' = t
RedZoneduetotrain
directionofRedZone
LevelCrossingState

Figure 3. Syntax and proof of the specification.

that: at most one train is allowed at crossing at one time; every segment must not be occupied by two train's moving blocks; and when train's moving block intersect a level crossing region, then barrier must be closed. Since specified model is not for a particular interlocking system. It's a general approach to specify railway control of moving block interlocking system.

REFERENCES

Akillilar M, Koppe C (2008). Increase of effectiveness and efficiency in the inter locking system ESTW L90 5. Signal und Draht, 100(4): 26-30.

Bjørner D (1998). The FME rail annotated rail bibliography. Department of Information Technology, Technical University of Denmark.

Calame JR, Goga N, Ioustinova N, Pol JVD (2007). TTCN-3 testing of hoorn-kersenboogerd railway interlocking. Canadian Conference on

Electrical and Computer Engineering, CCECE '06: 620-623

Daliri ZS, Shamshirband S, Besheli MA (2011). Railway security through the use of wireless sensor networks based on fuzzy logic. Int. J. Phy. Sci., 6(3): 448-458.

European Committee Electro Technical Standardization (2001). Railway applications communications, signaling and processing systems software for railway control and protection systems.

Guo J, Huang Z, Liu, M (2007). Research on the railway safety critical system with Petri nets. Proceedings of 6th International Conference on ITS Telecommunications, China, pp. 118 -121

Hei X, Takahashi S, Nakamura H (2007). Modeling and evaluation of a component-based distributed railway interlocking system using Petri nets. Report of the Re-search Institute of Science and Technology, Nihon University: pp.43-46.

Khan SA, Zafar NA (2007). Promotion of local to global operation in train control system. J. Digital. Inform. Manage., 5(4): 231-236.

Khan SA, Zafar NA (2011). Improving moving block interlocking system using fuzzy multi agent specification language. Int.l J. Innovative Comput. Inform. Control, 7(7B): 4517-4534.

Khan SA, Zafar NA, Ahmad F (2011). Petri net modeling of railway crossing system using fuzzy brakes. International J. Phy. Sci., 6(14): 3389–3397.

Lim M (2008). Lifecycle-based functional interlocking specification. Signal und Draht., 100(4): 37-40.

Liu S, Adams R (1995). Limitation of formal methods and an approach to improvement. Technical report, Hiroshima City University.

Meisels I, Saaltink M (1997). The Z/EVES reference manual. TR-975493-03, ORA, Canada.

Simpson A (1999). Towards the mechanical verification of moving block signaling systems. Technical report, Oxford Brookes University.

Spivey JM (1992). The Z Notation: A reference manual (2nd edition ed.). Prentice Hall International Series in Computer Science.

Wing JM (1990). A Specifier's. Introduction to Formal. Methods. IEEE Computer, 23(9): 8-24.

Yavuz E (2011). Istanbul (Bosphorus) immersed tube tunnel and survey works. Int.J. Phy. Sci., 6(9): 2303–2307.

Zafar NA (2006). Formal model for moving block railway interlocking system based on un-directed topology. In Int. Conf. Emerging Technol., pp. 217-223.

Zafar NA (2009). Formal specification and validation of railway network components using Z-notation. Software, IET., 3(4): 312-320.