Formalizing Moving Block Railway Interlocking System for Directed Network

Zafar, Nazir A.
Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University : Graduate Student

Araki, Keijiro
Department of Computer Science and Communication Engineering, Faculty of Information Science and Electrical Engineering, Kyushu University

https://doi.org/10.15017/1515841

出版情報：九州大学大学院システム情報科学紀要. 8 (2), pp.109-114, 2003-09-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第8巻 第2号 平成15年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.8, No.2, September 2003

Formalizing Moving Block Railway Interlocking System for Directed Network

Nazir A. ZAFAR* and Keijiro ARAKI**

(Received June 13, 2003)

Abstract: The safety and complexity of Railway Interlocking System (RIS) requires the use of advanced methodologies. Formal methods increase quality and provide highest confidence in this area. In this paper, safety analysis of moving block RIS is presented. The system is decomposed into four components, i.e., network topology, network state, controls and trains. The formal analysis of the components is presented after further decomposition. Finally, the safety requirements, no collision and no derailing, are defined abstractly and then refined by integrating with the notion of moving block. The railway network is modeled using directed graph. Formal specification is described in VDM-SL.

Keywords: Formal methods, Graph theory, Railway Interlocking System, Directed network, Moving block, Safety analysis, VDM-SL

1. Introduction

Railway Interlocking System (RIS) is a safety critical system, so there is a need to ensure that the system prevents dangerous situations [7]. Formal methods, because of tool support, increase quality and provide highest confidence in this area [6]. Primarily, the task of RIS is preventing trains from collisions and derailing. There are two existing technologies, fixed block and moving block, for RIS. Because of some disadvantages in fixed block RIS, the moving block RIS is getting important.

Most of the researchers have their publications on fixed block RIS while this research is for moving block RIS with a different approach. The work [2] of A. Simpson is close to ours but he uses Z, CSP and FDR2, and his work is the starting point in this area. Hansen [7] uses VDM [12] to model concepts of railway topology, and safety analysis is presented but again his work is for fixed block interlocking. A. E. Haxthausen [1] described elegant formal representation of a distributed railway control using RAISE, the system described therein is not, in a strict sense, a moving block interlocking. Some work of interest is also reported [4),8),5].

Our previous work [9),10] was for bi-directional railway network while this work is for directed network. This work is one step forward to develop an abstract model for dynamic topology, i.e., a topology in which at some parts one way movement while at other parts both way movement of train is possible and direction of any track segment in topology can also be changed if required.

There are three main objectives to be achieved in the paper: (i) applying formal methods in RIS, (ii) integration of formal and informal approaches and (iii) safety analysis preventing collisions and derailing, and allowing normal trains movements.

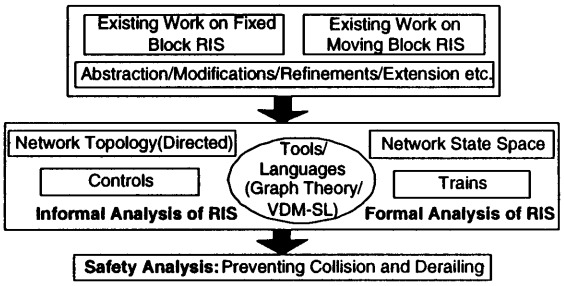

Fig.1 Approach to Construct Formal Model.

Figure 1 shows our approach. The system consists of network topology, network state, controls and trains. The formal analysis of the components is presented after the further decomposition. In our topology, it is supposed that if a train can move from one track segment to another it can not move in the opposite direction and as a result the topology is a directed graph. That is why ordered pairs are used to represent the connectivity of track segments. In the model topology, a track segment is represented by node and the connectivity of two track segments shows the permission for a train to move from one track segment to the other. The state space of the components is analyzed. The controls which are responsible for observing trains and network state are formalized. Finally, the components are composed to define the formal model for the whole system.

The abstract safety properties, no collision and no derailing, are formalized. It is supposed that there will be no collision if there is one train at a network component. Further, the direction of train and switch control must be same preventing derailing. In the refinement, consistency between the moving block and directed topology is analyzed.

* Department of Computer Science and Communication Engineering, Graduate Student

** Department of Computer Science and Communication Engineering

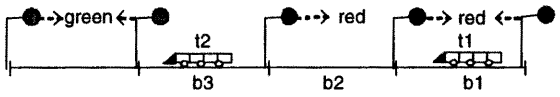

Fig.2 Fixed Block Interlocking System.

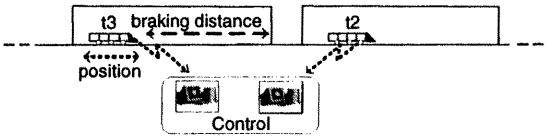

Fig.3 Moving Block Interlocking System.

It is assumed that the system will be safe if the moving block respects the state of network. The formal specification is described in VDM-SL [12].

In section 2, an introduction to RIS is given. In section 3, real topology is transformed to model topology. In section 4, formal model is presented. Safety properties are formalized in section 5. Conclusion and future work are discussed in section 6.

2. Railway Interlocking Systems

2.1 Fixed Block Interlocking

In fixed block RIS, the railway network is divided into fixed blocks which are separated by signals as shown in **Fig. 2**. At one time, only one train can move in a block and can enter into a block only if the next is clear. For example, in **Fig. 2**, the train t1 can enter block b2 only when train t2 has cleared the block b3. This means that there is always a distance of more than one block between two trains.

2.2 Moving Block Interlocking

In reality, the safe distance between two trains is the distance needed for a train to come to a complete stop which is much less than the length of a fixed block. The idea of moving block is based on this concept, i.e., keeping only safe distances between trains. Instead of cutting piece of railway line into fixed blocks, train's position and some distance in front of it becomes the moving block in which no other train can enter. For example, intersection of moving blocks of trains t2 and t3 is always empty for safe operation of the trains as in **Fig. 3**, which are observed by a computer based control system.

3. Model in Graph Theory

3.1 Real Topology

Real topology is composed of track segments as in **Fig. 4**. A track segment is switch if it is related with three track segments otherwise a linear segment. Railway crossing is composed of two linear segments having a certain relation. It is supposed without loss of reality that segment is small enough. Due to this assumption, for simplicity of the model, it is always possible to decompose a network in such a way that no two switches or crossings are connected physically. Moreover, it is assumed that a switch or a crossing is not at the end of topology.

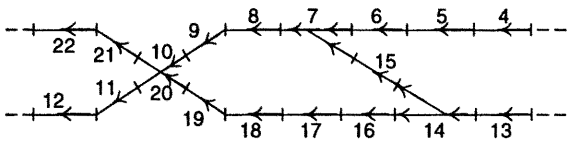

Fig.4 Real Topology.

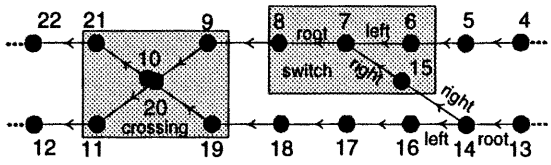

Fig.5 Model Topology.

3.2 Model Topology

A track segment in real topology is represented by a node in model topology in **Fig. 5**. The connectivity of track segment s1 with track segment s2 means that a train can move from s1 to s2 directly and is denoted by (s1, s2).

4. Formal Model

Formal model consists of network topology, network state, trains and controls. A brief informal description of definitions is given following the formal analysis of the components.

4.1 Directed Network Topology

First, the formal specification of connections, switches and crossings is described. And then, we composed these components to define the model topology.

4.1.1 Connections

Connections is defined by a relation on the track segments of directed topology. The interpretation of the relation is that each pair of track segments in the relation is physically connected. The Connections relation is modeled by a graph in **Fig. 5**. This graph is defined by

```
Connections = {(4, 5), (5, 6), (6, 7), (7, 8),
(8, 9), (9, 10), (10, 11), (11, 12), (13, 14),
(14, 15), (15, 7), (14, 16), (16, 17), (17, 18),
(18, 19), (19, 20), (20, 21), (21, 22)}
```

A track segment is described by `Track`, connectivity of two segments by `Connection` and the entire network, in **Fig. 5**, by `Connections` in VDM-SL.

```
1. Connections = set of Connection
.1 inv cons == (forall mk_(s1,s2) in set cons &
```

```
   mk_(s2,s1) not in set cons) and
.2 card { s | mk_(s,-) in set cons} <= 2 and
   card { s | mk_(-,s) in set cons} <= 2;

2. Connection = Track * Track
.1 inv mk_(s1,s2) == s1 <> s2;

3. Track = token;
```

(1) Connections Invariants

1.1) The Connections relation is asymmetric. 1.2) A segment is connected with at most two segments. 2.1) A segment is not connected to itself.

4.1.2 Switch

A switch is consisting of root, left branch and right branch as in **Fig. 5**. For a switch, either it is possible to derive from root to any branch or from any branch to root. It is not possible to drive from one branch to the other. A switch is specified by Switch with three fields switch an identifier of switch, root, and control. The control is a mapping describing control of a switch, either in left or right. For example, the formulas for switch 7, in **Fig. 5**, in the model topology are

```
switch = 7, root = (7, 8) and
control = {(15, 7) |-> <RIGHT>,(6, 7) |-> <LEFT>}

4. Switch :: switch : Track
    root : Connection
    control : map Connection to SwControl
.1 inv sw == (sw.control <> {|->}) and
.2 (forall mk_(s1,s2) in set dom sw.control &
  ((sw.control(mk_(s1,s2)) = <LEFT> =>
  exists mk_(s3,s4) in set dom sw.control &
  s3 = s1 and sw.control(mk_(s3,s4)) = <RIGHT>) and
  (sw.control(mk_(s1,s2)) = <RIGHT> =>
  exists mk_(s3,s4) in set dom sw.control &
  s3 = s1 and sw.control(mk_(s3,s4)) = <LEFT>)) or
  ((sw.control(mk_(s1,s2)) = <LEFT> =>
  exists mk_(s3,s4) in set dom sw.control &
  s4 = s2 and sw.control(mk_(s3,s4)) = <RIGHT>) and
  (sw.control(mk_(s1,s2)) = <RIGHT> =>
  exists mk_(s3,s4) in set dom sw.control &
  s4 = s2 and sw.control(mk_(s3,s4)) = <LEFT>)))and
.3 (forall mk_(s1,s2) in set dom sw.control &
  (sw.switch = s1 or sw.switch = s2) and
  (sw.switch=sw.root.#1 or sw.switch = sw.root.#2));

5. SwControl = <LEFT> | <RIGHT>;
```

(1) Invariants over Switch

4.1) The control mapping is non-empty, i.e., switch has a control. 4.2) A switch has two branches with one vertex in common. The control mapping acts on both branches, i.e., switch control has only two states. Both branches of the switch are either outwards or towards from the switch. 4.3) The ordered pairs defining root and both branches of switch are related to the switch identifier.

4.1.3 Railway Crossing

Crossing is composed of two crossovers as in **Fig. 5**. A train can not switch from one crossover to the other, i.e., for a train, having entered a crossing it is only possible to exit in one direction. Crossing is specified by Crossing with three components cross an identifier of crossing, cross1 and cross2 represent the crossovers . There is only one crossing with identifier (10, 20), in **Fig. 5**. The formulas for the crossing in the model topology are

```
cross = (10, 20), cross1 = {(9, 10),(10, 11)} and
cross2 = {(19, 20), (20, 21)}

6. Crossing :: cross : Track * Track
    cross1 : set of Connection
    cross2 : set of Connection
   inv xng ==
.1 let mk_(s1,s2) = xng.cross in s1 <> s2 and
.2 (forall mk_(s3,s4) in set xng.cross1 &
  (s1 = s3 or s1 = s4) and s2 <> s3 and s2 <> s4 and
  forall mk_(s5,s6) in set xng.cross2 &
  (s2 = s5 or s2 = s6) and s1 <> s5 and s1 <> s6) or
  (forall mk_(s3,s4) in set xng.cross2 &
  (s1 = s3 or s1 = s4) and s2 <> s3 and s2 <> s4 and
  forall mk_(s5,s6) in set xng.cross1 &
  (s2 = s5 or s2 = s6) and s1 <> s5 and s1 <> s6)and
.3 let crs = xng.cross1 union xng.cross2 in
  card {s | mk_(s,s') in set crs & s'= s1 } = 1 and
  card {s | mk_(s',s) in set crs & s'= s1 } = 1 and
  card {s | mk_(s,s') in set crs & s'= s2 } = 1 and
  card {s | mk_(s',s) in set crs & s'= s2 } = 1;
```

(1) Invariants over Railway Crossing

6.1) The track segments of crossing identifier are different. 6.2) Crossing identifier is an ordered pair of two track segments, one is related with cross1 and the other related with cross2. The track segment related with any one of the crossover has no relation with the other. 6.3) A track segment of crossing identifier is connected to and from a track segment. It proves that the segments of crossing identifier are not at the end of topology.

4.1.4 Topology

The network topology is composed of three components connections, switches and crossings. Well defined-ness properties showing relationships between the components of the topology are formalized as invariants in our model, which are not presented here because of the lack of space.

```
7. Topology :: connections : Connections
    switches : set of Switch
    crossings : set of Crossing
   inv top == ...
```

4.2 Network State

Network state is described briefly to be used in the safety analysis. The TrackState is composed of three components. The first one describes state of the component. Second one represents identifiers of trains occupying it. The last one describes control of the component (if switch) either in left or right. It is supposed that if the track segment identifying switch is occupied then the track segment connected

to the identifier is also occupied.

```
8. NetState = map Track to TrackState;

9. TrackState:: trackState : State
    occupiedBy : [set of TrainId]
    swControl : [SwControl];

10. State = <OCCUPIED> | <CLEAR>;
11. TrainId = token;
```

4.3 Trains

Trains is a mapping from TrainId to MBlock (moving block). From **Fig. 5**, an example of moving block of a train is given: {(13, 14), (14, 15), (15, 7), (7, 8), (8, 9)}.

```
12. Trains = map TrainId to MBlock;

13. MBlock = set of Connection
 .1 inv mb == mb <> {} and
 .2 forall mk_(s1,s2) in set mb &
    exists mk_(s3,s4) in set mb & s1 = s4 or s2 = s3;
```

4.3.1 Invariants over Train

13.1) Moving block is non-empty because train occupies some track segments, even at rest. 13.2) Elements in moving block are related.

4.4 Controls

Controls are computer based systems observing trains and network state. Controls is a mapping from control identifier to Control. The Control is composed of tracks, states and trains.

```
14. Controls = map ControlId to Control;
15. ControlId = token;

16. Control :: tracks : set of Track
     states : map Track to State
     trains : Trains
    inv cont == ...
```

4.5 Railway Interlocking System (RIS)

Finally, we formalize RIS which is composed of Topology, NetState, Trains and Controls. We have defined the invariants over the system which are not presented here because our main objective is to give the safety analysis of the system.

```
17. RISystem :: topology : Topology
     netState : NetState
     trains : Trains
     controls : Controls
    inv ris == ...
```

5. Safety Requirements Analysis

The abstract safety requirements, no collision and no derailing, are given with the formal definitions. The safety requirements are refined by introducing the notion of moving block of a train.

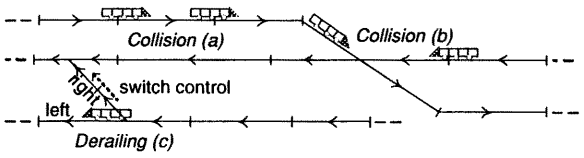

Fig.6 Abstract Definitions of Collisions and Derailing.

5.1 Abstract Safety Properties

The definitions of collisions and derailing, in the model, are illustrated in **Fig. 6**. It is assumed that there will be a collision if there are two trains at a track segment or crossing as in **Fig. 6** (a) and (b), respectively. If train does not respect the state of a switch control, it will cause derailing as in **Fig. 6** (c). The abstract safety properties can be stated as

- Two trains can not reside on the same segment.
- There must be one train at one crossing.
- A train must respect the state of a switch.

The safety function IsSafe is composed of NoCollision and NoDerailing functions. The function NoCollision formalizes the first two properties preventing collision and NoDerailing specifies the last one property preventing derailing.

```
IsSafe : Topology * NetState -> bool
IsSafe(top, ns) ==
 NoCollision(top, ns) and NoDerailing(top, ns);
```

The function NoCollision is formalized assuming that there must be one train at one component (track segment or crossing).

```
NoCollision : Topology * NetState -> bool
NoCollision(top, ns) ==
 OneTrain1Track(top, ns) and OneTrain1Xng(top, ns);
```

5.1.1 Formal Definition of Property 1

In this property, it is stated that the number of trains on a segment must not be greater than one.

```
OneTrain1Track : Topology * NetState -> bool
OneTrain1Track(top, ns) ==
 forall mk_(s1, s2) in set top.connections &
 card (ns(s1).occupiedBy) <= 1 and
 card (ns(s2).occupiedBy) <= 1 ;
```

5.1.2 Formal Definition of Property 2

In the function below, it is stated that the sum of the number of trains moving on the track segments of a crossing should not be greater than one.

```
OneTrain1Xng : Topology * NetState -> bool
OneTrain1Xng(top, ns) ==
 forall xng in set top.crossings &
 (card ns(xng.cross.#1).occupiedBy +
 card ns(xng.cross.#2).occupiedBy) <= 1;
```

5.1.3 Formal Definition of Property 3

In below, it is stated that if any branch of a switch is occupied then control of the switch must be in that branch.

```
NoDerailing : Topology * NetState -> bool
NoDerailing(top, ns) ==
```

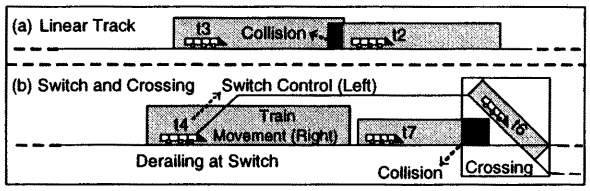

Fig.7 Definitions of Collisions and Derailing.

```
forall sw in set top.switches &
forall mk_(s1,s2) in set dom sw.control &
(ns(s1).trackState = <OCCUPIED> and
ns(s2).trackState = <OCCUPIED>) =>
sw.control(mk_(s1,s2)) = ns(sw.switch).swControl;
```

5.2 Refinement of Safety Properties

Since we are modeling RIS for moving block interlocking that is why the above safety properties are refined by applying it in the notion of moving block of a train as in **Fig. 7**. The stepwise refinement of the properties is restated as

- Intersection of moving blocks of two different trains must be empty preventing collision.
- Moving blocks of two different trains can not contain the same crossing.
- Train's direction (can be deduced from moving block) and switch control must be consistent.

In the refinement of properties, it is also stated that the moving block of a train must be consistent with directed topology. The safety function `IsSafeR` is refinement of the function `IsSafe`.

```
IsSafeR : RISystem -> bool
IsSafeR(ris) == forall cn in set rng ris.controls &
 NoCollisionR(cn, ris.topology, ris.netState) and
 forall sw in set ris.topology.switches &
 NoDerailR(cn.trains, sw, ris.netState);

NoCollisionR : Control * Topology * NetState -> bool
NoCollisionR(cont, top, ns) ==
 OneTrain1TrackR(cont, top) and
 forall xng in set top.crossings &
 OneTrain1XngR(cont, xng, ns);
```

5.2.1 Refinement of Property 1

The function `OneTrain1TrackR` is composed of two auxiliary functions. The first `InterOfMBEmpt` states that the intersection of moving blocks of two different trains is always empty. In the second function, `TrackDirOk`, it is stated that the moving block of a train and the topology must be consistent.

```
OneTrain1TrackR : Control * Topology -> bool
OneTrain1TrackR(cont, top) ==
 InterOfMBEmpt(cont.trains) and
 TrackDirOk(cont.trains, top);

InterOfMBEmpt : Trains -> bool
InterOfMBEmpt(trains) ==
 forall t1, t2 in set dom trains &
 t1 <> t2 => forall mk_(s1,s2) in set trains(t1) &
 forall mk_(s3,s4) in set trains(t2) &
 {s1,s2} inter {s3,s4} = {};

TrackDirOk : Trains * Topology -> bool
TrackDirOk(trains, top) ==
 forall t in set dom trains &
 forall mk_(s1,s2) in set trains(t) &
 mk_(s1,s2) in set top.connections ;
```

5.2.2 Refinement of Property 2

In the function `OneTrain1XngR`, we state that if a crossing is in the occupied state then it can be contained in the moving block of only one train.

```
OneTrain1XngR: Control * Crossing * NetState -> bool
OneTrain1XngR(ct, xng, ns) ==
 forall t1, t2 in set dom ct.trains & t1 <> t2 =>
 ((ns(xng.cross.#1).trackState = <OCCUPIED> and
 ns(xng.cross.#2).trackState = <OCCUPIED>) =>
 OneTrainAtXng(xng, ct.trains(t1), ct.trains(t2)));
```

In function `OneTrainAtXng`, it is stated that there must be only one train at one crossing. The function `TrainAt1X` describes that the train, which is occupying the crossing, can occupy only one crossover of that crossing.

```
OneTrainAtXng : Crossing * MBlock * MBlock -> bool
OneTrainAtXng(xng, mb1, mb2) ==
 let xngA = xng.cross1 union xng.cross2 in
 (xngA inter mb1 <> {} => (xngA inter mb2 = {} and
 TrainAt1X(xng, mb1))) and
 (xngA inter mb2 <> {} => (xngA inter mb1 = {} and
 TrainAt1X(xng, mb2))) ;

TrainAt1X : Crossing * MBlock -> bool
TrainAt1X(xng, mb) ==
 (xng.cross1 inter mb <> {} =>
 xng.cross2 inter mb = {}) and
 (xng.cross2 inter mb <> {} =>
 xng.cross1 inter mb = {});
```

5.2.3 Refinement of Property 3

In the function `NoDerailR`, it is stated that if the track segment of a switch is in the occupied state then the switch can be included in the moving block of only one train. And, no other train can occupy the switch. Further, the switch control and train direction must be consistent.

```
NoDerailR : Trains * Switch * NetState -> bool
NoDerailR(trains, sw, ns) ==
 forall t1, t2 in set dom trains & t1 <> t2 =>
 (ns(sw.switch).trackState = <OCCUPIED> =>
 OneTran1Sw(sw, trains(t1), trains(t2), ns));
```

In the function `OneTran1Sw`, it is stated that the switch can be contained in the moving block of only one train. Further, the switch control must be left appropriate for the train occupying the switch.

```
OneTran1Sw:Switch * MBlock * MBlock * NetState->bool
OneTran1Sw(sw, mb1, mb2, ns) ==
 let swA = {sw.root} union dom sw.control in
 (swA inter mb1 <> {} =>
 (swA inter mb2 = {} and SwControlOK(mb1,sw,ns)))and
 (swA inter mb2 <> {} =>
 (swA inter mb1 = {} and SwControlOK(mb2,sw,ns))) ;
```

In the function below, it is stated that if any switch is in the moving block of a train then the control of switch and state of the control in that

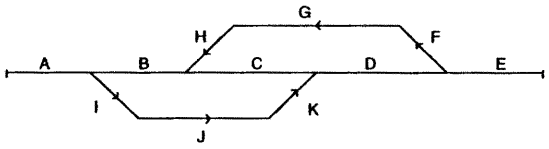

Fig.8 Dynamic Topology.

switch must be consistent preventing derailing.

```
SwControlOK : MBlock * Switch * NetState -> bool
SwControlOK(mb, sw, ns) ==
 forall mk_(s1,s2) in set dom sw.control &
 mk_(s1,s2) in set mb =>
 sw.control(mk_(s1,s2)) = ns(sw.switch).swControl;
```

6. Conclusion

This paper is one step forward to develop an abstract model for dynamic topology, i.e., a topology in which at some parts one way while at other parts both way movement of train is possible and direction of any track segment in topology can also be changed, if required. For example in **Fig. 8**, on the track ABCDE both way movement is possible but at tracks EFGHB and AIJKD only one way movement of train is allowed. In the paper, moving block Railway Interlocking System (RIS) is formalized for directed topology and the consistency is observed in the model. We developed several models and this model is the result after the series of refinements. The applicability of the VDM-SL toolbox [11] has been evaluated for the formal analysis.

We are aware that giving a complete definition of safe RIS is not easy because of the complex interaction of its components. Although we have treated a part of the problem but our contribution is significant because the moving block RIS is an emerging technology and there does not exist much work in this area. Further, the moving block and the network topology can not be separated in real railways and, hence, we have treated the problem by integrating both the components. It is observed that directed topology has increased its complexity.

Although our work does not represent a real world problem but it is useful for researchers interested in formal methods and their applications, analyzing safety critical systems, etc. We believe that our model can also be useful for such systems having domain knowledge because of abstraction.

Our specification is based on the model in graph theory by which we have achieved the objective of integrating formal and informal approaches. But formalizing graph theory is not easy, since there has been little tradition of formalization in it due to the concreteness of the graphs [3]. We also observed that the use of graph theory with VDM-SL increased the power of modeling in this research.

In future, this work for directed topology and our previous work [9),10)] for bi-directional topology will be analyzed to refine the model for the dynamic topology. Initially, we have taken some assumptions to make the model simple and as a result the model has some limitations which will be relaxed. We know that RIS is a distributed real time system, such aspects will also be analyzed.

The system is formalized using VDM-SL because it is a formal language used both at abstraction and detailed level. The use of VDM-SL tool-box has eased the model development, as we were able to check the specification and thereby could observe the consequences of our definitions.

Acknowledgments

Zafar would like to give his sincere thanks to Dr. Yoshinari Nomura, who helped for preparing this paper.

References

1) A.E. Haxthausen and J. Peleska, "Formal Development and Verification of a Distributed Railway Control System," In Proc. 1st FMERail Seminar, June 1998.
2) A. Simpson, "Towards the Mechanical Verification of Moving Block Signaling Systems," Technical Report CMS-TR-99-06, School of Computing and Mathematical Sciences, Oxford Brookes University, UK, 1999.
3) C.T. Chou, "A Formal Theory of Undirected Graphs in Higher-Order Logic (HOL)," In Proc. 7th International Workshop on HOL, Theorem Proving and Its Application, LNCS 859, Springer-Verlag, 1994, pp. 144-157.
4) J.F. Groote, J.W.C. Koorn, and S.F.M. van Vlijmen, "The Safety Guaranteeing System at Station Hoorn-Kersenboogerd," In Proc. 10th IEEE Conference on Computer Assurance, (COMPASS'95), Gaitherburg, 1995.
5) J. Hoenicke, "Specification of Radio Based Railway Crossing with the Combination of CSP, OZ and DC," http://www.i-u.de/fbt2001/fbt2001_docs /hoenicke.ps, June 2001.
6) J.P. Bowen, M.G. Hinchey, "Seven More Myths of Formal Methods," IEEE Software, July 1995, pp. 34-41.
7) K.M. Hansen, "Formalizing Railway Interlocking Systems," In FME Rail Workshop 2, Denmark, 1998.
8) M.J. Morley, "Safety in Railway Signaling Data: A Behavioral Analysis," In Proc. 6th Annual Workshop on Higher Order Logic, Theorem Proving and Its Application, LNCS 780, Springer-Verlag, 1993.
9) N.A. Zafar and K. Araki, "Safety Analysis in Route Allocation for Moving Block Railway Interlocking System," Proc. International Workshop on Informations & Electrical Engineering, Korea, 2002, pp.190-195.
10) N.A. Zafar and K. Araki, "Modeling and Safety Analysis of Moving Block Railway Interlocking System," Proc. International Symposium on Future Software Technology, China, 2002.
11) The VDM-SL Tool Group, "Users Manual for the IFAD VDM-SL Tools," Technical Report IFAD-VDM-4, The Institute of Applied Computer Science, 1994.
12) The VDM-SL Tool Group, "The IFAD VDM-SL Language," Technical Report IFAD-VDM-1, The Institute of Applied Computer Science, 1994.