A Tool for Automatic Formal Modeling of Railway Interlocking Systems

Muhammed Ali Nur Oz, Ibrahim Sener, Ozgur Turay Kaymakci, Ilker Ustoglu, Galip Cansever
Control and Automation Engineering
Yildiz Technical University
Istanbul / Turkey
Email: {maoz, isener, kaymakci, ustoglu, cansever}@yildiz.edu.tr

***Abstract*— This paper introduces a new software tool, which can be used for automatic generation of Timed Arc Petri Net (TAPN) models from the railway station topology for interlocking systems. The introduced software tool has two components, 'Graphical User Interface' to draw the station topology and 'Application Software' to generate TAPN models from the station topology. TAPN is a highly recommended formal modeling method by the CENELEC EN50128 standard. Generated models, belonging to the station, are stored in an XML file and can be viewed using TAPAAL.**

Keywords— Software tool; Automatic model generation; Interlocking system; Railway; Timed arc Petri net.

I. INTRODUCTION

A railway interlocking system is a kind of control system, which is developed to control railway traffic securely. The basic objective of interlocking system is to prevent trains from colliding and derailing by controlling field equipments such as points, signals and track circuits in a railway stations. Due to the fact that interlocking systems play a major role in ensuring the safety and reliability of the railway transportation, these systems are usually realized based on the CENELEC (Comite Europeen de Normalisation Electrotechnique) EN5012x family of railway standards including EN50126, EN50128 and EN50129 standards, which concentrate on the modeling methods necessary for ensuring safety and reliability in railway transportation systems. These standards apply to both heavy rail systems and light rail systems [1]. Automatic modeling and designing of it is very important to make railway transportation safer and more reliable. However, the modeling and designing of interlocking systems is currently manual process, which is inefficient and error-prone due to the complexity of the railway yard. On the contrary, automatic modeling and designing of interlocking system reduces the human errors and improves the efficiency in the generation of the models, which describe the system. Automatic generated models may be more reliable than manual models. So the reliability and safety of the whole system increases. For this reason, the automatic generation of models, which belong to interlocking system, based on the station topology is very significant. Automatic generation of railway interlocking models are performed by using developed software tools. There exist a limited number of studies in literature regarding the development of software tools, which can be used for railway interlocking systems. Interlocking tables were generated from the station topology automatically in some studies [2,3]. A tool, which can be used to automatically generate and verify the interlocking table of railway station designed by DSL-CBI (Domain Specific Language for Computer Based Interlocking), is developed in [4]. Another paper [5] describes a tool, which is formally developed using the RAISE method, for formal modeling relay interlocking systems. In [6], an automatic transformation tool from UML (Unified Modeling Language) model to Petri net by using the semantic transformation rules is presented for distributed railway interlocking system. With the proposed tool, generated Petri net model is verified. A developed software is used to generate Statecharts model automatically from station layout in [7]. In previous work [8] of us, the TAPN models in [9] were generated automatically from the station topology through a software tool, which we developed for point automation system.

This study, which is a further research of our previous work mentioned in [8] and [9], where double track universal point automation system have been studied, focuses on developing a software tool for automatic generation of TAPN models for interlocking system with single track. It was not possible to form models that were powerful enough to reflect the interlocking system in previous studies since temporal movements in the system were not transferred into the model due to the fact that Timed Arc Petri Nets [10, 11] had not been used. This special software tool, which is developed by using C# programming language, generates TAPN models unlike the other formal models. So, more powerful models were formed since TAPN was used in modeling. Use of the developed tool is not limited to any station topology. It will also be able to use for modeling of different station topologies. Automatically generated models are stored in XML file. These models can be viewed using TAPAAL [12], which is a tool for modeling, simulation and verification of Timed-Arc Petri nets.

II. RAILWAY INTERLOCKING SYSTEM COMPONENTS

Railway system consists of some fundamental components such as points, signals and track circuits.To understand any railway topology better, components mentioned above are explained first: A railway point is a mechanical tool which allows trains to be guided from one track to another on railway intersection according to the desired route. There are two

978-1-4799-8569-2/15/$31.00 ©2015 IEEE

positions named as normal and diverging. Railway signals are a system used to control railway traffic safely so that collision of the trains can be prevented. Signals transmit colored light (green, red, yellow) notice, which notifies the trains regarding the proceeding of the trains and feed up until the next signal. There are three types of signal; 3-aspect long signals, 4-aspect long signals and 3-aspect ground signals.

TABLE I. SIGNAL INDICATORS AND THEIR MEANINGS

Indicator	***Meaning***
Red	Stop at signal, proceeding track is busy.
Yellow	Slow down at signal, proceeding track is empty but the next track is busy
Green	Proceeding track and the next track is empty
Green over Yellow	Slow down at signal, proceeding track and the next track is empty.
Yellow over Yellow	Slow down at signal, proceeding track is empty but the next track is busy.
Red over Yellow	Moving to a Signalless road, might be train ahead move very slowly.

3-aspect long signals include red, green and yellow lights. There is also an additional yellow light in 4-aspect long signals. This extra yellow light indicates whether there is a deviation from the route or not. 3-aspect ground signals are used inside of the stations. it is understood to be a deviation from the color of the notification of 3-aspect ground signals. Track circuit is a simple electrical circuit designed to detect the absence or presence of a railway vehicle in a certain part of a railway. They provide information whether the route is available or occupied by a railway vehicle.

III. SOFTWARE TOOL

In order to automatically generate TAPN models for interlocking systems, a tool was developed using C# programming language, which is recommended by CENELEC EN 50128. This specially developed tool can be examined in two stages. First stage is creating a user friendly graphical user interface that enables the user to easily produce and edit the station topology. Second stage is creating TAPN model from the station topology that was produced in the first stage. Created TAPN models must be easily understandable by experts in order to make improvements or corrections to the model. At last TAPN models are saved for further use as xml files.

A. Graphical User Interface

Using the graphical user interface, the user can easily create and modify a single line diagram of the station without the need for an additional tool. This interface is composed of three main parts; a toolbox, an information panel and a layout editor. The toolbox includes blocking blocks of railway systems, which represent station topology components such as tracks, un-signaled tracks, 3-connection points and 4-connection points. The Information panel is used to notify the user on how to use the interface and to warn the user of unauthorized actions. The layout editor is the third part of the interface, where the single line diagram is graphically created. It is made up of grids to make modeling easier. A screenshot of the interface with an example station model can be seen at Fig. 1.

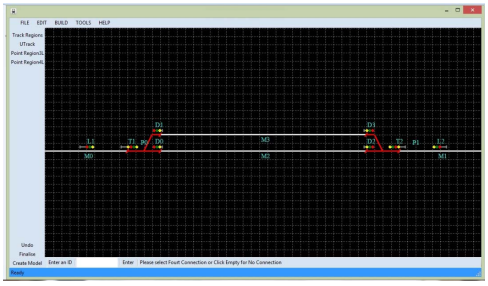

Fig. 1. Screenshot of the interface

The single line diagram is modeled in two parts using the proposed tool. When a new project is created the main tracks that lead in an out of the station are initially created. And the station is modeled between these two main tracks. In the first part, the user models the tracks inside the station by first choosing the track component from the toolbox panel and naming it using the information panel. Secondly the tracks are drawn on the interface by specifying the start and end points of the tracks on the layout panel. In the second part same steps are applied but this time to points that combine the tracks modeled in the first part. There are two point icons in the toolbox panel to be able to model points that connect three or four tracks. Finally the user clicks the finalize button and the signals are created automatically.

B. Application Software

The drawn single line diagram is converted into three matrices named Track, Signal and Point matrices. These matrices hold the data of the components such as their amounts, tags and connections. Using these three matrices, two matrices called routes and overlapping routes, which are essential for producing TAPN models, is created as seen in algorithm 1.

```
1:  Connection=null
2:  Routes[]=null
3:  for (i € Track[i])
4:      for (j € Track[j])
5:          Connection=findnextconnection(Track[i]))
6:          if connection is equal to a point then
7:              if connection is a Normal point then
8:                  add connection with its position to Routes[]
9:              else If connection is a Reverse point then
10:                 add connection with its position to Routes[]
11:             end if
12:             else if connection is equal to a track
13:             if connection is equal to Track[j] then
14:                 add target track to Routes[]
15:                 Break
16:                 else
17:                 delete last added route on routes[]
18:                 Break
19:             end if
20:             else if connection is null
21:                 Break
22:         end if
23:     end for
24: end for
```

Algorithm 1. Algorithm to find routs

Since routes are opened from one signal to the next adjacent signal only, one signal and one point exist inside a route matrix. Overlapping routes must be found and route model should forbid opening to overlapping routes instantaneously. Overlapping routes are found using the algorithm as seen in Algorithm 2.

```
1:  overlappingRoutes[,]=null;
2:  for (i € Routes[i,])
3:      for (k € Routes[k,])
4:          if any point region on routes[i] equal to a point
              region on routes[k]
5:          Add route to overlappingRoutes[i,]
6:          else if if the start of routes[i] is equal to the
              start of routes [k]
7:          Add route to overlappingRoutes[i,]
8:          else if the destination of routes[i] is equal to
              the destination routes[k]
9:          Add route to overlappingRoutes[i,]
10:     end if
11:   end for
12: end for
```

Algorithm 2. Algorithm that finds overlapping routes

The TAPN model is generated in parts to reduce complexity and to increase readability. These parts are named signals, points, routes and field models. Point TAPN model is a very simple model, which sets the appropriate point to normal reverse position according to the states that are shared with routes TAPN model. Signal model is also a simple model, which sets signals to appropriate states using the states that are shared with routes and field models. Using the routes matrix, which has been created earlier our developed software, will create a route model for every route possible. These Route TAPN models open routes by setting appropriate signals and points to desired states. The purpose of this TAPN model is to make sure no overlapping routes can be set at the same time. Another purpose is to translate points to their desired position. Points have been set and finally route is set and locked. Algorithm used for this purpose is given in Algorithm 3.

```
1:  for(i € Routes[i,j])
2:      Place initial positions and connections to the model
3:      for(k € OverlappingRoutes[])
4:          Place overlapping positions as inhibitors to the
            model
5:      end for
6:      for(j € Routes[i,j])
7:          if Routes[i,j] is null then
8:              Break
9:          else
10:         if Routes[i,j] is a normal positioned point then
11:             Place point as normal to the model
12:         else
13:             Place point as reverse to the model
14:         end if
15:     end if
16:   end for
17: end for
```

Algorithm 3. Route TAPN model creating algorithm

Field model governs which paths a train can take, using the signal indicators to navigate the train with the help of track circuits that are used to locate the train. The algorithm that generates the field TAPN model is shown in Algorithm 4.

```
1:  Firstroute=true
2:  for(i € Track[i])
3:      Firstroute=true
4:      for(k € Track[k])
5:          for(j € Routes[j,])
6:              if start of routes[j,] is equal
7:              to Track[i] and destination of routes[j,]
                is equal to Track[j] then
8:              if firstroute equals true then
9:                  Initialize field model with the data from
                    routes[j,] Firstroute = false
10:             end if
11:             if signal of route[j,] is 4- aspect then
12:                 Place the signal from routes[j,]
14:             else
15:                 place the signal from routes[j,]
16:             end if
17:             for(l € Routes[,l])
18:                 if routes[j,l] is not equal to null then
                      Place point to model
19:                 end if
20:             end for
21:         end if
22:     end for
23:   end for
24: end for
```

Algorithm 4. Field TAPN model creating algorithm

IV. CASE STUDY

The station given in Fig. 1 is chosen as a model, this station has two points, eight signals and four track circuits. Sets to represent the following items at this station, whose topology is drawn at Fig. 1, were defined: two points P={P0, P1}, four track circuits TC={M0, M1, M2, M3} as well as eight signals S={ L1, L2, T1, T2, D0, D1, D2, D3}. According to the station topology, route and track circuit TAPN models were generated automatically by using the developed tool. The routes identified can be opened for the trains if and only if the relevant track circuits are not occupied by another train and the train proceeding on the second route to be opened should not be facing the train proceeding on the first route. Based on this, separate Timed- Arc Petri Net models were formed for each route. As an example, the route model, which will be opened for the trains that move between M0 and M2, is given in Fig. 2. Similarly, all other route TAPN models were also formed.

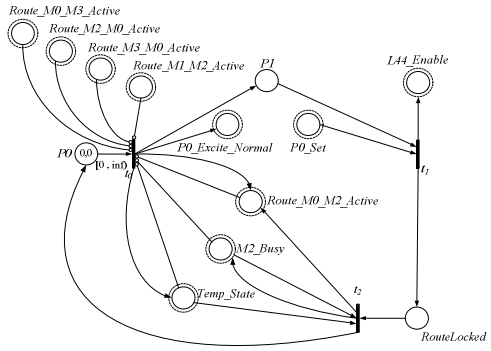

Fig. 2. Route TAPN model

The automatic generated track circuit Timed-Arc Petri Net model, which indicates the movement of the trains between M0 and M2, can be seen in Fig. 3. Similarly, all other track circuits TAPN models were also formed.

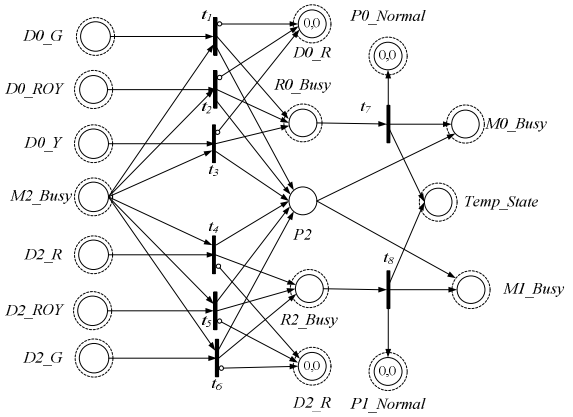

Fig. 3. Route TAPN model

CONCLUSION

A software tool was developed successfully by using C# programming language, which is recommended by CENELEC EN 50128, to generate automatically the interlocking system's TAPN models from the station topology. The developed tool was tried for different stations, operated by TCDD (Turkish Railway Company) in Turkey, and successful models were obtained. It was concluded that the tool significantly reduced the modeling faults caused by human factor and improve the efficiency in the generation of the models. Moreover, reliability and safety of the whole system increases thanks to automatic generation of the system models.

REFERENCES

[1] CENELEC EN 50128. Railway applications - communication, signaling and processing systems - Software for Railway Control and Protection Systems, 2011.

[2] A. Kuzu, O. Songuler, A. Sonat, S. Turk, B. Birol, E. H. Dogruguven, "Automatic interlocking table generation from railway topology," IEEE International Conference on Mechatronics (ICM), 2011, pp. 64-70.

[3] U. Yildirim, M. S. Durmus, M. T. Soylemez, "Automatic interlocking table generation for railway stations using symbolic algebra," 13th IFAC Symposium on Control in Transportation Systems, Vol. 13, Part 1, pp. 171-176, 2012.

[4] Y. Cao, T. Xu, T. Tang, H. Wang, L. Zhao, "Automatic generation and verification of interlocking tables based on domain specific language for computer based interlocking systems (DSL-CBI)," IEEE International Conference on Computer Science and Automation Engineering (CSAE), Vol. 2, pp. 511-515, 2011.

[5] A. E. Haxthausen, A. A. Kjær, M. L. Bliguet, "Formal development of a tool for automated modelling and verification of relay interlocking systems," Lecture Notes in Computer Science, Vol. 6664, pp 118-132, 2011.

[6] X. Hei, L. Chang, W. Ma, J. Gao, G. Xie, "Automatic transformation from UML Statechart to Petri Nets for safety analysis and verification," IEEE International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2011, pp. 948-951.

[7] C. Xiangxian, H. Hai, H. Yulin, "Automatic generation of relay logic for interlocking system based on Statecharts," IEEE Second World Congress on Software Engineering (WCSE), Vol. 2, pp. 183-188, 2010.

[8] M. A. N. Oz, I. Sener, O. T. Kaymakci, I. Ustoglu, G. Cansever, "Topology based automatic formal model generation for point automation systems," Information Technology And Control, Vol. 44, No. 1, pp. 98-111, 2015.

[9] I. Sener, O. T. Kaymakci, I. Ustoglu, G. Cansever, "Specification and formal verification of safety properties in point automation system," Turkish Journal of Electrical Engineering & Computer Sciences, in press.

[10] L. Jacobsen, M. Jacobsen, M. H. Moller, J. Srba, "Verification of timed-arc Petri nets," Lecture Notes in Computer Science, Vol. 6543, pp. 46-72, 2011.

[11] M. Andersen, H. G. Larsen, J. Srba, M. G. Sørensen, J. H. Taankvist, "Verification of liveness properties on closed timed-arc Petri nets," Lecture Notes in Computer Science, Vol. 7721, pp. 69-81, 2013.

[12] A. David, L. Jacobsen, M. Jacobsen, K. Y. Jørgensen, M. H. Møller, J. Srba, "TAPAAL 2.0: integrated development environment for timed-arc Petri nets," Lecture Notes in Computer Science, Vol. 7214, pp. 492-497, 2012.