Automatic generation of relay logic for interlocking system based on Statecharts

Chen Xiangxian, Huang hai
Department of Instrumental Engineering
Zhejiang University
Hangzhou, China

He Yulin
System Development Center
Zhejiang Insigma Rail Transportation Engineering Co. LTD
Hangzhou, China

***Abstract*—Computer-based Interlocking System (CIS) is a kind of widely applied safety-critical system in current signaling industry. CIS always uses relay logic to describe interlocking safety logic. Normally, all logic is designed by experienced signaling engineer manually, this develop mode is low-efficient and expensive. This paper presents a new logic design model. In this new model, a component-based model is used to describe topology of station layout, the topology data is analyzed and statecharts are used to describe safety logic of interlocking. Finally the statecharts are transformed to relay logic. The entire logic generating procedure can be fulfilled by software tools automatically, so efficiency is improved greatly. The introduction of statecharts also makes formal verification of safety logic possible, which can guarantee that the generated logic is safe.**

Keywords-interlocking; relay logic; Statecharts; formal verification

I. INTRODUCTION

A Computer-based Interlocking System (CIS) is a safety-critical system widely applied in signaling industry. Interlocking system supervises and controls wayside devices such as track circuits, semaphores and points in railway station, and set up safe routes for passing trains. Interlocking software can be modeled as three layer architecture. The highest layer is dedicated to the interaction with operators, the ultimate user of CIS, or other systems, which send commands to CIS; the lowest layer is the actual devices interface which detects devices states and controls station devices according to logic processing results; the middle layer is the core of the interlocking software which implements "safety logic" of kinds of operations requested by external operators [1].

For historical reasons that CIS originates from relay-based interlocking, relay ladder logic, a low level language representing Boolean-valued assignments, is widely used by signaling engineers to specify and develop safety logic [2]. This kind of interlocking which uses relay ladder logic language to implement safety logic is called PLC-based interlocking. Normally, the basic procedure of developing relay ladder logic program for interlocking is concluded as follows:

- Define and specify system requirements including function, safety and reliability requirements using natural language.
- Logic designers use ladder diagram following standard IEC61131-3 to implement interlocking safety logic according to requirements.
- Simulation and test method are used to verify that the logic is correct and safe.

Several problems reside in this develop mode. First, it needs experienced logic designers who can not only understand interlocking principles but also excel in designing ladder logic. They must know clearly how interlocking systems work and imagine many possible safety issues. When they create control logic they must attempt to consider every possible occasion. Second, the develop procedure is time-consuming and inefficient. Although different interlocking systems share a lot of common safety rules, the entire logic needs to be redesigned whenever station layout changes and no logic module can be reused. Third, simulation and test methods can find many flaws in software but can't guarantee correctness of the ladder logic because it is impossible to test all the possible states of the interlocking systems manually [3].

This paper presents a new software develop model which incorporates Statecharts [4], an extension of the classic formalism of Finite State Machines (FSM), into normal relay logic design process. This system uses statecharts to define and specify interlocking safety logic. It takes a station layout topology data as input, and generates logic specification in form of statecharts automatically according to interlocking rules. The statecharts can be transformed to relay ladder logic with the same functions and safety properties automatically by special tools. Meanwhile application of statecharts makes it possible to implement formal verification on interlocking logic by means of model checker using some available tools [5]; this can highly increase the safety level of interlocking software.

This paper is structured as follows. It starts by introducing architecture and principles of the software develop model. Then a station is taken as an example to discuss the procedure and algorithm of generating statecharts and relay logic. Then a set of software tools that are implemented according to this research is introduced. Next the paper analyzes advantages of this new model. The last

978-0-7695-4303-1/10 $26.00 © 2010 IEEE
DOI 10.1109/WCSE.2010.31

section presents some conclusions to this work and some future works need to be fulfilled.

II. PRINCIPLE

A CIS system ensures safe operations of wayside devices in a railway station. Such a system controls an arrangement of semaphores and track points so interconnected that their actions shall succeed each other in proper sequence and for which interlocking safety logic is defined [6]. The safety logic is based on two inputs: station layout topology data and safety rules. Station layout topology describes relative positions of wayside devices to each other and their connection relationships, and the basic routes information can be generated from it. Safety rules describe a set of general logic that every route and every device should follow. Safety rules are relatively stable, for most countries have their own national interlocking standards. But for systems applied in multinational environment, they should still consider gaps between standards of different countries.

A. component-based model of station layout topology

A station layout is constructed by a group of independent elements interconnected. To describe a station layout, we abstract each element to a component and establish relationships between them. We define three basic kinds of station components: semaphore, switch and track section. Each instantiated component corresponds to a real wayside device [7]. The links between different components connect these components together like a network representing the structure of the station. As shown in Fig.1, the solid line link between two components means that they are adjacent to each other. The dashed line link between a switch and a point section means that the switch position decides where a train will go when it passes the point section.

B. Statecharts

Statechart is an extension of the classic formalism of Finite State Machines (FSM). A statechart is a type of graph consisting of a number of interconnected nodes corresponding to states of the process. The link between two nodes means a transition exists between two states. Transition is driven by trigger event, which can refer to the state of other machines or global variables.

The nodes of the statechart are states of the FSM under study from the set of state S= {s_1, s_2… s_n}. Let Z= {z_1, z_2… z_k}, the set of discrete state-variables of the machine. Each state s_i is a value set of discrete state-variables which are defined in Z. Every machine has an initial state with all variables in Z are in off status. The set A= {a_1, a_2, a_3…} corresponds to the events and conditions cause transitions from one state to another and can be expressed as logic expressions of variables of other parallel machines, global variables or combinations of them.

The prototype model presented here uses special tool to generate statecharts describing interlocking safety logic according to station layout topology and Chinese standard safety rules. The safety logic is organized in the unit of device or route [1, 6]. Each device or route has one or more statecharts itself, while a statechart can only define states and state transitions of one device or route. The state set S and variables set Z can be generated according to device type. The transitions can be generated by analyzing devices location, searching involved routes and applying related safety rules. Because state changes of one statechart can trigger transitions of another statechart, different statecharts are interconnected, and they describe the safety logic of interlocking altogether.

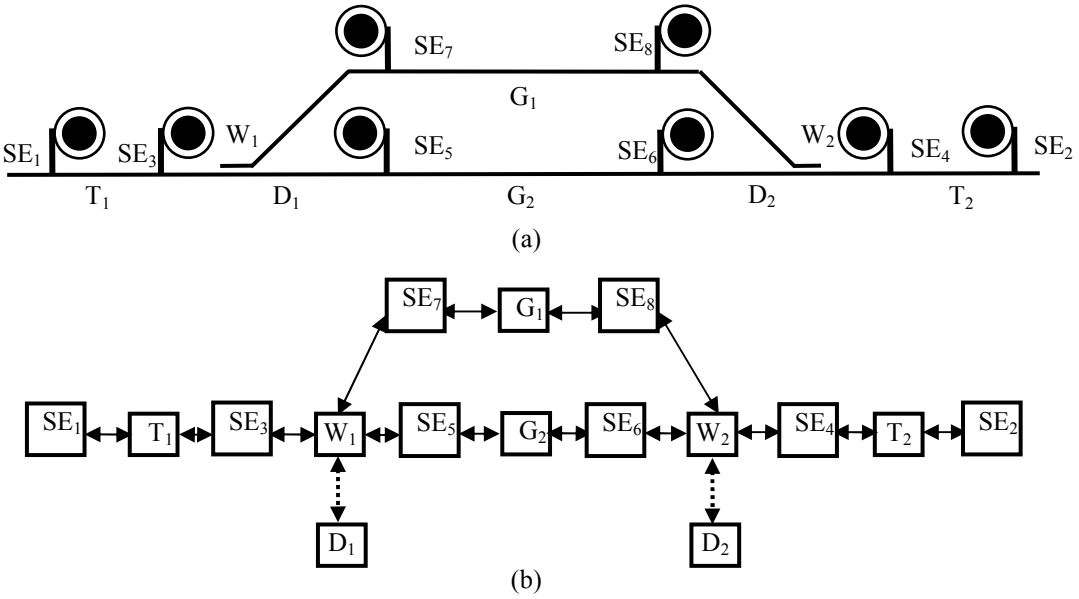

Figure 1. (a) Example of a station layout, (b) component-based model of topology.

C. Relay logic

Relay logic is a sequence of Boolean assignments. To express relay logic more concisely, a list of Boolean expressions is applied.

It is possible to get the Boolean expressions of all variables in set Z from statechart. To get the Boolean expression of a variable N, the tool first needs to search all transitions in the statecharts to determine the logic set "*Off*" of events that cause changes of the variable from 1 to 0. Second the tool needs to determine the logic set "*On*" of events that cause changes of the variable from 0 to 1. Third, the Boolean expression of variable N is as follow [8]:

Assign ~ (*Off*) * (*On* + Present State of variable *N*) to *N*.(1)

In (1), the phrase "assign to" means evaluation to variable *N*. "*" represents conjunction and "+" represents disjunction. "~" means negation. "Present state of variable *N*" plays the role of a bit memory which is updated in each change of the variable *N*.

Finally the Boolean expressions need to be simplified and synthesized. The whole procedure can be simplified as shown in Fig. 2.

III. AN EXAMPLE: SECTION LOCKING

In this section, we will use section lock as an example to show how Boolean expressions are generated.

When a route is set up, interlocking system will lock all track sections of this route in the same direction with the route. Locking is a safety-critical action. When a section is locked, any other routes passing through it are not available any more until it is unlocked. A locked section is unlocked in two conditions: the train enters the route and passed through it or the route is cancelled manually.

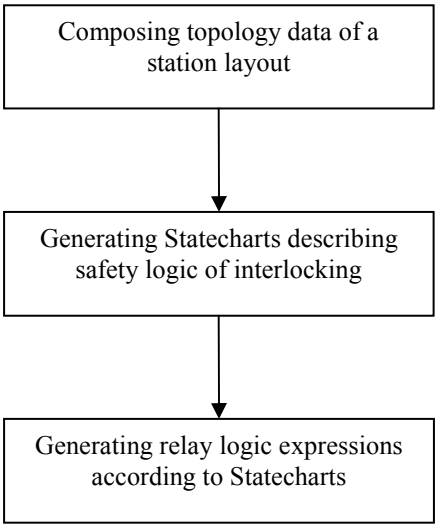

Figure 2. The procedure of generating relay logic can be simplified as three steps: 1) compose topology data, 2) acquire statecharts, 3) generate relay logic .

The statechart describing the logic of locking section G_2 of station in Fig. 1 is shown in Fig. 3. In this statechart, the variable set Z contains two variables, G_2_WS and G_2_ES. G_2_WS represents locking section G_2 in westward direction while G_2_ES represents locking section G_2 in eastward direction. The value 0(false) means that the section is locked while the value 1 (true) means that the section is unlocked. The meanings of other variable are explained in Table 1. S_0 is the initial state of section G_2 with both G_2_WS and G_2_ES set to 0 (false). S_1 means section G_2 is locked by a westward route, S_2 means that section G2 is locked by an eastward route and S_3 means that section G_2 is free.

The transitions between different states can be generated according to topology data shown in Fig. 1 and Chinese safety rules. Take transitions between S_1 and S_3 as an example:

- A section is locked in westward direction when a westward route including the section is set up. During searching routes of station shown in Fig. 1, we can find only one westward route passes section G_2 which starts at semaphore SE_4 and ends at semaphore SE_5. The switch D_2 should be at normal position in this route. Thus section G_2 should be locked in westward direction while this route is available and set up. So we can get a transition from S_1 to S_3 with condition of SE_4_AS==0 and W_2_NWP==1.
- When a train enters the route and passes through it, the sections of the route are unlocked one by one from start semaphore to end semaphore. In this case, a specified section of the route is unlocked automatically when the interlocking decides that the preceding section, the specified section and the next section are occupied in sequence and then unoccupied in sequence. According to this rule, the section G_2 can be unlocked if the interlocking can decide that section D_2, G_2, and D_1 are occupied in sequence and then unoccupied in sequence. The condition is equivalent to section D_2 is unlocked , G_2 is unoccupied and section D_1 is occupied while switch W_2 is at normal position or G_2 is unoccupied and section D1 is occupied while switch W_2 is at reverse position. Thus we can get two transitions from S_1 to S_3. One has the condition of W_2_RWP==1, G_2_P==1 and D_1_P==0; the other has the condition of D_2_WS==1, W_2_NWP==1, G_2_P==1 and D_1_P==0.
- When a route cancel request is received and the route cancel check condition is satisfied (route is complete and no train is approaching), all the sections in the route can be unlocked simultaneously. In this case, the condition is equivalent to a route cancel request on semaphore SE_4 is received, the preceding section D_2 is unlocked and the specified section G_2 is unoccupied. Thus we get the third transition from S_1 to S_3: D_2_WS==1, W_2_NWP==1, G_2_P==1 and S_4_CANPB==1.

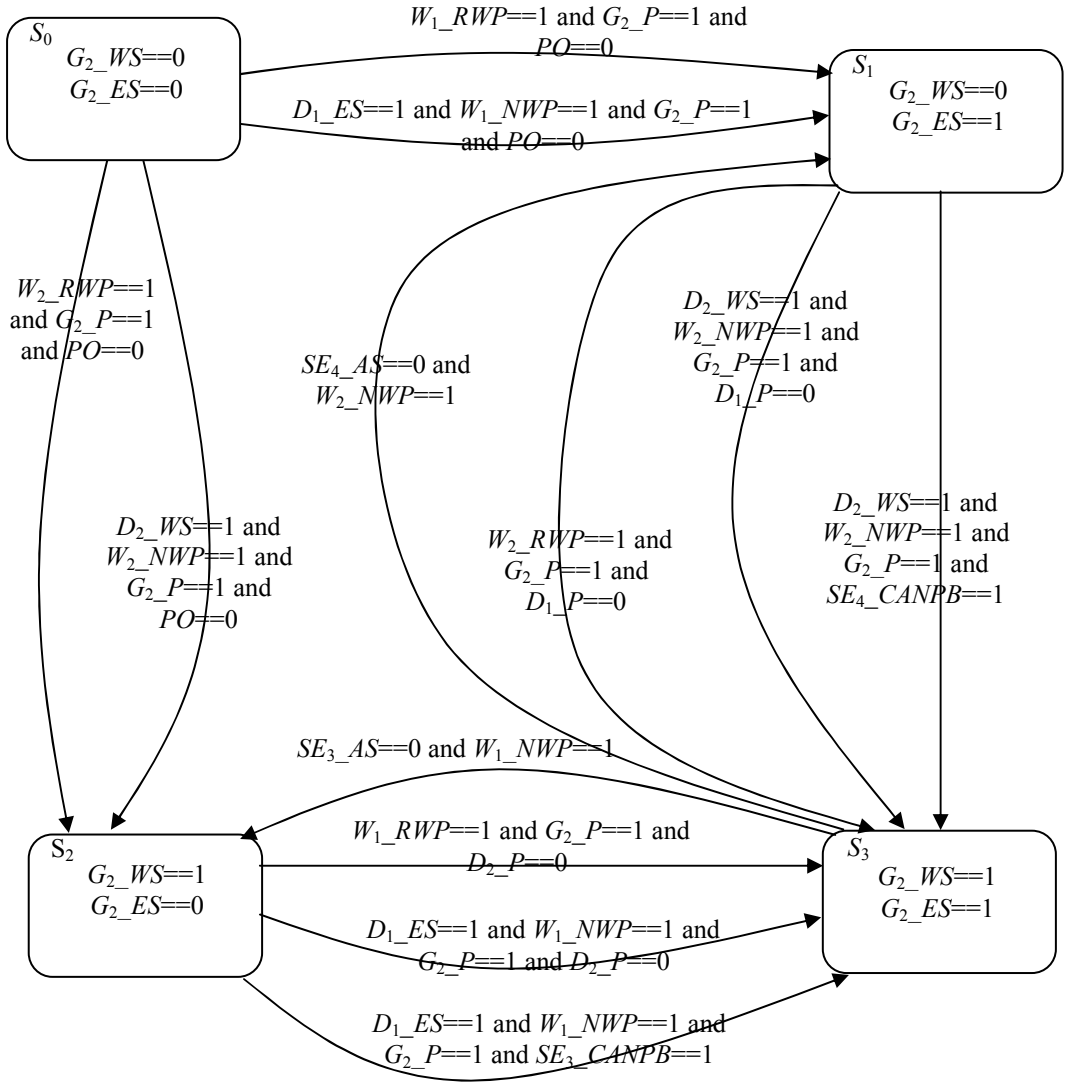

Figure 3. Statecharts is selected to describe safety logic of locking track section G_2 of station shown in Fig. 1 .

TABLE I. EXPLANATION OF VARIABLES APPEARING IN STATECHART SHOWN IN FIGURE 3.

variable name	variable meaning	variable value	
		0 (false)	*1 (true)*
W1_NWP *W2_NWP*	switch position	switch is not at normal position	switch is at normal position
W1_RWP *W2_RWP*	switch position	switch is not at reverse position	switch is at reverse position
SE3_AS *SE4_AS*	*route availability*	*route is available and to be locked*	*route is not locked*
D1_P *D2_P* *G2_P*	*section occupancy*	*section is occupied*	*section is unoccupied*
SE3_CANPB *SE4_CANPB*	*route cancel request*	*no request is received*	*a request is received*
PO	*initialization flag*	*system is initializing.*	*system finishes initialization*

As discussed above, continue applying the rules to the specific station layout, all involved transitions in the statechart can be generated.

Once statechart is finished, we can get Boolean expressions of variable G_2_WS and G_2_ES separately. Take variable G_2_WS as an example, the "*Off*" set is:

$$Off = \sim S_4\_AS * W_2\_NWP. \quad (2)$$

And the "*On*" set is:

$$On = W_2\_RWP * G_2\_P * \sim PO + D_2\_WS * W_2\_NWP * G_2\_P * \sim PO + W_2\_RWP * G_2\_P * \sim D_1\_P + D_2\_WS * W_2\_NWP * G_2\_P * \sim D_1\_P + D_2\_WS * W_2\_NWP * G_2\_P * S_4\_CANPB. \quad (3)$$

Applying (1) discussed in section 2, we can get the Boolean expression for variable G_2_WS is shown as follows:

Assign ~ (~S_4_AS* W_2_NWP)*(W_2_RWP * G_2_P * ~PO + D_2_WS * W_2_NWP * G_2_P * ~PO + W_2_RWP * G_2_P * ~D_1_P+ D_2_WS * W_2_NWP * G_2_P * ~D_1_P + D_2_WS * W_2_NWP * G_2_P * S_4_CANPB +G_2_WS) to G_2_WS. (4)

The above expression is a little redundant and disordered. The expression can be synthesized and rearranged as follows. The algorithm is omitted here.

Assign (S_4_AS + ~W_2_NWP)*((~PO + ~D_1_P)* W_2_RWP * G_2_P + (~PO + ~D_1_P + S_4_CANPB)* D_2_WS * W_2_NWP * G_2_P + G_2_WS) to G_2_WS. (5)

IV. IMPLEMENTATION

According to the principles introduced above, we implement a prototype software system. The system consists of three software tools. The first is a station layout editing tool with graphical interface. People can instantiate different kinds of wayside devices and compose a station layout. The software can generate topology data according to user's design. Fig. 4 shows the editing interface of this tool. The second tool is a relay logic generating tool. It takes station layout topology data as input, and generates relay logic of interlocking in the form of Boolean expressions. As we introduce above, this tool analyzes topology data, generates statecharts describing safety logic first, and then transforms statecharts to Boolean expressions. The general safety rules are integrated into this tool. The Boolean expressions are easy to read but can not be used directly by interlocking system, so the third software is provided to transform Boolean expressions to ladder diagram following standard IEC-61131-3. The ladder diagram can be directly parsed and executed by interlocking system to implement safety-critical functions. Fig. 5 shows the Boolean expressions generated according to station shown in Fig. 4 and Fig. 6 shows the transformed ladder diagram.

V. COMPARISON BETWEEN STATECHARTS-BASED DEVELOP MODEL AND TRADITIONAL DEVELOP MODEL

A. *Simplify interlocking logic design procedure and improve efficiency*

In traditional develop mode, the safety logic of interlocking is all designed by experienced signaling engineers. Once the layout changes, the whole logic is rewritten. Most of the time is wasted, and no logic module can be reused. The efficiency is poor and the cost is high. In this newly introduced develop mode, you need only to redraw the station layout using special tool shown in Fig. 4, and all the left works are fulfilled automatically by software tools as long as general safety rules are not changed. In practice, the time needed for designing safety logic for a small station is only tens of minutes.

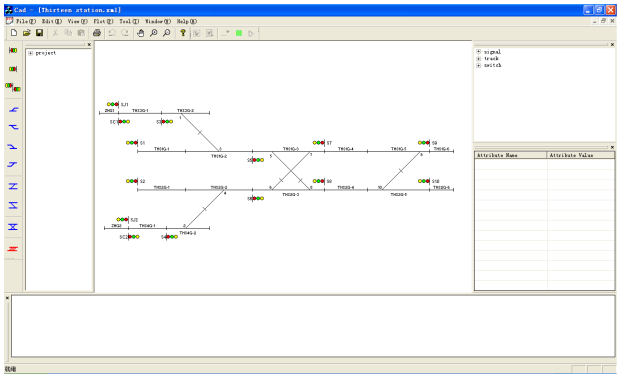

Figure 4. The tool used to specify station layout has a graph-oriented editing interface.

```
ASSIGN  SC2AS*(THS4G_1P+THS4G_1ES)  TO  THS4G_1ES;

ASSIGN  ~THS1G_4WRE*((THS1G_5WS+RWP10_9)*THS1G_4P+THS1G_4WS) TO  THS1G_4WS;

ASSIGN  S9RC*NWP10_9  TO  THS1G_4WRE;

ASSIGN  ~THS2G_4WRE*(THS2G_5WS*THS2G_4P+THS2G_4WS)  TO  THS2G_4WS;

ASSIGN  S3AS*(THS3G_2P+THS3G_2ES)  TO  THS3G_2ES;

ASSIGN  S7RC*NWP6_7*NWP5_8*RWP1_3+S8RC*RWP5_8*NWP6_7*RWP1_3  TO
    THS3G_2WRE;

ASSIGN  ~THS3G_2WRE*((THS1G_2WS+NWP1_3)*THS3G_2P+THS3G_2WS)  TO  THS3G_2WS;

ASSIGN  S3RC*RWP1_3  TO  THS1G_2ERE;
```

Figure 5. The Boolean expressions are generate automatically according to station layout shown in Fig. 4.

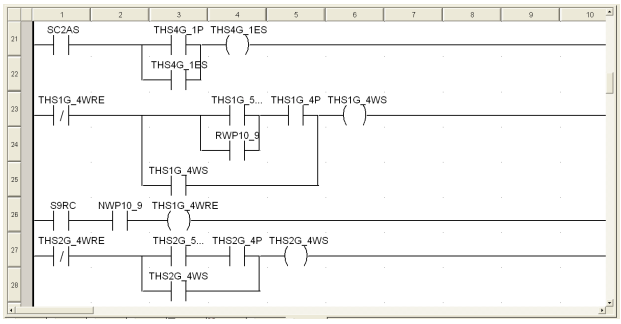

Figure 6. Ladder diagram transformed from Boolean expressions can be parsed and executed by interlocking.

B. *Logic described in form of statecharts is easy to understand and communicate*

As a kind of programming language presented in standard IEC61131-3, ladder diagram is a high-efficient and powerful low level language. But because of its complexity, it is not a kind of good language for people to communicate and exchange opinions. In contrast, statecharts use graphs to demonstrate logic relationship, so they are easier to

understand. The model introduced here uses statecharts to describe safety logic, and this is very useful for technical exchange between people.

C. Possibility to introduce formal verification

Validation is a very important step during interlocking development. Normally, simulation and test are the most popular techniques applied in validation. These techniques can catch many flaws in software, but cannot guarantee correctness. Formal verification is a better method in interlocking validation; this technique will check if the design satisfies specified signaling principles [9, 10, 11]. Statecharts is a classical formal language, and formal methods can be used to verify the logic described in statecharts. Some tools such as Statemate already provide powerful model checker to fulfill this task [5].

VI. CONCLUSIONS

This paper presents a new model of developing safety logic of interlocking system. This model introduces Statecharts to describe interlocking safety logic and uses software to generate Statecharts model automatically from station layout. The Statecharts model can be transformed to relay logic which can be executed in interlocking. This solution can improve efficiency a lot in interlocking development, and it also makes it possible to introduce formal verification technique in system validation. But there are still a lot of works need to be researched on this topic. For example, the general safety rules are integrated into the logic generating tools, so once the rules change the software needs to be update. Thus it is useful to separate safety rules from tools and provide developers an efficient way to define their own safety rules.

REFERENCES

[1] Fokko van Dijk, Wan Fokkink, Gea Kolk, Paul van de Ven and Bas van Vlijmen, "EURIS, a Specification Method for Distributed Interlocking," Lecture Notes in Computer Science, vol. 1516, pp.296-305, 1998.

[2] R.C. Short, "Safety Assurance of Configuration Data for Railway Signal Interlockings," The First Institution of Engineering and Technology International Conference On System Safety, pp. 174-178, 2006.

[3] Karim Kanso, Faron Moller, Anton Setzer, "Automated Verification of Signaling Principles in Railway Interlocking Systems," Electronic Notes in Theoretical Computer Science , vol. 250, pp. 19-31, 2009.

[4] D. Harel., "Statecharts: A Visual Formalism for Complex Systems," Sci. Comput. Programming , vol. 8, pp. 231-274, 1987.

[5] J. Klose, W. Damm, "Verification of a Radio-based Signaling System Using the STATEMATE Verification Environment," Formal Methods in System Design, vol. 19(2), pp. 121-141, 2001.

[6] Michele Banci, Alessandro Fantechi, "Geographical Versus Functional Modelling by Statecharts of Interlocking systems," Electronic Notes in Theoretical Computer Science, vol. 133, pp. 3-19, 2005.

[7] Eugenio Roanes-Lozano, Luis M. Laita, "An applicable topology-independent model for railway interlocking systems," Mathematics and Computers in Simulation, vol. 45, pp. 175-183, 1998.

[8] S. Maneis, K. Akantziotis, "Automated synthesis of Ladder automation circuits based on state-diagrams," Advances in Engineering Software, vol. 36, pp. 225-233, 2005.

[9] Nelson Guimaráes Ferreira and Paulo Sérgio Muniz Silva, "Automatic Verification of Safety Rules for a Subway Control Software," Electronic Notes in Theoretical Computer Science, vol. 130, pp. 323-343, 2005.

[10] Vicky Hartonas-Garmhausen, Sergio Campos, Alessandro Cimatti, Edmund Clarke, Fausto Giunchiglia, "Verification of a safety-critical railway interlocking system with real-time constraints," Science of Computer Programming, vol. 36, pp. 53-64, 2000.

[11] A.Cimatti, F. Giunchiglia, G. Mongardi, D. Romano, F. Torielli and P. Traverso, "Formal Verification of a Railway Interlocking System using Model Checking," Formal Aspects of Computing, vol. 10, pp. 361-380, 1998.