Formal Methods && Tools Group
IEI-CNUCE (ISTI) - Area della Ricerca CNR - Via G. Moruzzi, 1 - 56124 PISA

ALSTOM

Technical Report

Model based design of interlocking systems

Index

Index of figures

1 References

[1] A. Torchi, *"SMARTLOCK - Generazione delle equazioni booleane da specifiche funzionali espresse in forma di diagrammi di stato"*, 2004

[2] M. Banci, *"Schemi di Principio espressi con Statemate - SCENARI DI TEST"*, ISTI-FMT internal report, 2004

[3] M. Banci, *"RIS - model report"*, ISTI-FMT internal report, 2004

[4] M. Banci, *"COMANDO_1 – model report"* ,ISTI-FMT internal report, 2004

[5] M. Banci, *"COMANDO_3_5 – model report"*, ISTI-FMT internal report, 2004

[6] M. Banci, Presentation at Progress Review Meeting in Pisa, 22nd January 2004 *"Progress Review Meeting Report"*

2 Introduction

This report describes the results of the collaboration between ISTI-CNR and ALSTOM performed from 1-9-2003 to 31-3-2004, finalized to study the impact of a model-based specification of interlocking systems based on the Statechart formalism on the current techniques and practices adopted in ALSTOM for the development of interlocking software with high safety integrity level.

The results of the collaboration are reported starting form section 3, in which a model-based specification of an interlocking system for an example station is shown. This model has been developed using the Statemate tool. Actually, two separate models have been developed, the first one which directly models the interlocking controlling one specific station layout, the second one which starts form a generic modelling of the signalling principles, to be then instantiated on different specific layouts. With this second model, two different layouts have been instantiated from the same generic Statecharts description.

Section 4 shows the simulations performed with the models to verify their overall correctness.

The next two sections refer to two separate documents, one containing a study on the feasibility of translating Statecharts into boolean equations [1], the other containing a study about the possibility of generating test cases and scenarios from a Statechart specification [2].

The final chapter reports a list of open topics that need to be further analysed and studied in a future collaboration.

3 The system modelled using Statemate

The interlocking system, the definition of which has been supplied from ALSTOM, has been modelled using an high level language named Statecharts and, in particular, the Statemate Statecharts has been used. I-Logix Statemate is a comprehensive graphical modelling and simulation tool for the development of complex embedded systems.

Statemate Statecharts allow to model reactive systems. The Statemate language includes a lot of different statements and allows the construction of very complex applications.

However, a railway interlocking system is a simpler sort of application, which needs only a small part of the language power to be modelled. This peculiarity will turn to be useful, since it will allow to restrict the language for railway applications, for which a set of tools can be more conveniently designed and developed.

Interlocking systems which are controlled by a set of boolean equations were originally investigated. The study started modelling one actual implementation of the interlocking, then a new model was developed, abstracting it from the implementation. Consequently, two Statemate models have been implemented:

- The first model, non parametric and with logic functionalities coming from relay-based circuits, consists on implementing the (relay based) boolean equations in a Statecharts model. This first step was needed to understand the behaviour of the model with respect to the behaviour of the system, and to study the correspondence between them.
- The second model is a generic model, which can be instantiated on different station layouts, which includes parametric elements and logic functionalities independent from relay-based circuits; two example instantiations, using two different layouts, have been provided.

Both models have been developed in the Statemate environment, and have been used to analyse the different relevant Statemate features. These features are mainly the following ones:

- simulation;
- verification reporting;
- mimic panel animation;
- possibility of exporting the model in text-based format;
- access to the Statemate model internal information using a particular API (DataPort);
- testing.

3.1 The first model

The first model is strictly related to the relay based technology: in fact the model consists of a number of similar charts, each one implementing the behaviour of a Boolean equation, in a way derived from the corresponding relay schemes in a one-to-one correspondance.
In Figure 1 an example of chart that models only one equation can be found.

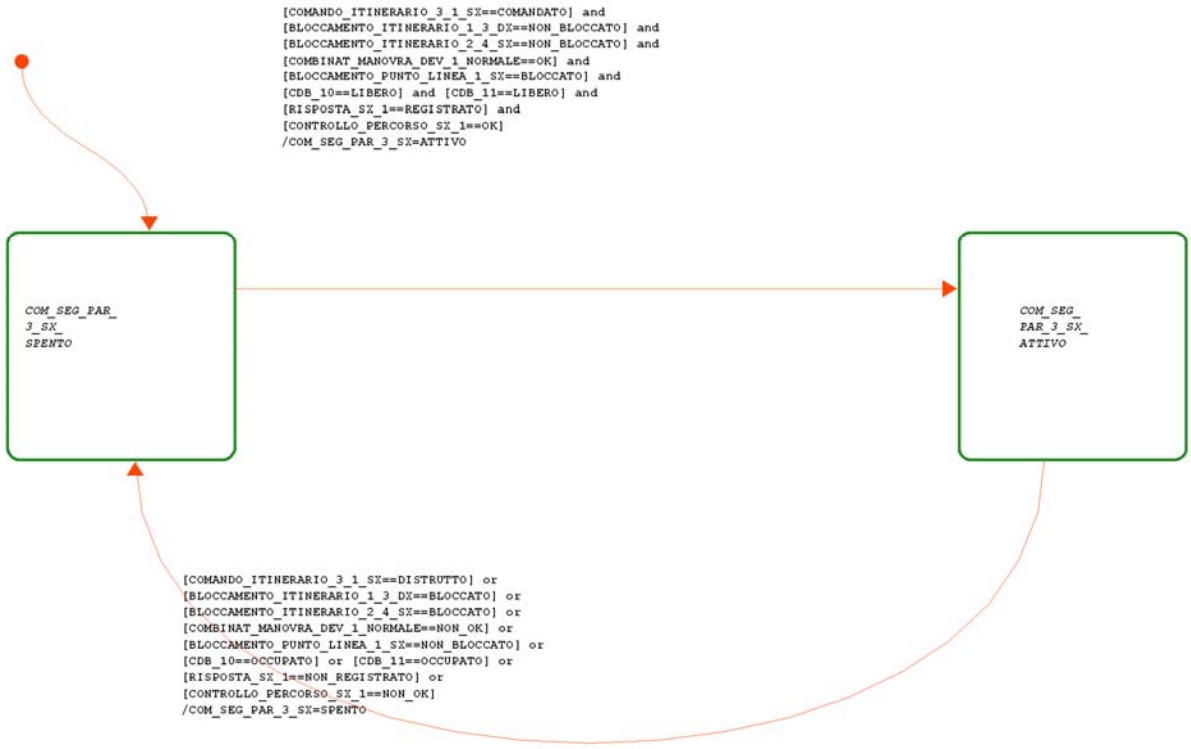

Figure 1 - A chart modelling a Boolean equation

The first model has been mainly implemented for investigating the functionalities offered by the Statemate tool. This approach was required in order to have a proficient cooperation and because an existing model to train the team was not available.

3.1.1 Description of the model

The model consists of two parts: the interlocking system block (RIS = Railway Interlocking System) and the field devices block (ENTI_DI_PIAZZALE). The second part consists in modelling the behaviour of physical devices included in the railway yard.

Figure 2 shows the first level of the model (SYSTEM), two blocks (named *'activities'* in Statemate terminology), which contain sublevels.

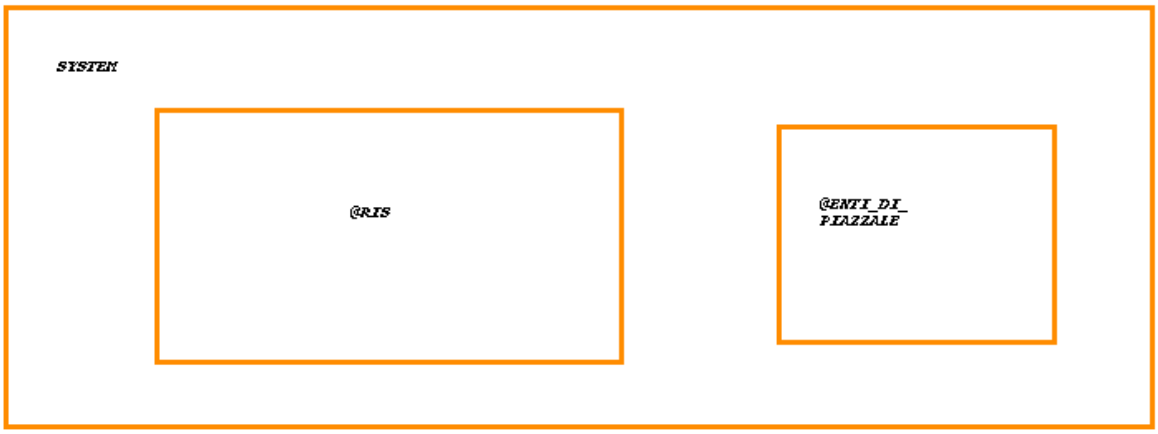

Figure 2–The main level of the model

Figure 3 shows a particular blocks grouping similar to that used in Italian relay based interlockings, where there are different phases in which different verifications are performed.

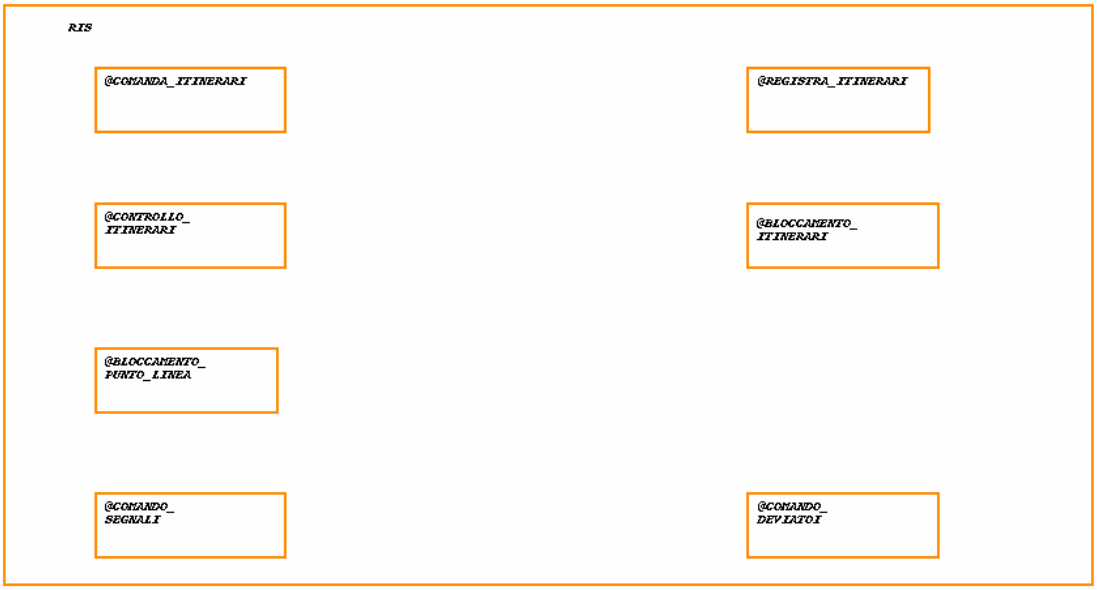

Figure 3 - The RIS (Railway Interlocking System) block

The system modelled using Statemate

This kind of implementation was used in the first model, in order not to deviate from a well known implementation of the system.

The objective was to understand the modelling methodology, therefore the railway field devices *(enti di piazzale)* have been modelled as well.

At this level there are 7 different phases, that is: COMANDO_ITINERARI (routes command), REGISTRA_ITINERARI (routes setting), COMANDO_DEVIATOI (switch points command), CONTROLLO_ITINERARI (routes verification), COMANDO_SEGNALI (signals command), BLOCCAMENTO_ITINERARI (routes locking), BLOCCAMENTO_PUNTO_LINEA (line section locking). Each of these blocks is further split up into lower level parts.

As an example the routes command block, shown in Figure 4, includes 8 activities. These are the command activities for each of the 8 routes which have have implemented. The same decomposition has been done for each main block.

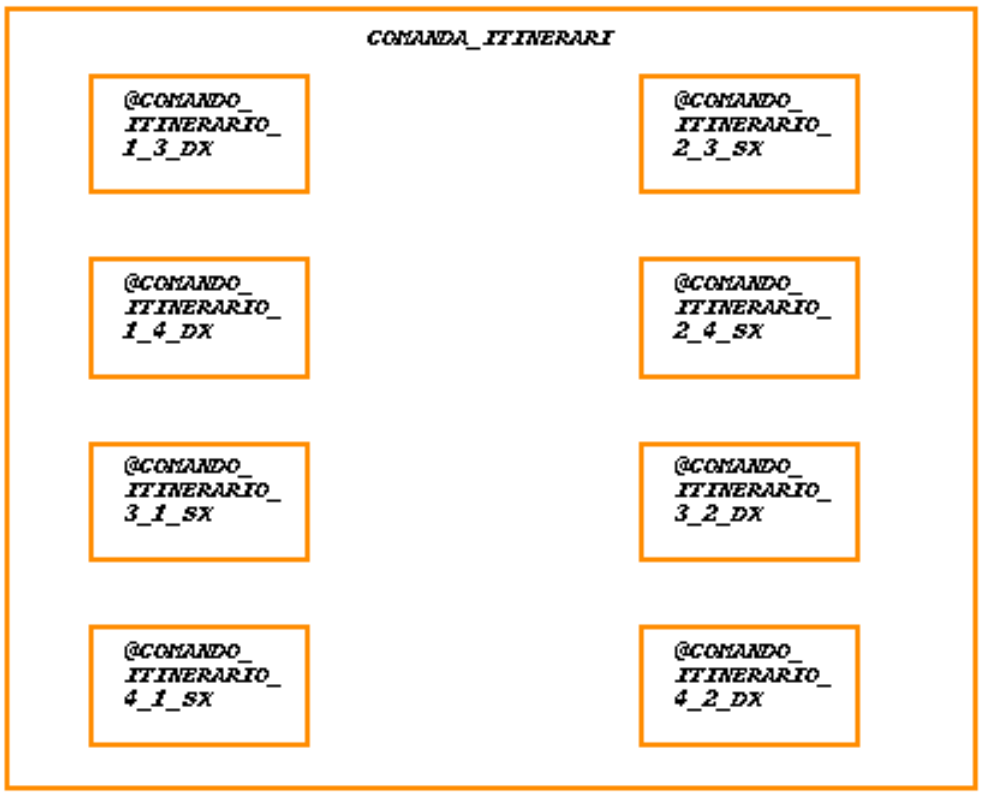

Figure 4 - 8 routes

The bottom of the hierarchical structure consists at the lowest level of a single chart which reproduces the behaviour of a Boolean equation, as implemented in the interlocking system (see Figure 5).

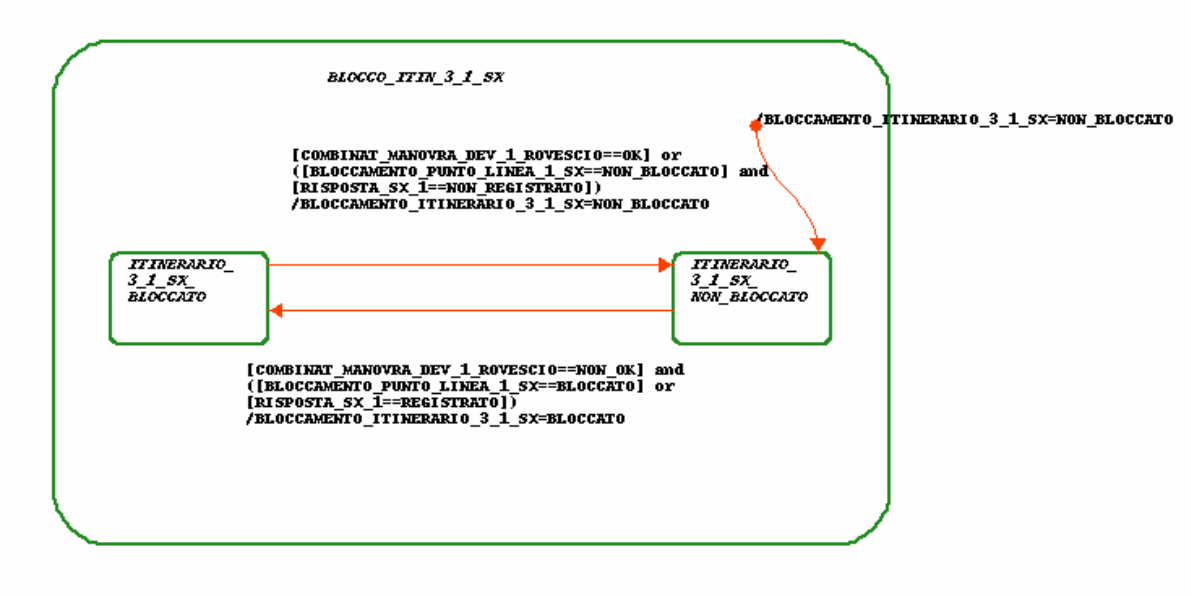

Figure 5 - The lowest level

For further details, and for the description of the entire model, see [3].

3.2 The second model, using generic charts

The second model has been implemented by using "*generic charts*" of which as many instances as required can be used.

Another difference from the first model is the abstraction of the model, that is implemented using a design methodology not based on relays. Therefore the model is more compact than the first one, and also more readable.

3.2.1 Description of the model

The model consists of the same two main blocks implemented for the first one (Figure 2), but the activities the model has to perform have been implemented in a different way, as it is shown in Figure 6.

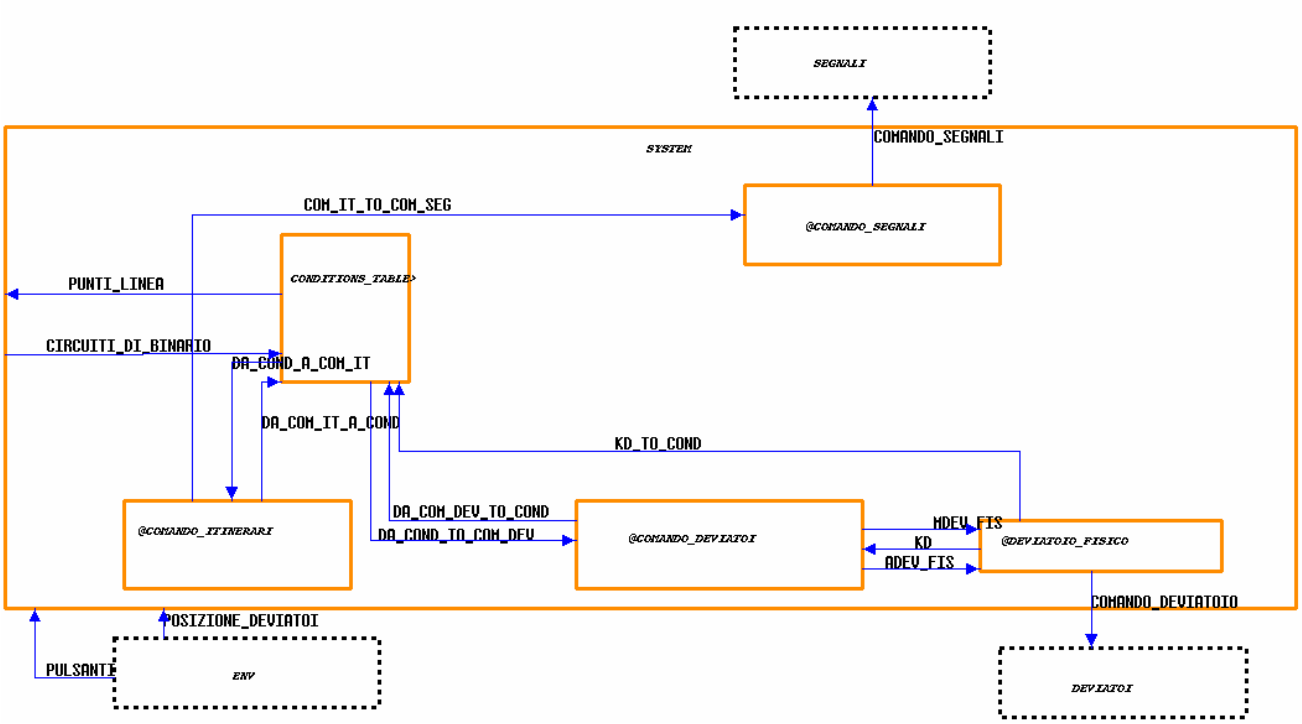

Figure 6 - The main structure blocks

The interlocking system model shown in Figure 6 is composed by 5 activity charts: COMANDO_SEGNALI (signals command), COMANDO_ITINERARI (routes command), COMANDO_DEVIATOI (switch points command), DEVIATOIO_FISICO (Physical switch points) and CONDITIONS_TABLE.

The external environment activity simulates the behaviour of the environment by sending to the system button commands and switch point positions, while the external activities COMANDO_SEGNALI and COMANDO_DEVIATOI receive signals from the system.

The external activities are out of the system, in fact they are only simulated by the simulator tool and the associated panel (Figure 18).

The activities above will be now accurately described.

3.2.1.1 COMANDO_ITINERARI (routes command)

The objective of this activity is to manage the command of a route. By receiving signals from the environment, the other activities and the other statecharts, the activity is elaborates these signals and decides accordingly which is the correct command to perform.

The behaviour of this activity is well understandable referring to the Figure 7, where the states and transitions that control the activity are illustrated.

Of the 8 states, the initial one is N_COM_IT which is an idle state where the signal of the route is red and some others variables of the route are assigned in a way that not permit, in a safe-mode, the occupation of the route.

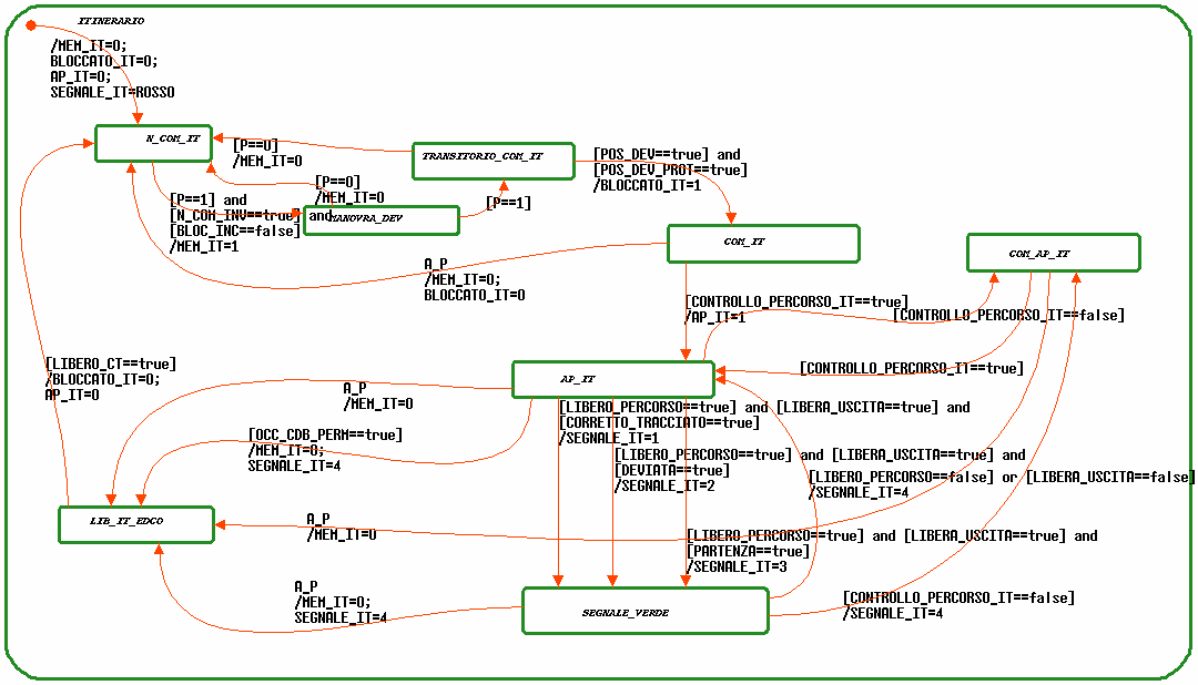

Figure 7 - Generic chart with generic terms

From this state, in consequence of pressing a button, the transition to another state is possible only in presence of correct values of some other variables. Each transition from a state to another is formed from a set of conditions, to which control variables have to obey. These conditions perform a safety control on the evolution of the command. This action pass through 5 intermediate states:

N_COM_IT :	Starting point, idle.
MANOVRA_DEV :	State where switch points are commanded (if it is possible), otherwise goes back to N_COM_IT
TRANSITORIO_COM_IT :	The system is waiting the stabilization of the commands related to physical equipments
COM_IT:	When each physical device is in a correct position, the route is blocked and the system passes to this state.
AP_IT :	Before to arrive in this state the system controls the correctness of each device again, by using different values of the variables, and if everything is correct passes to AP_IT, which is a logical (not physical) place in the track layout.
SEGNALE_VERDE :	After other controls about possibly conflicting route commands the green can be turned on green.

3.2.1.2 COMANDO_DEVIATOI (switch points command)

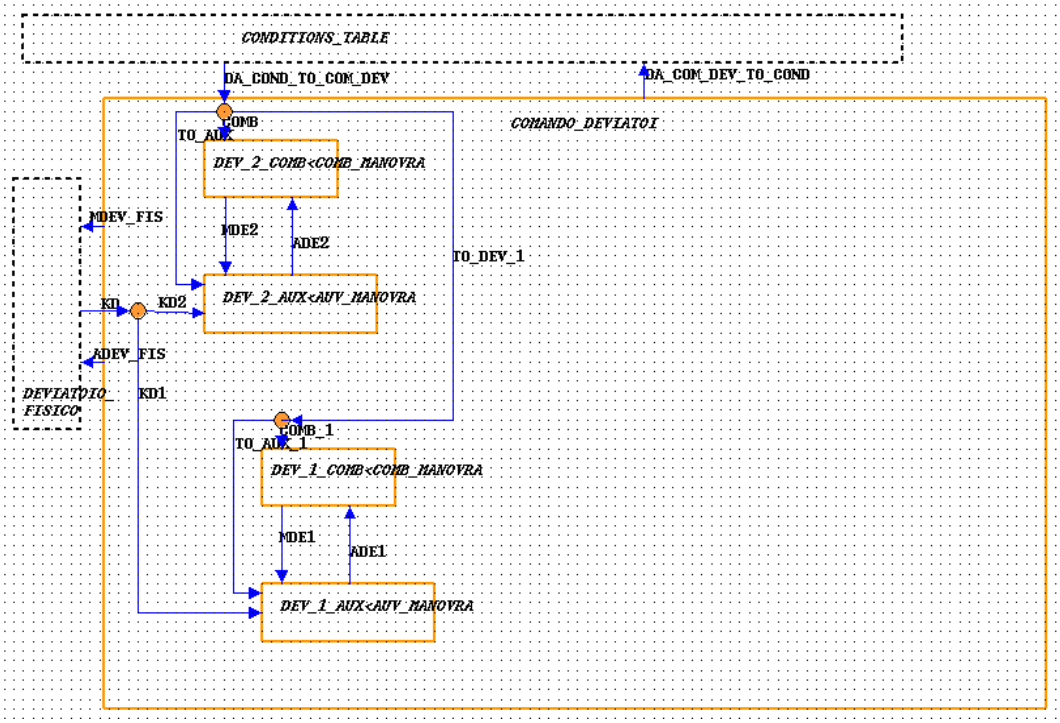

Figure 8 – The COMANDO_DEVIATOI activity

It is composed by the instances of the two switch points. Every switch point is composed by two activities: COMB_MANOVRA and AUV_MANOVRA that interact each other.

COMB_MANOVRA: It commands the direction of the motor rotation.

AUV_MANOVRA: It commands the motor action.

3.2.1.3 DEVIATOIO_FISICO (Physical switch points)

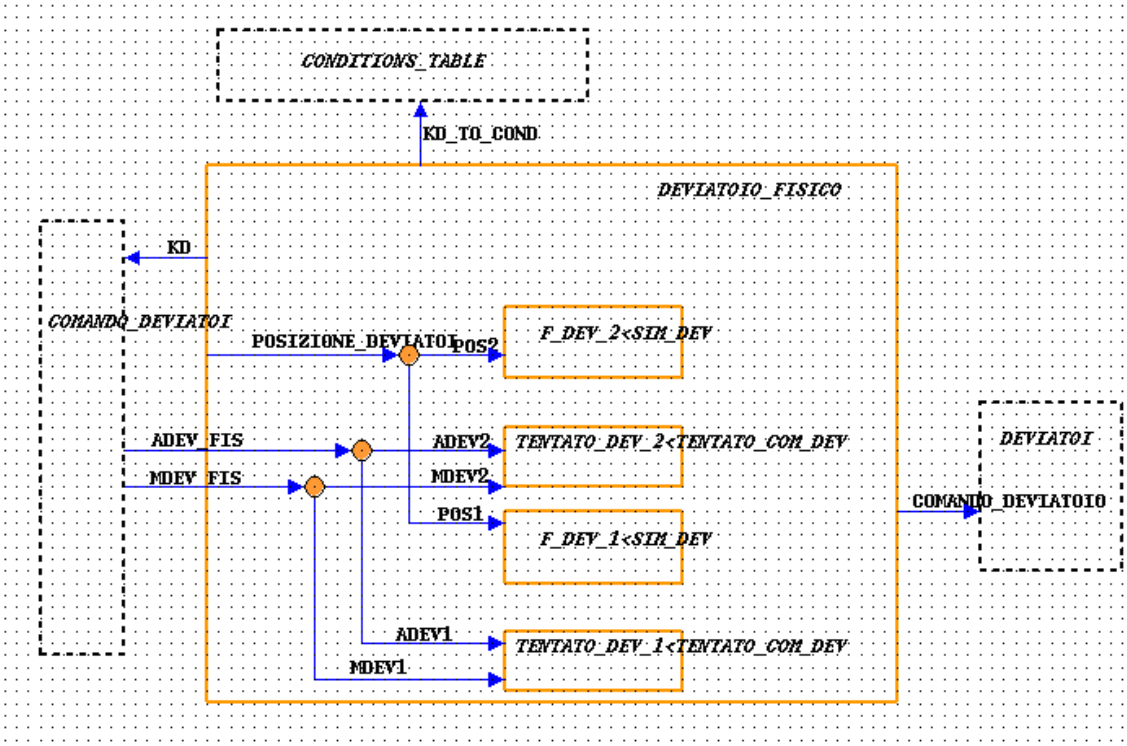

Figure 9 – The DEVIATOIO_FISICO activity

This activity check the position of the physical switch points.

It is composed by the instances of the two switch points. Every switch point is composed by two activities: TENTATO_COM_DEV and STM_DEV.

TENTATO_COM_DEV: It checks that the command signals are correct.

STM_DEV: It commands the motor action.

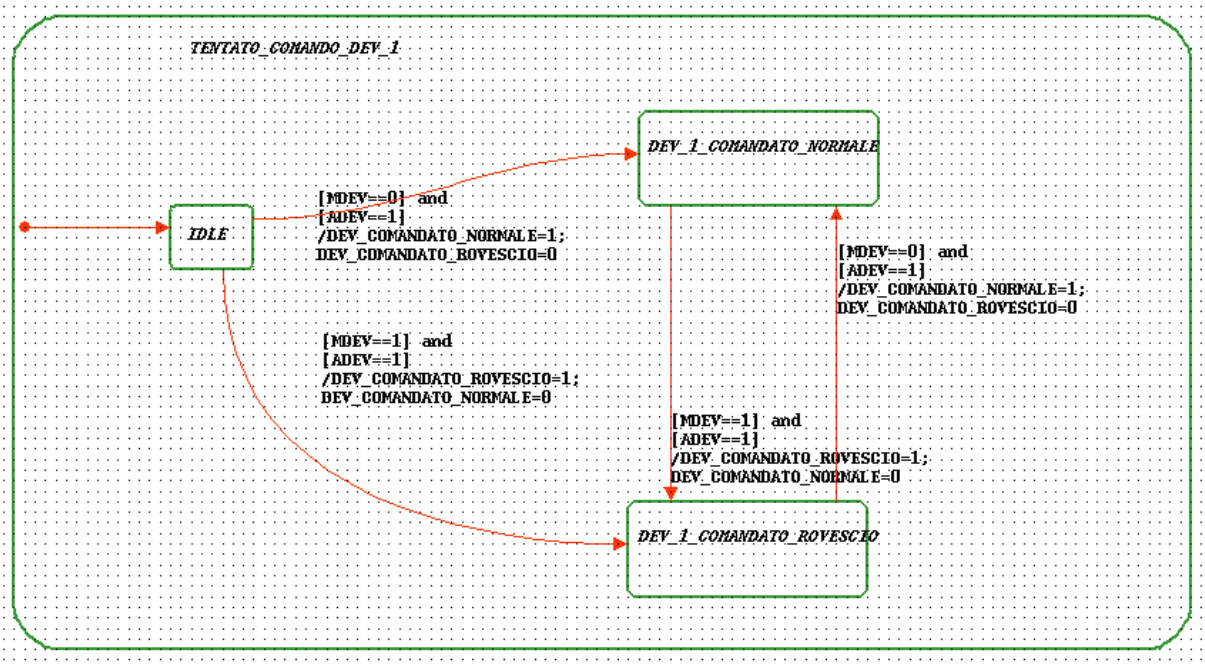

Figure 10 – The TENTATO_COMANDO_DEV generic chart

In Figure 10 the generic chart that commands the position of the switch point (normal and reverse) is shown.

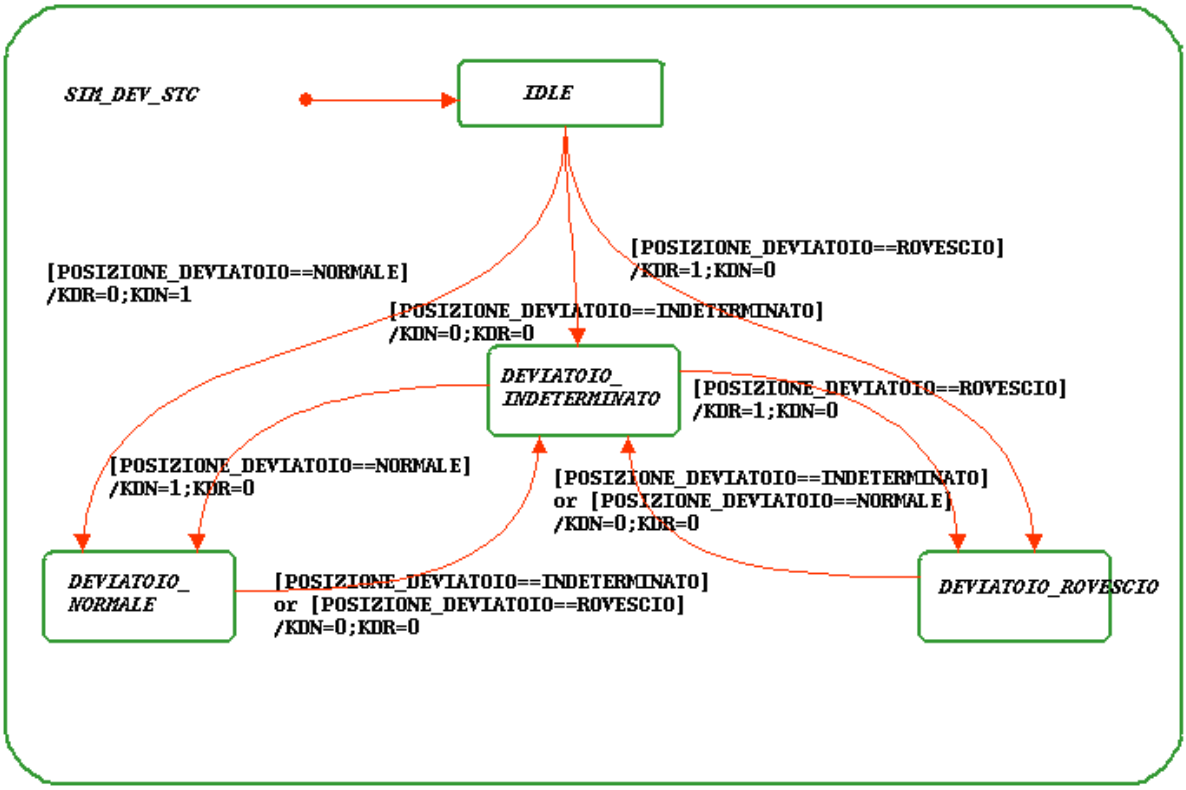

Figure 11 – The SIM_DEV_STC generic chart

The system modelled using Statemate

In Figure 11 the generic chart that control the position of the switch point (normal and reverse) is shown.

3.2.1.4 COMANDO_SEGNALI (signals command)

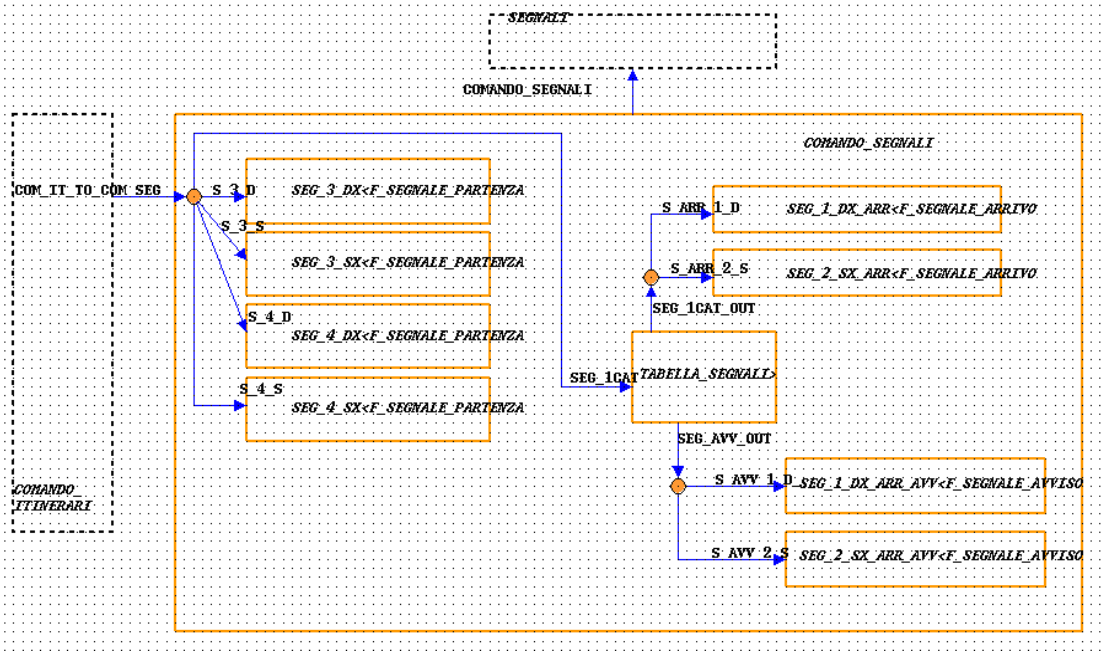

Figure 12 – The COMANDO_SEGNALI activity

This activity (Figure 12) commands the signals of the interlocking system.

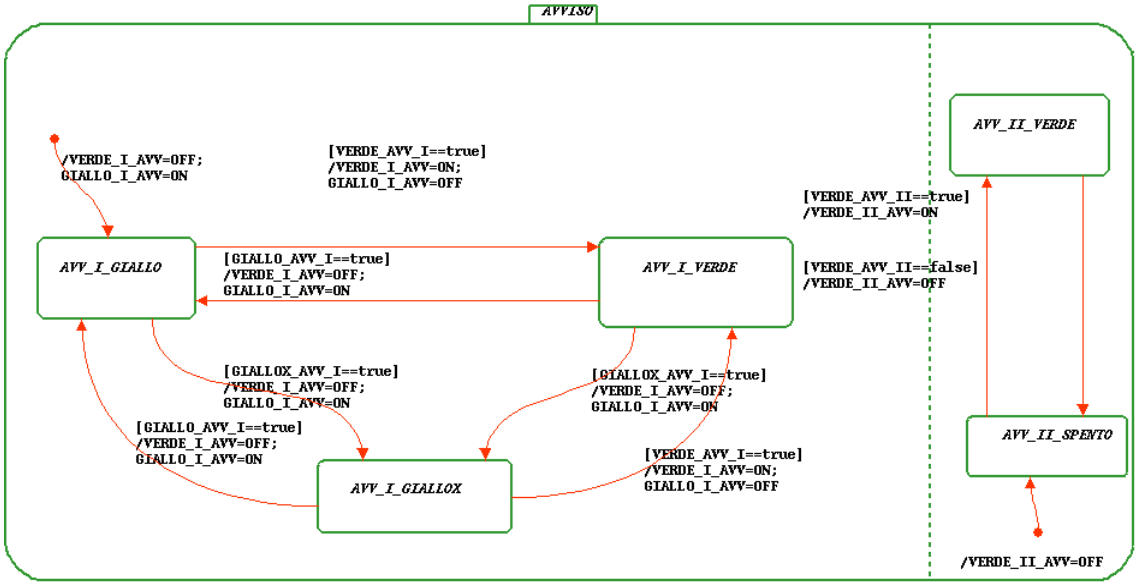

Figure 13 – A signal generic chart.

The system modelled using Statemate

3.2.1.5 CONDITIONS_TABLE

```
LIBERO_CT_6_2_D_PAR=(CDB_20==true) and (CDB_21==true);
N_COM_INV_1_3_D_ARR=(MEM_IT_3_1_S_PAR==0);
N_COM_INV_1_4_D_ARR=(MEM_IT_4_1_S_PAR==0);
BLOC_INC_1_3_D_ARR=(BLOCCATO_IT_1_4_D_ARR==1) or (BLOCCATO_IT_4_1_S_PAR==1) or
(BLOCCATO_IT_3_2_D_PAR==1) or (BLOCCATO_IT_3_1_S_PAR==1) or (BLOCCATO_IT_2_3_S_ARR==1);
BLOC_INC_1_4_D_ARR=(BLOCCATO_IT_1_3_D_ARR==1) or
(BLOCCATO_IT_4_1_S_PAR==1) or (BLOCCATO_IT_2_4_S_ARR==1) or (BLOCCATO_IT_3_1_S_PAR==1) or (BLOCCATO_IT_4_2_D_PAR==1);
COMANDATO_NORMALE_1=(MEM_IT_1_3_D_ARR==1) or (MEM_IT_3_1_S_PAR==1);
COMANDATO_ROVESCIO_1=(MEM_IT_1_4_D_ARR==1) or (MEM_IT_4_1_S_PAR==1);
LIBERI_CDB_IMMOBILIZZAZIONE_1=(CDB_11==true);
POS_DEV_1_3_D_ARR=(MDEV_1==0);
POS_DEV_1_4_D_ARR=(MDEV_1==1);
POS_DEV_PROT_1_3_D_ARR=true;
POS_DEV_PROT_1_4_D_ARR=true;
CONTROLLO_PERCORSO_IT_1_3_D_ARR=(KDN_1==1);
CONTROLLO_PERCORSO_IT_1_4_D_ARR=(KDR_1==1);
LIBERO_PERCORSO_1_3_D_ARR=(CDB_10==true) and (CDB_11==true) and (CDB_I==true);
LIBERO_PERCORSO_1_4_D_ARR=(CDB_10==true) and (CDB_11==true) and (CDB_II==true);
CORRETTO_TRACCIATO_1_3_D_ARR=true;
CORRETTO_TRACCIATO_1_4_D_ARR=false;
DEVIATA_1_3_D_ARR=false;
DEVIATA_1_4_D_ARR=true;
PARTENZA_1_3_D_ARR=false;
PARTENZA_1_4_D_ARR=false;
LIBERA_USCITA_1_3_D_ARR=true;
LIBERA_USCITA_1_4_D_ARR=true;
OCC_CDB_PERM_1_3_D_ARR=(CDB_10==false);
OCC_CDB_PERM_1_4_D_ARR=(CDB_10==false);
LIBERO_CT_1_3_D_ARR=(CDB_10==true) and (CDB_11==true);
LIBERO_CT_1_4_D_ARR=(CDB_10==true) and (CDB_11==true);

N_COM_INV_3_2_D_PAR=(MEM_IT_2_3_S_ARR==0);
BLOC_INC_3_2_D_PAR=(BLOCCATO_IT_2_3_S_ARR==1) or (BLOCCATO_IT_2_4_S_ARR==1) or (BLOCCATO_IT_4_2_D_PAR==1);
LIBERI_CDB_IMMOBILIZZAZIONE_2=(CDB_21==true);
POS_DEV_3_2_D_PAR=(MDEV_2==0);
POS_DEV_PROT_3_2_D_PAR=true;
CONTROLLO_PERCORSO_IT_3_2_D_PAR=(KDN_2==1);
LIBERO_PERCORSO_3_2_D_PAR=(CDB_20==true) and (CDB_21==true);
CORRETTO_TRACCIATO_3_2_D_PAR=false;
DEVIATA_3_2_D_PAR=false;
PARTENZA_3_2_D_PAR=true;
LIBERA_USCITA_3_2_D_PAR=true;
OCC_CDB_PERM_3_2_D_PAR=(CDB_20==false);
LIBERO_CT_3_2_D_PAR=(CDB_20==true) and (CDB_21==true);

COMANDATO_NORMALE_2=(MEM_IT_3_2_D_PAR==1) or (MEM_IT_2_3_S_ARR==1);
COMANDATO_ROVESCIO_2=(MEM_IT_4_2_D_PAR==1) or
(MEM_IT_2_4_S_ARR==1) or (MEM_IT_2_5_S_ARR==1) or (MEM_IT_5_2_D_PAR==1) or (MEM_IT_2_6_S_ARR==1) or (MEM_IT_6_2_D_PAR==1);

COMANDATO_NORMALE_3=(MEM_IT_5_2_D_PAR==1) or (MEM_IT_2_5_S_ARR==1);
COMANDATO_ROVESCIO_3=(MEM_IT_6_2_D_PAR==1) or
(MEM_IT_2_6_S_ARR==1) or (MEM_IT_2_5_S_ARR==1) or (MEM_IT_5_2_D_PAR==1);
```

Figure 14 – A part of the list of boolean equations

In Figure 14 is shown a part of the list of boolean equations that are implemented into the conditions_table. These equations expand the generic terms used.

As it can be seen, in the second model there are less activity charts then the first model, because some functionalities have been integrated into high level charts. For this reason the second model consists of a smaller number of charts than the first one, but with increased complexity. In other words, the complexity has been moved from the transitions to the states (Figure 6) and from a lot of different little concurrent charts into only a bigger one.

At a lower level, the same 8 routes commands can be seen, which, differently from the first model, have been instantiated 8 times from the same generic chart (Figure 15).

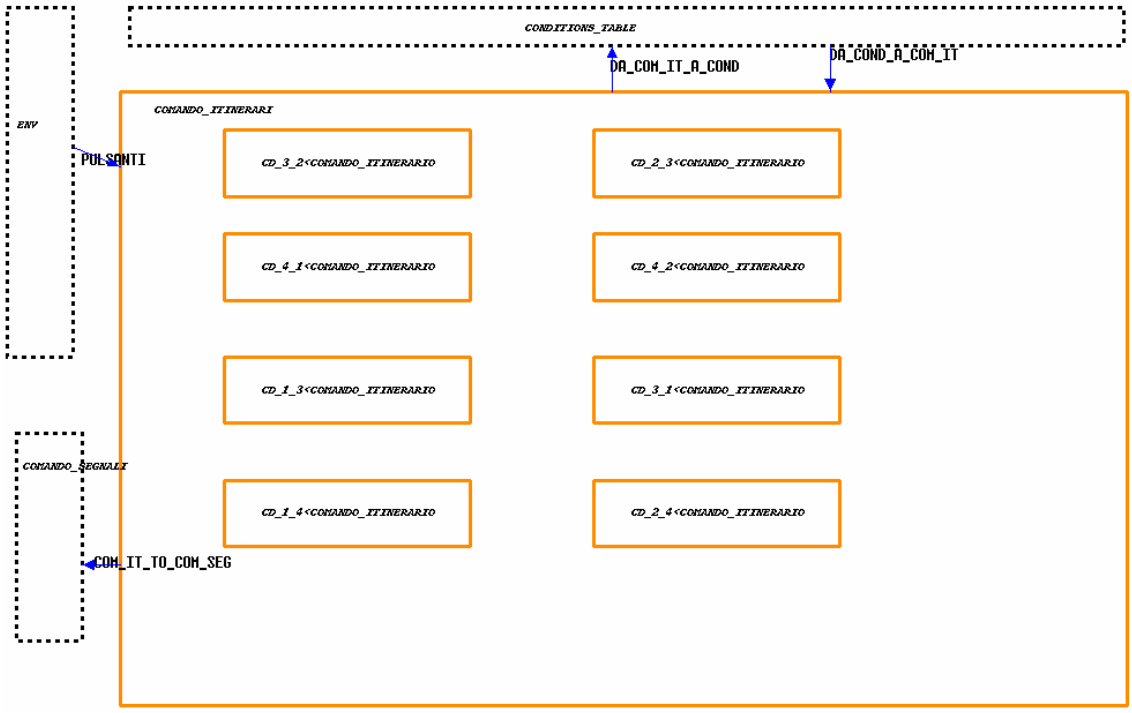

Figure 15 - Instances of the route command generic chart

The generic charts feature allows reusing parts of the specification. A generic chart makes it possible to represent common portions of the model as a single chart that can be instantiated in many places, similarly to a procedure in a conventional programming language. Generic charts are linked to the rest of the model via parameters.

Parameters are used to specify the particular instances of a generic chart and to link it to its environment. Parameters are the main feature by which an instance of a generic chart is able to share data with the rest of the model. Each generic chart has a set of *formal parameters*.

The parameters are defined explicitly by the designer in the Data Dictionary entry of the generic chart. They are given by their name, element type and mode. Each formal parameter has a Data Dictionary entry in which more information about the element can be added, such as its structure and data-type.

For each instance of a generic chart there is a binding of actual elements to the formal parameters, where the actual binding must have the same type and structure as the formal parameter.

As an example, in Figure 16 a generic chart and 3 formal parameters SEGNALE_IT, ROSSO, VERDE are shown.

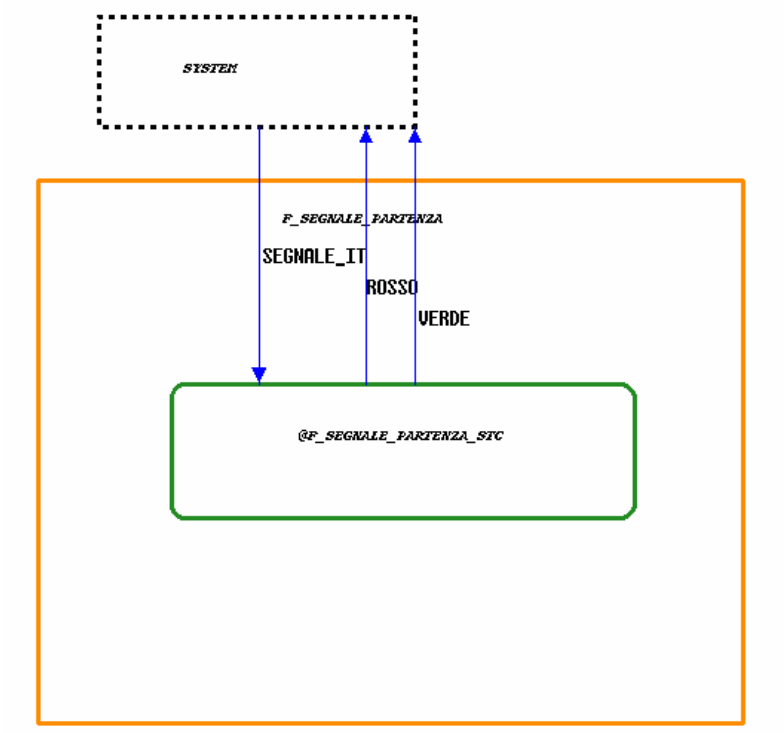

Figure 16 – Generic charts and formal parameters

3.2.1.6 Description of the usage of the generic charts

The use of generic chart is strictly related to replication, e.g. if there are 3 channels, the activity that manages a single channel 3 times can be replicated simply changing the actual parameters.

The implemented model is even more complex, because it also includes generic terms (where for terms both the conditions and the actions which are associated to a transition of a Statechart are intended).

In fact, as shown in Figure 17, there are no terms related to a specific route, but there are only generic terms, e.g. P, A_P, MEM_IT. These terms have to perform different elaborations, when different routes are concerned.

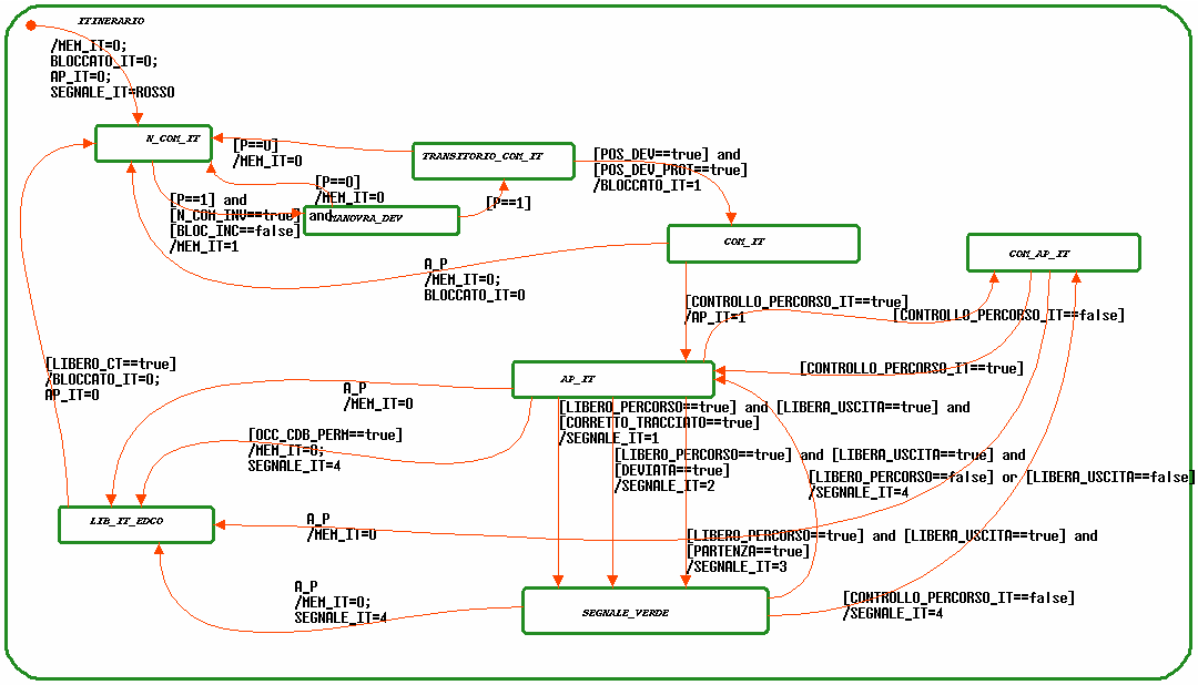

Figure 17– Generic chart with generic terms

For this reason the generic terms have to be expanded with different expressions, which are collected in one additional activity, named CONDITION_TABLE (Figure 14).

This activity is actually the core of the model, since it includes all the rules that allow to calculate the interlocking logic.

For further details related to the implemented model see [4], [5].

4 Simulation of the models

The models have been simulated by means of the Statemate simulator, which is embedded in the environment. The simulation is carried on sending the correct inputs to the models using a simulation panel. As an example, the panel in Figure 18 includes the station track plan and the buttons that allow to simulate inputs from the environment and to analyse the outputs elaborated by the model.

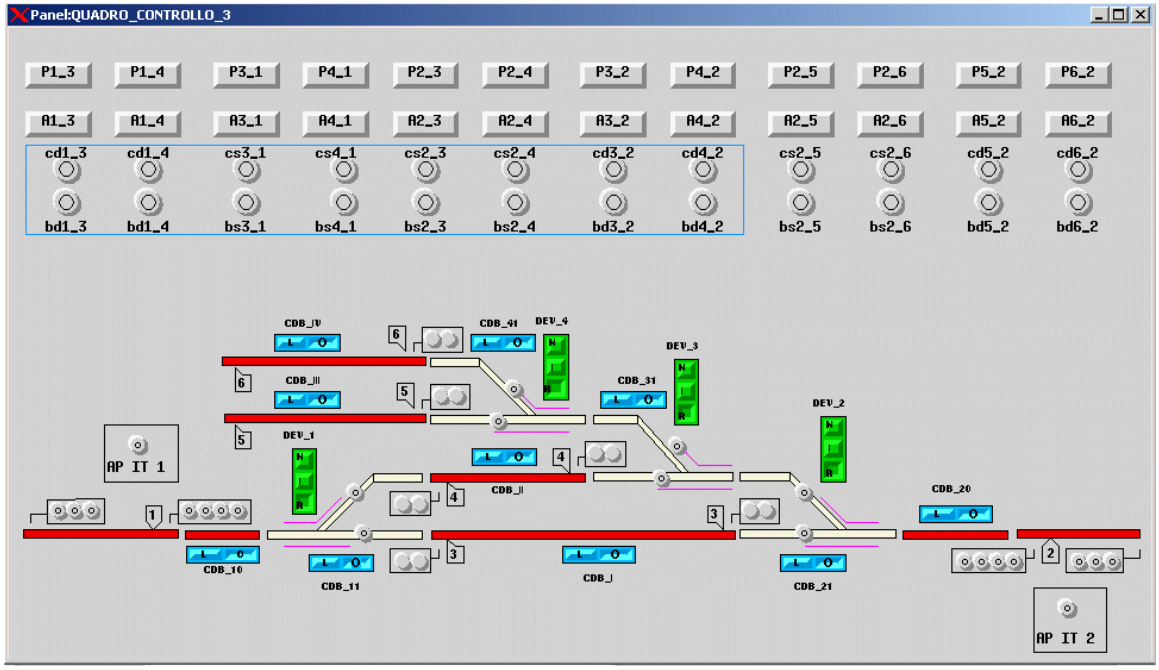

Figure 18– Railway panel

The simulation has been used to verify the correctness of the modelling; one tool which is particularly useful to visually check the correct behaviour of the model is the one that displays the real-time evolution of each variable of the system as waveforms (Figure 19).

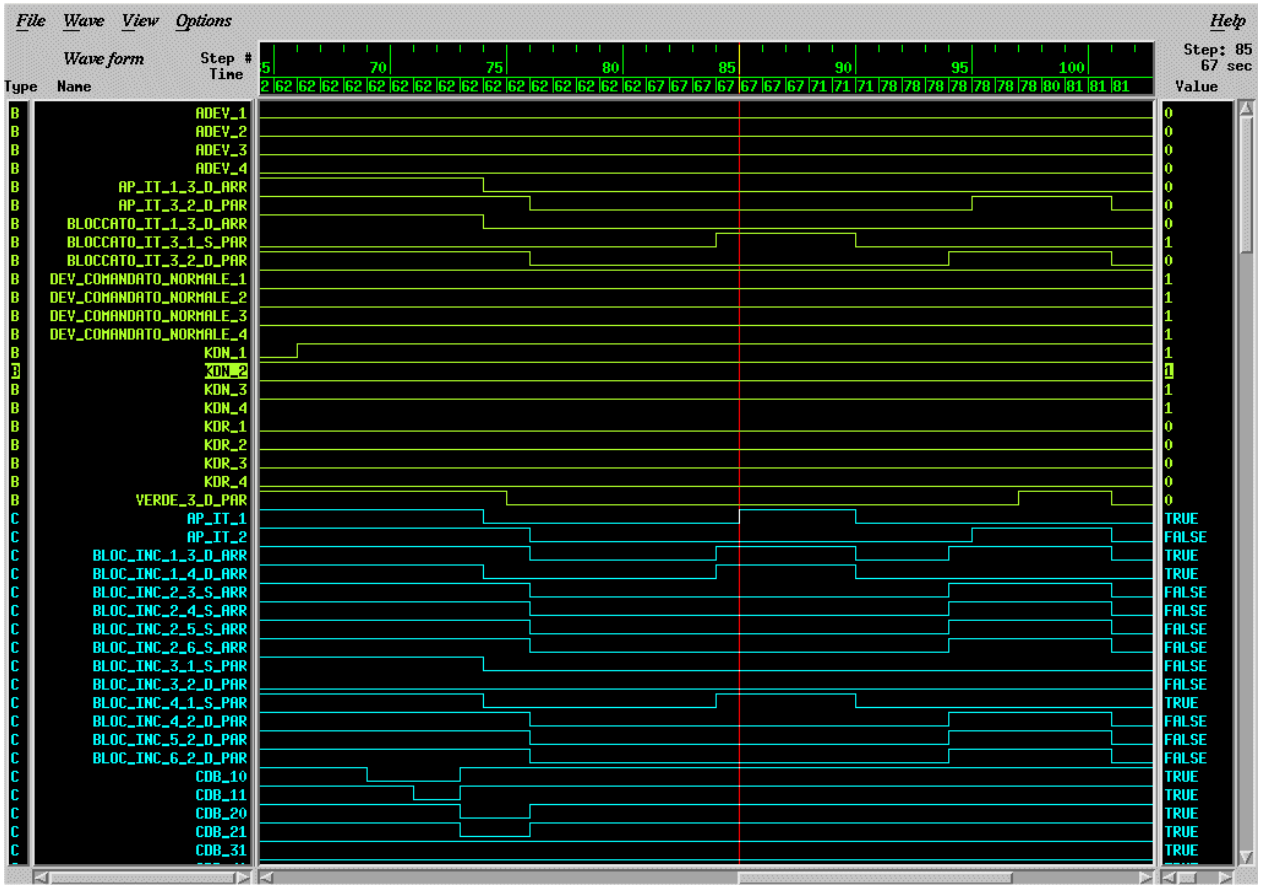

Figure 19– Simulation waveforms

5 Data format and data access offered by the Statemate tool.

During the collaboration several functionalities of the tool have been investigated, with the aim to find out the possibilities of integrating the tool with other tools and to convert Sattemate proprietary formatted data into data suitable for proprietary ALSTOM tools or for other commercial or publicly available tools.

5.1 Model Import-Export possibilities

The tool allows to export the model in a .TXT format, with a proper syntax.

Afterwards we can Import the .TXT file

The .TXT file contains all information about the model needed to recreate it, even with graphic details.

```
chart-file format version 4.1
-- Statemate MAGNUM version 3.2
chart :
      name : F_BLOCCO_AUTOMATICO_1_GRAFO
      type : STATECHART
      usage : REGULAR
      created : 1068805430  -- Fri Nov 14 11:23:50 2003
      creator : banci
      modified : 1069244683  -- Wed Nov 19 13:24:43 2003
      scale with zoom : NO
      arrow style : SPLINE
end chart

state :
      name : STATE#0
      type : DIAGRAM
      line width : 0
      color : BLACK WHITE OFF
      name color : WHITE WHITE OFF
      name font : Fixed 0
      name alignment : Left ExtremeBottom
      graphics coordinates :
            0.0000000000 0.0000000000
            0.0000000000 19.2000000000
            30.3733528551 19.2000000000
            30.3733528551 0.0000000000
end state

state :
      name : BA_1_OCCUPATO
```

Figure 20 – The Statemate export format

5.2 LSC (Live Sequence Chart)

Live sequence Charts (LSCs) highlight the interaction between the model and the environment (Figure 21), and between the different automata of the model. Below we show an example LSC derived form the simulation of the model of the interlocking. Each LSC is able to describe only a single computation (or simulation trace) of the model

This format, similar to the message sequence diagram (UML) permit to specify a test scenarios in a friendly format, useful for visual inspections.

The LSC format can be exported as a text file as well.

Data format and data access offered by the Statemate tool.

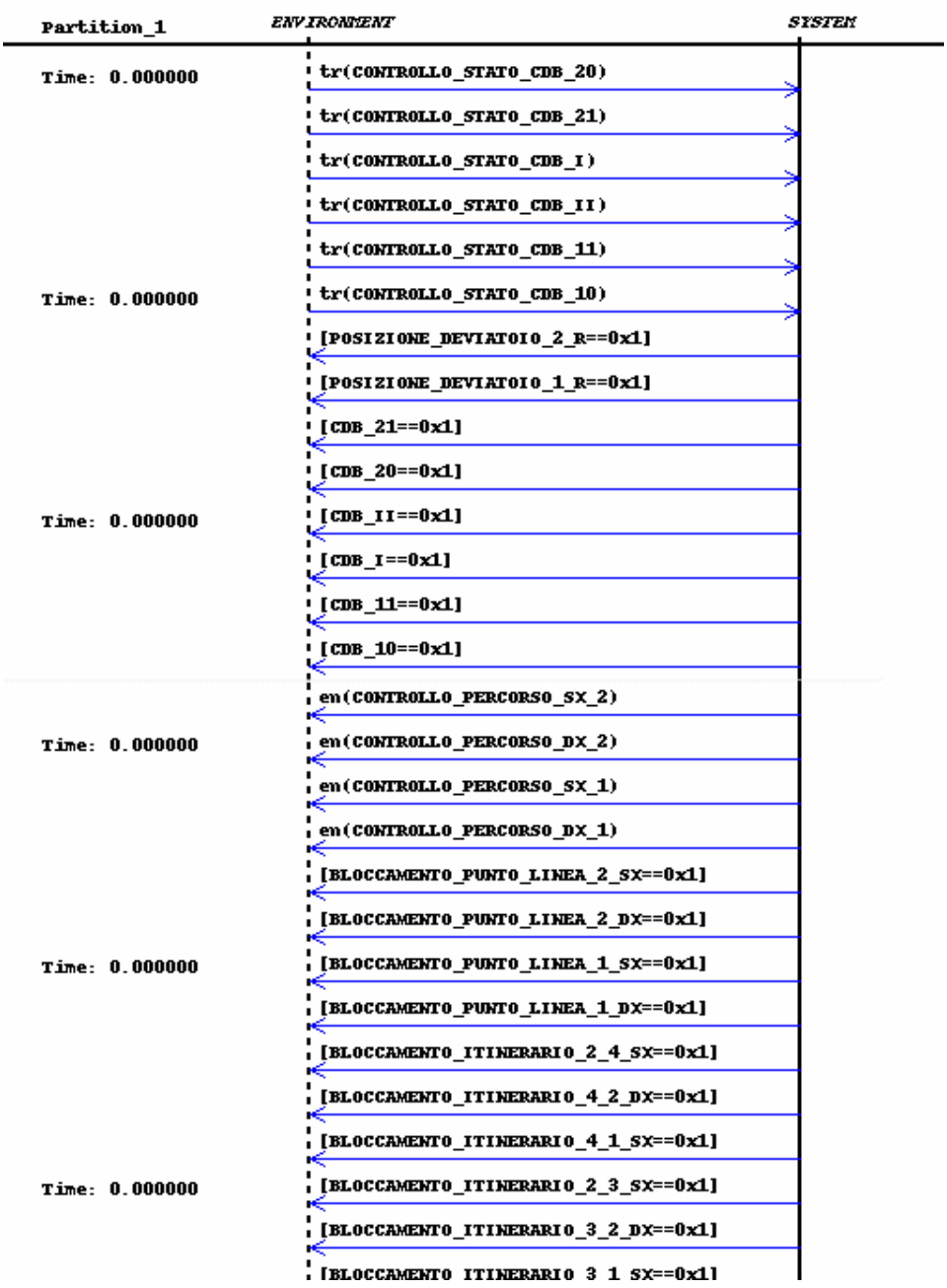

Figure 21 – A Live sequenze Chart

Data format and data access offered by the Statemate tool.

5.3 Dataport and Data Import

The Statemate Dataport Library contains functions that allow to extract information from the Statemate database from within an external program.

- It allows to communicate with the Statemate tool
- We can get any information from the internal database of Statemate tool
- We can use Dataport functions within our programs
- Dataport library:
    - Functions to perform a wide variety of database extraction operations
    - Using the library's function, we can extract from the specification database the information pertaining to an element.
    - We'll be able to use these informations to translate a model to an XMI specification

The Data Import consists of functions that allow to import information from another resource to the Statemate database. The import functions have a C language interface.

Call a series of creation functions to build the desired model.

As a result, the given workarea will be loaded with the new created model. No further operation is required. (i.e. no need to manually load charts into the workarea).

Data format and data access offered by the Statemate tool.

6 Translation from statecharts to boolean equations

During the collaboration a translation from generic charts to boolean equations has been studied.

By using this translation methodology, developed by ALSTOM, a generic chart can be translated in boolean equations describing the behaviour of the statecharts. The methodology consist first of performing a transformation which recalls the classical transformation from Mealy to Moore state machines.

6.1 Classical transformation from Mealy to Moore Machine

In a Mealy machine arcs (state transitions) "carry" output symbols. In a Mealy machine any individual state can have incoming transitions, which carry different output symbols.
Otherwise in a Moore machine any Moore state is not allowed to contain more than one output symbol. When you enter a Moore state, it must have one symbol that it will output.

The methodology of converting Mealy states to Moore states amounts to:

- duplicate states which have incoming arcs that carry different output symbols;
- create one clone for every distinct output symbol carried by all arc entering each Mealy state.

For each Mealy state, for each incoming arc which has a distinct output symbol make a duplicate state (duplicating all outgoing arcs) and connect all incoming arcs (which carry this distinct output symbol) to the newly created cloned state, then transfer the Mealy output symbol to the cloned Moore state (Figure 22).

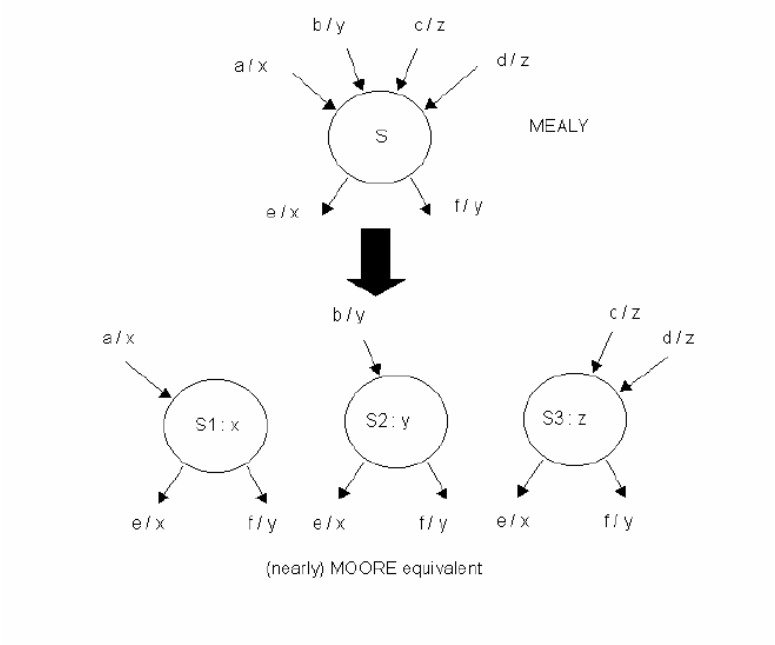

Figure 22 – Mealy state to Moore state transformation

Loops i.e. state transitions, which loop back to the same state introduce a problem. Loops have to be handled by treating each loop as a combination of (Figure 23):

- outgoing arc **AND**
- incoming arc

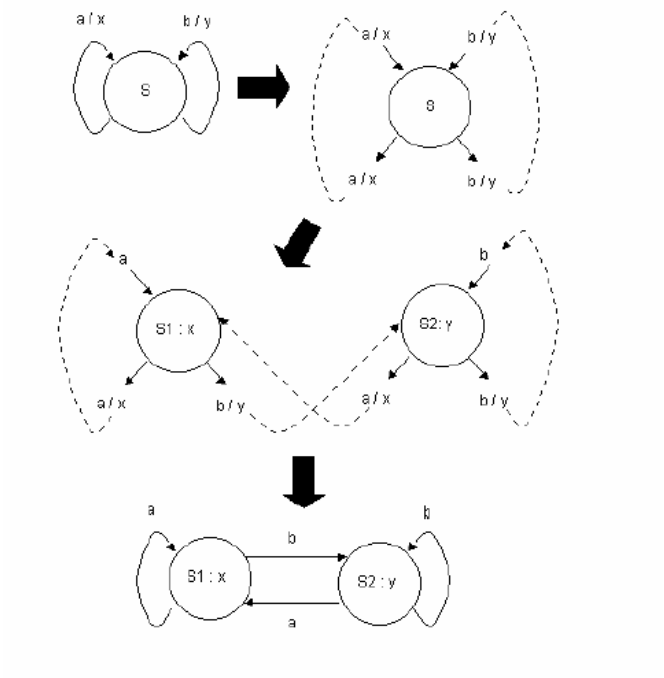

Figure 23 – loop transformation

Translation from statecharts to boolean equations

6.1.1 Example: Mealy to Moore Transformation

Mealy description of a simple FSM (Figure 24):

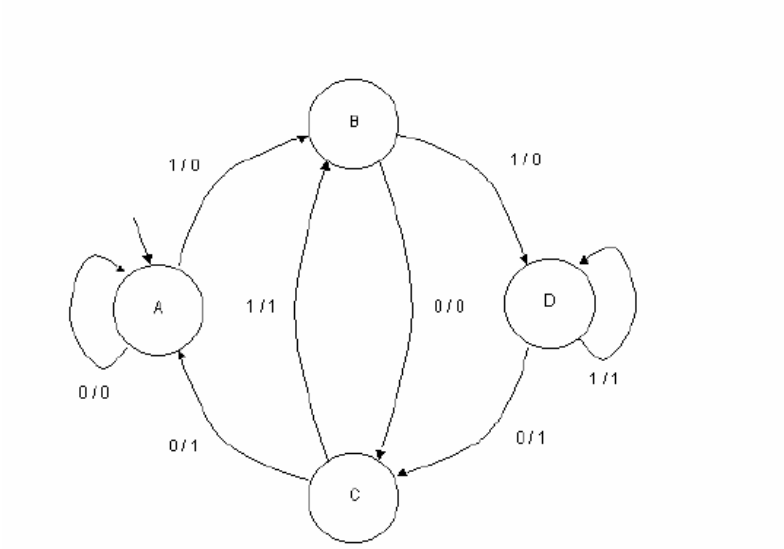

Figure 24 – A simple Mealy machine

The process described above produce the following (Figure 25) machine:

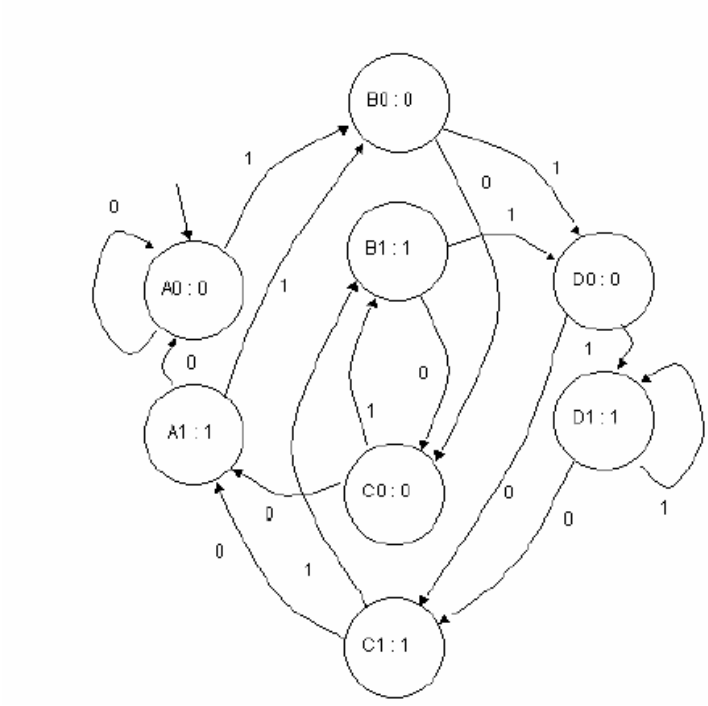

Figure 25 – The final Moore machine

6.2 The Translation

The first step to perform the translation from a stetechart to a set of boolean equation consists in converting the original statechart (Figure 26) into a chart similar to a Moore FSM (see Figure 26).

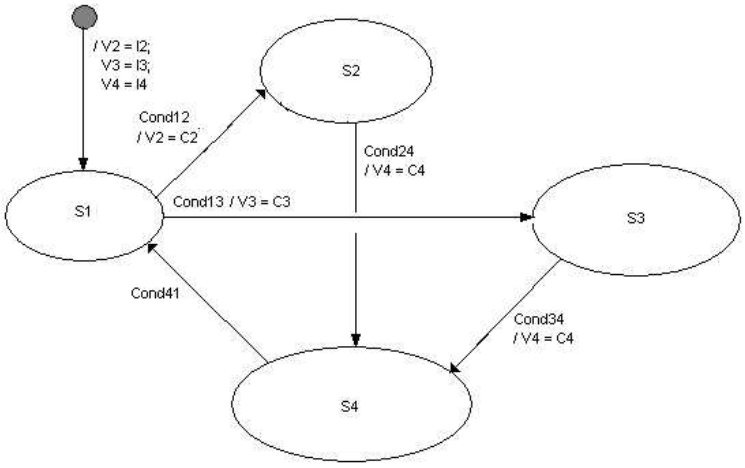

Figure 26 – the original statechart

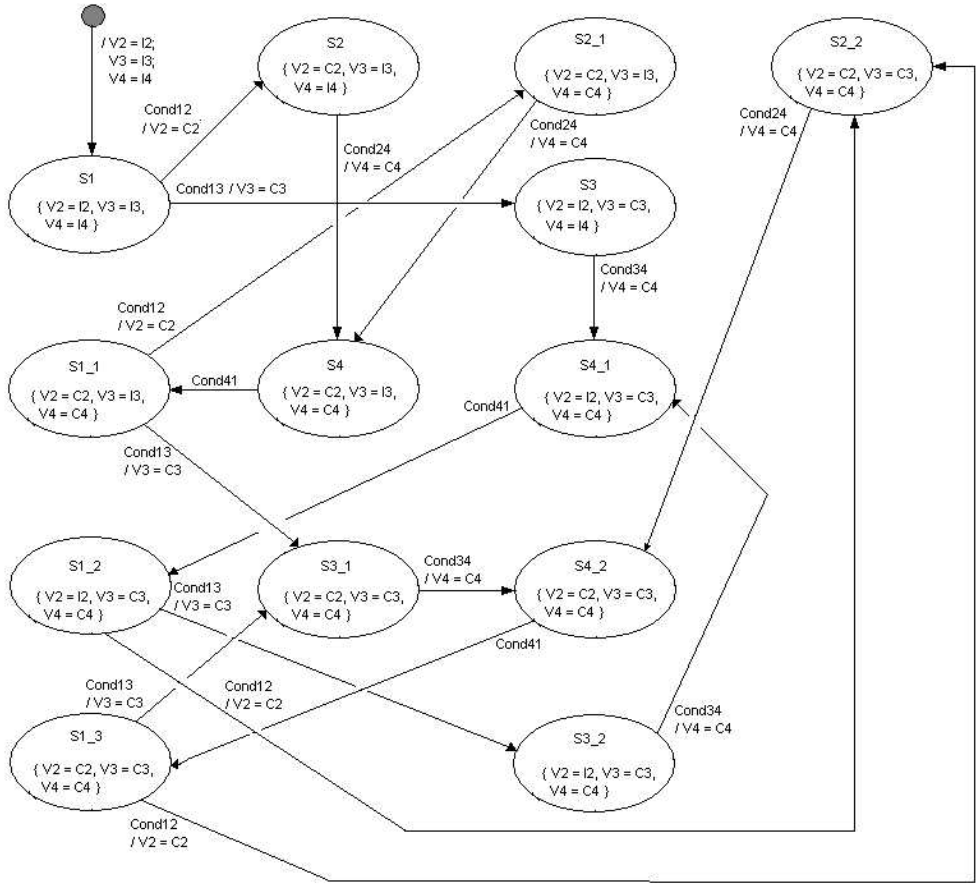

Figure 27 – The “Moore-like” chart

After this conversion a coding in boolean equation is possible. All the types of condition used in statechart have been considered to perform the correct translation.

We have also studied the translation of some special conditions used in the statecharts model, as:

- Delayed transition (Figure 28)

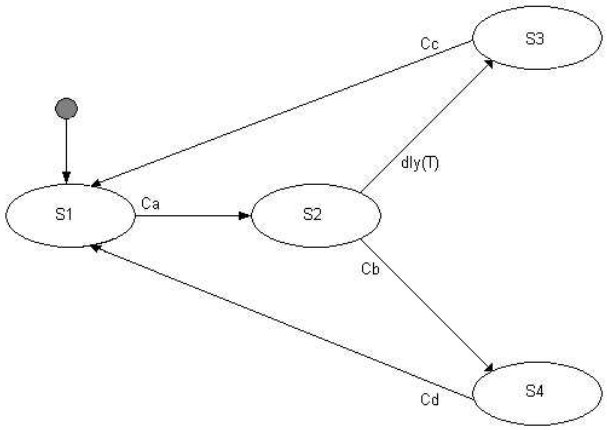

Figure 28 – delayed transition

- The *timeout event (*Explicit timing information inside a statechart). The form is **tm(E,T)**, where **E** is an event and **T** is an integer expression. This expression defines a new event, which will occur **T** time units after the latest occurrence of the event **E** (Figure 29).

After entering in the state S2 a timer is started, and after T time units perform the transition from S3 to S1.

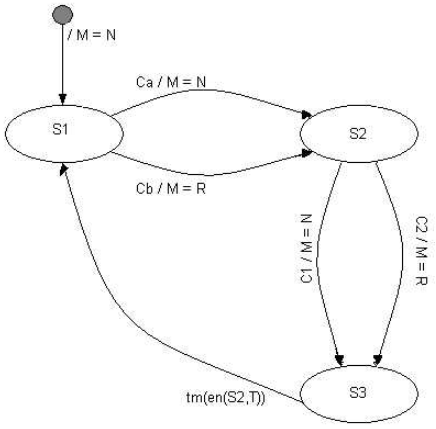

Figure 29 – chart with timeout event

- Testing the value of condition **C** is changed to true,.
- Testing the value of condition **C** is changed to false.
- Testing the value of **X** is changed, **x** is data-item or condition expression or array

For more information about the translation of the boolean equations, refer to the document "*SMARTLOCK – Creazione delle equazioni booleane da specifiche funzionali espresso in forma di diagrammi di stato*" [1].

7 Test scenarios

During the collaboration a test scenario analysis has been performed. Particularly, the possibility to generate test scenarios by using the Statemate tool has been analyzed. These scenarios can then be used to perform verification activities using Alstom proprietary verification tools.

7.1 Simulation test

The simulator tool allows to generate test scenarios during the simulation of the model.

There is the possibility to export these scenarios in different TXT file formats, like:

- Input/Output variables
- Simulation trace

The Statemate tool supplies also the possibility to generate automatic test vector using the ATG (Automatic Test Generation) tool.

7.1.1 Output by using API Library

During the collaboration a library has been done. This libray allows to generate an output text file similar to that already used in Alstom testing process (Figure 30).

```
Project Name: COMANDO_3_5_BIN
Work Area Directory: /home/rep1/banci/wa_comando_3_5_bin
Profile Name: OUTPUT_API_1
Date/Time Produced: Tue Feb  3 12:21:15 2004
Recording upon Output change
Recorded time mode: Absolute
Recording starts at simulation time: 0.000000

#Data Section:
C VERIFIER AP_IT_1 0
C VERIFIER AP_IT_2 0
C COMMANDER CDB_10 0
C COMMANDER CDB_11 0
C COMMANDER CDB_20 0
C COMMANDER CDB_21 0
C COMMANDER CDB_I 0
C COMMANDER CDB_II 0
C COMMANDER P_1_3_D_ARR 0
C COMMANDER P_3_2_D_PAR 0
C VERIFIER ROSSO_1CAT_I_ARR_1_D 0
C VERIFIER VERDE_II_AVV_1_D 0
C VERIFIER VERDE_I_ARR_1_D 0
C VERIFIER VERDE_I_AVV_1_D 0
C VERIFIER ROSSO_1CAT_I_ARR_1_D 1
C VERIFIER AP_IT_1 1
C VERIFIER ROSSO_1CAT_I_ARR_1_D 0
C VERIFIER AP_IT_2 1
C VERIFIER VERDE_I_AVV_1_D 1
C VERIFIER VERDE_I_ARR_1_D 1
C VERIFIER ROSSO_1CAT_I_ARR_1_D 1
C VERIFIER VERDE_I_AVV_1_D 0
C VERIFIER VERDE_I_ARR_1_D 0
C VERIFIER AP_IT_1 0
C VERIFIER AP_IT_2 0
```

Figure 30 – Testing output file

We have also investigated the symbolic simulation performed on generic charts.
For more information about creating the test scenarios, refer to the document *"Schemi di Principio espressi con Statemate - Scenari di Test"* [2]. This document analyzes the possibility to generate test scenarios by using the Statemate tool, extracting them from a model previously done that describes the interlocking system.

8 Conclusion, investigation areas and still open topics

The collaboration between ISTI-CNR and ALSTOM has addressed several important issues about Model-based design of Interlocking Systems, but several issues have not been investigated. In the following table a set of open issues that are proposed as possible activities for a further collaboration is presented. Particularly, for each activity the estimated required effort is given.

The 'Old Id' column represents the identifier number used for the open issues identified in the 'Progress Review Meeting Report' of the 23/1/04. With respect to this document, new open issues have been added, and some have been deleted either because they have been already addressed in the meanwhile, or because of established lack of interest or low priority.

Id	Old Id	Proposal	Description	Outcome	Effort estimation
1	**2**	*ADES rules converted in StateCharts*	It could be useful for documentation purpose, and to extend the range of available verification features.	Preliminary report	3 M/M
2	**4**	*Safety properties automatically generated from StateCharts*	Eventually by Theorem-proving.	Feasibility Report	6 M/M
3	**6**	*Safety properties automatically generated from boolean equations*	Eventually by Theorem-proving.	Feasibility Report	8 M/M
4	**7**	*Consistency*	Equivalence analysis between Boolean equation sets. Same test scenarios applied to different sets of logic.	Methodology for consistency check	5 M/M
5	**8**	*Non regression (i)*	If Statemate is able to supply a simulation coverage measure, where modification is made it would be possible to obtain the list of test scenarios which have to be re-executed, and how to modify the existing ones.	Feasibility Report	6 M/M

6	**9**	*Non regression (ii)*	Using model checking techniques, if a certain state is identified as “not covered” by simulation, the property (related to a certain test scenario) covering that state can be obtained. As a consequence, following a modification, the test scenarios assuring full test coverage can be obtained.	Feasibility report	5 M/M
7	***I***	*Non regression (iii)*	More complex is the generation of non regression tests when the system is specified by boolean equations. Two sets of boolean equations should be analyzed, to make a comparison and highlight the differences. After this analysis is simpler to make only the new test scenarios that stimulate the part of the system not tested.	Feasibility report	9 M/M
8	**10**	*Model checking on generic StateCharts*	This issue regards the use of model checking (depending on availability and costs, both the Statemate model checker and other model checkers which will be demonstrated compatible) to prove properties of the instantiated Statecharts. Model checking of non-instantiated statecharts is also studied.	Experience report	6 M/M
9	***I***	*Embedded Statecharts interpreter*	Formalizing the subset of statecharts language. Make an application which uses the given formal semantic to interpret Statecharts.	Prototype of the interpreter	15 M/M
10	***I***	*Tool that allows to instantiate generic statecharts automatically*	Generating an instantiated model by using generic charts and control table data.	Prototype of the instantiator	6 M/M

11	*I*	*LSC describing test scenarios*	Developing and defining generic test scenario by using LSC (Live Sequence Chart). Similar to the use of generic charts, to generate some generic LSC for tests documentation.	Definition of the process	3 M/M
12	*I*	*Equivalence between boolean equations and statecharts*	Demonstration of the equivalence. Two ways are possible: analytical equivalence verification vs. consistency in the same test scenarios.	Study of the two alternatives and definition of possible tool support	8 M/M
13	*I*	*Correctness of generic rules from instantiated boolean equations*	Beginning from a set of instantiated stations that use instantiated boolean equations, verify the correctness of generic rules generating the boolean equations. From testing performed on instantiated stations is possible, by using meta-testing, to infer the correctness of the generating rules.	Feasibility study	6 M/M