PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 42, NO. 1, PP. 135-146 (1998)

THE ISM: A FORMAL TOOL FOR MODELLING AND VERIFICATION

Janine MAGNIER, Mireille LARNAC and Vincent CHAPURLAT

LGI2P
EMA/EERIE
Parc Scientifique Georges Besse
30000 - Nimes, France
email: {magnier, larnac, chapurla}@eerie.fr
Phone: +33 0 46 638 7020 - Fax: +33 0 46 638 7074

Received: December 10, 1997

Abstract

This paper addresses the issue of modelling and analysis of systems. The necessity of carrying out verification and/or validation tasks is discussed, the factors which influence the choice of a model are shown, and the differences between simulation tools or formal methods is explained. Within the framework of discrete time modelling of systems, a method for formally analysing the behaviour of a system described by a Finite State Machine permits to prove properties; the method is based on the translation into a formal system which has got a temporal logic interpretation, and the analysis of the sensitivity of the temporal evolution of the system with respect to some events involves the use of the Temporal Boolean Difference. Furthermore, in order to improve the expressiveness power of the model, an extension of the Finite State Machine model, called Interpreted Sequential Machine (ISM) supports the representation of complex data. The verification process has been adapted to this model.

Keywords: discrete time state modelling, verification and validation, temporal logic, Temporal Boolean Difference, Interpreted Sequential Machine.

1. Introduction

Managing the complexity of systems requires to be able to analyse their behaviour and evolution. Obviously, the method to be used for fulfilling this task depends on the existence of the system or if it is under development. In the first case, it can be possible to run real tests on the system itself. But in general, should the system be achieved or not, the only possibilities to better understand it, consist in establishing a model and then to perform some analysis. The necessity of verification and validation, as well as the criteria for the choice of a representation model, on the modelling point of view and abstraction level, and of the verification or validation methods are discussed in the first part of this paper. The formal verification method defined for the Finite State Machine model is then described; it is based on the translation of the behaviour of the model into a temporal logic formalism,

and the proof of properties of the system is proven by involving formal tools, and especially the Temporal Boolean Difference. The third part presents the application of this modelling and verification method to the Interpreted Sequential Machine, which constitutes an extension of the FSM model by taking complex data into account.

2. Models, Verification and Validation

Due to the increasing complexity of artificial systems, should they be industrial, environmental, integrated circuits, and so on, it is now well-known that it is necessary, in order to reduce design or exploitation costs and risks, to detect design errors early or to understand how dysfunction can occur. Let us first remind with the definitions of the two main terms used when mentioning analysis domain: verification and validation. Considering the ISO-8402 norm which defines the quality vocabulary, *verification* is the *confirmation by examination and provision of objective evidence that specified requirements have been fulfilled*, since *validation* is the *confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled; objective evidence is the information which can be proved true, based on facts obtained through observation, measurements, test or other means.* In other words, verification answers the question 'does the system really implement what it has been designed for?', and validation must evaluate 'is the system the one the user expected?'. It follows that within design and development phases, validation concerns the process of examining a product to determine its conformity with user needs, and then multiple validations may be carried out if there are different intended uses. In what follows, the term 'verification' will refer to the process of performing analysis, should the goal be verification or validation.

In order to understand and analyse the behaviour of a real system, it is often necessary to use a representation model so that some verification is performed. However, if the system is complex, it can be impossible for a human user to exactly and completely describe it into any formalism. Moreover, it happens that a black-box approach for modelling is sufficient, and that only an input/output relationship is suitable for the analysis of accurate properties of the real system. In other words, there exists no general rules for determining what a 'good' modelling of a real system is; this depends first on the possibility to represent some of its aspects into detailed models or through global approaches, and second on the kind of analysis one intends to perform.

A global approach for modelling a complex system consists in describing the outputs which are obtained when inputs are applied, without expressing the transition function between inputs and outputs, and without giving the inner structure of the system (the knowledge of which can be in-

complete or even totally unknown). The class of models thus obtained, like for instance Neural Networks, is usually qualified as 'adaptative' or 'auto-organising', since the inner parameters of the model auto-evaluate through an iterative learning phase.

The verification of a description of a system based on such models most of the time consists in applying new input sequences and verifying that the produced outputs are those which were expected. This kind of verification is thus based on simulation.

An internal model of a system is based on the description of one or several points of view (structural, behavioural, data flows, geographical, etc.) of the real system into a given formalism. Nevertheless, whatever model is chosen, the description will always be oriented towards the internal aspects of the model.

As far as the model is based on a formal system and is associated a clear semantics, it must be possible to carry out some verification. Two methods can be envisaged: simulation or formal verification.

Simulation is the most widely spread method; it consists in applying values on the inputs, executing the model and examining the values of the outputs. The key point here is that the verification method is not simulation itself, but consists in the analysis and the interpretation of the results of the simulation. These tasks are usually carried out manually (i.e. the user looks at the simulation results and compares them with what he expected), but can also be based on some formal methods and tools.

Formal verification gathers all the methods which are not based on execution of the model, but on an expression of the model thanks to a formal system and on the bringing into play of formal reasoning methods. This can be a means for analysing the behaviour of the system, but in addition to that, these methods may provide some tools to extract some inherent properties which are contained in the system but which do not explicitly appear in its description.

Finally, simulation is very useful if the user wants to carry out a partial verification for a few particular cases. The main problem is to generate (if possible) the accurate input vectors which correspond to the property of the system one intends to test. Furthermore, classical simulation is not really applicable when data (inputs) vary on an interval with an infinite number of values. On the other hand, a major advantage of this approach is that several industrial, efficient and user-friendly tools exist on the market for simulating any kind of system.

The purpose of the work presented in the following is to show what and how to formally verify modelling approach based on Finite State Model. The first step concerns the classical Finite State Machine model; then the method has been extended to the Interpreted Sequential Machine whose goal is to overcome the main expressiveness limitations of the FSM.

3. Verification of the Finite State Machine Model

DEFINITION 1 The Finite State Machine (FSM) model is classically defined by the 5-tuple (HARTMANIS – STEARNS, 1966; KOHAVI, 1978):

$$M =< S, I, O, \delta, \lambda >$$

where: S is a finite, non empty set of states
I is a finite, non empty set of inputs
O is a finite set of outputs
δ is the transition (next state) function: $\delta : I \times S \rightarrow S$
λ is the transition output function: $\lambda : I \times S \rightarrow O$

We denote by $\#S$, $\#I$ and $\#O$ the cardinalities of S, I and O respectively. We consider only deterministic machines, which obey the following rules:

- each state has one and only one following state for each relevant input
- no two distinct inputs can be applied simultaneously
- no two distinct outputs can be output simultaneously

Also, we consider machines which are completely specified. This means that for all states, the next state and output are specified for all inputs. It is possible to study machines which are incompletely specified within the framework of our model, by defining a new type of variable for representing the unspecified inputs or outputs. However, we concentrate on completely specified machines in this article.

Expression in Temporal Logic: We define the following: Three sets $\mathbf{S}$, $\mathbf{X}$ and $\mathbf{Z}$ each representing propositions of the same type.

- $\mathbf{S}$ is the set of state type propositions:

  $\mathbf{s}_i \in \mathbf{S}, \forall\ s_i \in S,\ i = 0, \ldots, \#S - 1$, $\mathbf{s}_i$ is TRUE when the state of M is s_i.

- $\mathbf{X}$ is the set of input type propositions:

  $\mathbf{x}_j \in \mathbf{X}, \forall\ i_j \in I,\ j = 0, \ldots, \#I - 1$, $\mathbf{x}_j$ is TRUE when the present input of M is i_j.

- $\mathbf{Z}$ is the set of output type propositions:

  $\mathbf{z}_k \in \mathbf{Z}, \forall\ o_k \in O,\ k = 0, \ldots, \#O - 1$, $\mathbf{z}_k$ is TRUE when the present output of M is o_k.

These definitions allow us to describe the temporal evolution of M (by expressing the behaviour of the transitions of M) into the following notation (based on the DUX temporal logic (GABBAY et al, 1980; MANNA – PNUELI, 1982)), called Elementary Valid Formula (EVF):

Let us suppose that we have $\delta(s_i, i_j) = s_k$ and $\lambda(s_i, i_j) = o_l$; it follows

$$\mathrm{EVF} ::= \Box(\mathbf{s}_i \wedge \mathbf{x}_j \supset \bigcirc \mathbf{s}_k \wedge \mathbf{z}_l)$$

whose interpretation is the following: 'it is always true ($\Box$ operator) that if s_i is the current state (and therefore $\mathbf{s}_i$ is true) and i_j is the current input ($\mathbf{x}_j$ is true), then the next state ($\bigcirc$ operator) will be s_k ($\mathbf{s}_k$ will be true), and the current output is o_l ($\mathbf{z}_l$ becomes true)'.

It follows that the set of all the EVF's (each of which expresses the existence of a transition of the FSM) provides an equivalent representation of the behaviour of the FSM model. This statement is true if we also take into account the set of formulae which represent the determinism constraints. The first set contains the state determinism concept, which says that at a given time step, there is one and only one current state. This determinism formula can be written, using the DUX formalism:

$$\mathrm{DF1} ::= \Box[\mathbf{s}_i \supset \neg \mathbf{s}_j] \forall\, \mathbf{j} \neq \mathbf{i}, \mathbf{i}, \mathbf{j} \in \{0, \ldots, \#\mathbf{S} - 1\}.$$

Similarly, DF2 and DF3 express that at a given time step, the machine cannot have two different inputs, and cannot produce two different outputs:

$$\mathrm{DF2} ::= \Box[\mathbf{x}_i \supset \neg \mathbf{x}_j] \forall\, \mathbf{j} \neq \mathbf{i}, \mathbf{i}, \mathbf{j} \in \{0, \ldots, \#\mathbf{X} - 1\},$$

$$\mathrm{DF3} ::= \Box[\mathbf{z}_i \supset \neg \mathbf{z}_j] \forall\, \mathbf{j} \neq \mathbf{i}, \mathbf{i}, \mathbf{j} \in \{0, \ldots, \#\mathbf{Z} - 1\}.$$

Within the framework of a verification process, it is often necessary to consider time intervals. This leads to the definition of a state, input and output sequences, which are noted, respectively:

$$\mathbf{s}_i^n ::= \mathbf{s}_{i1} \wedge \bigcirc \mathbf{s}_{i2} \wedge \bigcirc^2 \mathbf{s}_{i3} \wedge \ldots \wedge \bigcirc^{n-1} \mathbf{s}_{in},$$

$$\mathbf{x}_j^n ::= \mathbf{x}_{j1} \wedge \bigcirc \mathbf{x}_{j2} \wedge \bigcirc^2 \mathbf{x}_{j3} \wedge \ldots \wedge \bigcirc^{n-1} \mathbf{x}'_{jn}$$

$$\mathbf{z}_k^n ::= \mathbf{z}_{k1} \wedge \bigcirc \mathbf{z}_{k2} \wedge \bigcirc^2 \mathbf{z}_{k3} \wedge \ldots \wedge \bigcirc^{n-1} \mathbf{z}_{kn}.$$

Then, in order to provide the user with a more global view of the system evolution, first the concept of Temporal Event (Et) which represents the possible effects of the machine functioning has been defined, and then all the conditions which lead to obtaining a given temporal event are gathered into one single formula, called Unified Valid Formula (UVF). A temporal event will be a future state (Et= $\bigcirc \mathbf{s}_i$), a future state within n time steps (Et= $\bigcirc^n \mathbf{s}_i$), a state sequence (Et= $\mathbf{s}_i^n$), a present output (Et= $\mathbf{z}_k$), a n-future output (Et= $\bigcirc^n \mathbf{z}_k$), or an output sequence (Et= $\mathbf{z}_k^n$). The Unified Valid Formula associated with a temporal event Et is thus:

$$\mathrm{UVF(Et)} ::= \bigvee_{(p,q)/\mathbf{s}_p \wedge \mathbf{x}_q^n \supset \mathrm{Et}} \left(\mathbf{s}_p \wedge \mathbf{x}_q^n\right)$$

It appears that the calculation of an UVF only consists in manipulating the set of EVF's.

Verification of properties: Let us come back to the goal of this work; it addresses the formal verification of properties of systems which are represented thanks to a FSM model. The first question to answer is: 'what kind of properties can be proven?'. To start with, the structure of the FSM can be exploited, and properties of some states can be of great interest. For instance, it is worth knowing if two states are equivalent, or if a state is a source or a sink. More sophisticated properties consist in establishing the conditions (on input sequences) to make a state being a 'functional' sink, even though it is not a structural one, or to generate input sequences to resynchronise the machine into a given state.

In most cases, the state evolution of the machine is not available, and the only means for the user to get some information on the machine evolution is to examine the outputs. So the analysis process of output sequences is very important, and a tool for generating input sequences in order to obtain outputs, or to distinguish internal states must be provided.

Last, it seems very important to be able to formally establish the influence of a current factor (input or state) on the future evolution. This relates to the 'sensitivity' of the future with respect to a present situation or decision.

The verification method is based on two approaches:

- in some cases (for some properties), it is sufficient to analyse the EVF's and UVF's (either search if a given formula exists, or what its form is). For example, if s_p is a sink state, it means that all the transitions which leave s_p go back to this state. It follows that all the EVF's which contain s_p in their left part must be of the form: $\mathrm{EVF}::= \Box(\mathbf{s}_p \wedge \mathbf{x}_j \supset \bigcirc \mathbf{s}_p \wedge \mathbf{z}_l)$, for all $\mathbf{x}_j$. Similarly, if s_p is a source state, it means that if there exist some transitions whose destination is s_p, they come from s_p. The consequence is that either $\mathrm{UVF}(\bigcirc \mathbf{s}_p)$ is empty, or that it has the following form: $\mathrm{UVF}(\bigcirc \mathbf{s}_p) = \vee (\mathbf{s}_p \wedge \mathbf{x}_j)$.
- Unfortunately, this approach of verification based on the study of EVF's and UVF's is not sufficient for analysing the influence of the present on the future. This is the reason why a formal tool for analysing the sensitivity has been defined: the Temporal Boolean Difference.

The Temporal Boolean Difference: The Temporal Boolean Difference (TBD) is the extension of the classical Boolean Difference (KOHAVI, 1978) defined on propositional logic, to temporal logic, especially the DUX system.

The formal definition of TBD of $f(x_1, \ldots, x_n)$ with respect to a variable x_i is the following:

$$\frac{\partial f(x_1, \ldots, x_n)}{\partial x_i} = f(x_1, \ldots, x_i = \text{False}, \ldots, x_n) \oplus f(x_1, \ldots, x_i = \text{True}, \ldots, x_n).$$

The important point is that there is a strong analogy with the Boolean Difference of Boolean functions, but the fundamental differences are that the formulae are expressed in Temporal Logic, and that the variables which are manipulated are typed (states, inputs, outputs) and non independent (because of the determinism properties).

For the verification of properties of a FSM model, the TBD is applied on an UVF for obtaining a Temporal Event which concerns future states or outputs (cf. the definition of Temporal Event), with respect either to a current state or to an input. The formula is called the Derived Valid Formula (DVF) and is the following:

$$\text{DVF}(\text{Et}, q) = \frac{\partial \text{UVF}(\text{Et})}{\partial q} = \text{UVF}\left(\text{Et}_{|q}\right) \oplus \text{UVF}\left(\text{Et}_{|\neq q}\right).$$

The result of the calculation of DVF(Et, q) can be:

- False. This means that UVF(Et) is independent of q; in other words, the fact that q changes value has no influence on the fact that Et will occur or not.
- not False. In this case, we obtain a Temporal Logic formula which expresses the sensitivity of UVF(Et) to changes in q, i.e. the conditions for UVF(Et) to pass from True to False (or conversely) when q changes value.

The applications of TBD are various. It permits to generate input sequences for resynchronising the machine into a given 'initial' state, or for distinguishing the inner states (which are unknown) through the generation of distinct output sequences. Moreover, a very wide field of application is the study of the impact of a decision (or a current event) on the future evolution of the system. Further, even though the user has got no possibility to change the present, he knows all the conditions which, when made True, make the system evolve into a given way. It is then up to him to choose his strategy for modifying some parameters and then determine what he wants to get into the future.

In conclusion, we have defined a formal method for providing the user with an equivalent symbolic representation of the behaviour of the Finite State Machine model, and then with a tool which supports proof of properties and formal analysis. In order to do this, it has been necessary to define the concept of Temporal Boolean Difference for evaluating the sensitivity of the evolution of a system with respect to some variable change. The details

of the modelling and verification approach, as well as the demonstrations of all the theorems can be found in (MAGNIER, 1990).

The limitations of this approach are linked to the weak expressiveness of the FSM model, which only handles Boolean data, and which needs to express any data influencing the system through inputs or states. It follows that the number of states or transitions tends to increase exponentially as soon as new data have to be taken into account. This is the reason why we have defined an extension of the FSM, called the Interpreted Sequential Machine model.

4. Extension to the Interpreted Sequential Machine

The Interpreted Sequential Machine (ISM) model is a behavioural model which extends the expressiveness power of the FSM model by associating with a Control Part (the core of which is a sequential machine), a Data Part in which data and operations are represented. The underlying concepts of the ISM were adapted from the Extended Finite State Machine (EFSM) defined by CHENG – KRISHNAKUMAR (1993) for functional testing purpose.

Definition of the ISM model: The main characteristics of this model are the following:

- the core of the model is a state system (extension of state machine)
- the inputs and outputs can be of any type (Boolean, integer, real, data, event, etc.)
- the data which constitute the environment of the system and which influence the functioning of the sequential part of the system are represented separately

As any sequential model, the state diagram, called Control Graph, is composed of an alternation of states and transitions. Informally, a transition between two states is activated if first an event appears on the inputs, and then if some conditions on the environment (data) are fulfilled; then the effects of this transition firing appear on the outputs and on the data of the environment which change value.

Structurally, the ISM model owns inputs and outputs, and is made up of two parts (*Fig. 1*):

- the Control Part (CP) contains the Control Graph and some necessary interpreters
- the Data Part (DP) is made up of the set of data which represent the environment of the system, and of the operations on these data

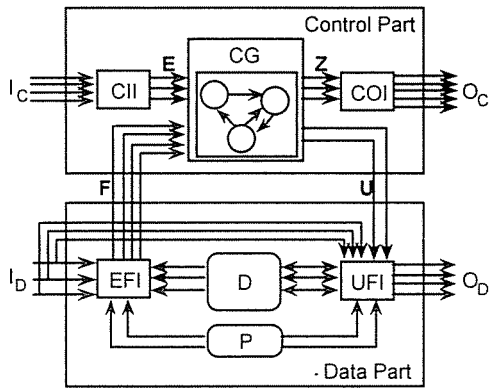

Fig. 1. Structure of the ISM model

It is possible to partition the set I of inputs into two disjoint sets: the set I_C of the Control Part inputs and the set I_D of the Data Part inputs. Similarly, the set O of outputs is split up into Control Part outputs (O_C) and Data Part outputs (O_D).

The formal model of the ISM is thus the following:
ISM $=< I, O, CP, DP >$ with:

- $CP =< I_C, CII, \mathbf{E}, \mathbf{F}, CG, \mathbf{U}, COI, \mathbf{Z}, O_C >$ where:
  - I_C is the set of the Control Part inputs
  - $\mathbf{E}$ is the set of propositional input variables of the Control Graph
  - CII is the Control Input Interpreter. Its role is to evaluate some conditions on the Control Part inputs of I_C, and therefore to give some propositional (Boolean) values to the elements of $\mathbf{E}$
  - $\mathbf{F}$ is the set of propositional enabling variables
  - CG is the Control Graph: $CG =< \mathbf{S}, \mathbf{E}, \mathbf{F}, \mathbf{Z}, \mathbf{U}, \delta, \lambda, \beta >$ where:
    - $\mathbf{S}$ is the set of propositional symbolic state variables
    - δ is the propositional transition function (next state): $\delta : \mathbf{S} \times \mathbf{E} \times \mathbf{F} \rightarrow S$
    - λ is the propositional output function: $\lambda : \mathbf{S} \times \mathbf{E} \times \mathbf{F} \rightarrow \mathbf{Z}$
    - β is the propositional updating function: $\beta : \mathbf{S} \times \mathbf{E} \times \mathbf{F} \rightarrow \mathbf{U}$

    A transition t of CG is defined as follows: $t : (\mathbf{s}_i, \mathbf{e}_j, \mathbf{f}_k) \rightarrow (\mathbf{s}_l, \mathbf{z}_m, \mathbf{u}_n)$ where:
    $$(\mathbf{s}_i, \mathbf{s}_l) \in \mathbf{S}^2, \quad \mathbf{e}_j \in \mathbf{E}, \quad \mathbf{f}_k \in \mathbf{F}, \quad \mathbf{z}_m \in \mathbf{Z}, \quad \mathbf{u}_n \in \mathbf{U}$$
    $$\mathbf{s}_l \in \delta(\mathbf{s}_i, \mathbf{e}_j, \mathbf{f}_k), \quad \mathbf{z}_m \in \lambda(\mathbf{s}_i, \mathbf{e}_j, \mathbf{f}_k), \quad \mathbf{u}_n \in \beta(\mathbf{s}_i, \mathbf{e}_j, \mathbf{f}_k)$$
  - $\mathbf{U}$ is the set of propositional updating variables (assigned when β is evaluated)
  - $\mathbf{Z}$ is the set of propositional output variables of the Control Graph (assigned when λ is evaluated)
  - COI is the Control Output Interpreter. Its role is to give values to the Control Part outputs from the values of the propositional

variables of the variables of $\mathbf{Z}$.
  - O_C is the set of the Control Part outputs
- $DP =< I_D, D, P, EFI, UFI, \mathbf{F}, \mathbf{U}, O_D >$ where:
  - I_D is the set of the Data Part inputs
  - D is the set of internal variables
  - P is the set of parameters (fixed characteristics of the system)
  - EFI is the Enabling Function Interpreter. Its role is to evaluate some conditions on the Data Part inputs, on the internal variables and on the parameters and therefore to give some propositional (Boolean) values to the elements of $\mathbf{F}$
  - $\mathbf{F}$ is the set of propositional enabling variables
  - $\mathbf{U}$ is the set of propositional updating variables
  - UFI is the Updating Function Interpreter. It contains all the functions that are used for calculating the new values of the Data Part outputs and of the internal variables of D. These updating functions are activated by the variables of $\mathbf{U}$
  - O_D is the set of the Data Part outputs

The behaviour of an ISM model is described by the dynamic evolution both of the Control Graph, and of the internal variables of D..

The behaviour of CG is expressed by the sequence of fired transitions.

The interpretation of a transition t (*Fig. 2*), where $t : (\mathbf{s}_i, \mathbf{e}_j, \mathbf{f}_k) \rightarrow (\mathbf{s}_l, \mathbf{z}_m, \mathbf{u}_n)$ is the following: if $\mathbf{s}_i$ is true ($\mathbf{s}_i$ denotes the current state of the Control Graph), then if $\mathbf{e}_j$ is true (the value of the inputs of CP verifies the conditions which make $\mathbf{e}_j$ be true), then, if $\mathbf{f}_k$ is true (the value of the inputs of DP, of the internal variables in D and of the parameters of P verifies the conditions which make $\mathbf{f}_k$ be true), then:

- at the same time, $\mathbf{z}_m$ and $\mathbf{u}_n$ become true until the next transition firing on CG,
  - the functions associated with $\mathbf{z}_m$ in COI update the Control Part outputs
  - the functions associated with $\mathbf{u}_n$ in UFI update both the internal variables and the Data Part outputs
- the next state will be $\mathbf{s}_l$ ($\mathbf{s}_l$ will be true, while $\mathbf{s}_i$ will become false).

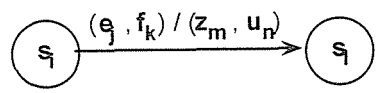

Fig. 2. A transition of CG

Verification: Given an ISM model of a real system, it is possible to carry out formal verification by proof of properties. The process described above for the verification of FSM models has been adapted and extended to

the ISM (LARNAC et al, 1997). At present, only the first step of verifying the behaviour of the Control Graph has been fulfilled. Of course, numbers of properties which were verified on the FSM (e.g. independence and exclusivity of the inputs) are no longer verified, and it has been necessary to state new hypotheses for expressing a realistic view of determinism within this model. Roughly speaking, it consists in no longer considering the only inputs, but the couples constituted by the propositional input variables and the propositional enabling variables (VANDERMEULEN, 1996). Furthermore, the properties which are verified on the Control Graph appear to be only sufficient at the ISM level.

5. Conclusion and Perspectives

A formal method for the verification by proof of properties of discrete finite state systems has been defined, first on the Finite State Machine model, and then on an extension called Interpreted Sequential Machine. The advantages of this approach concern a better understanding of the temporal behaviour and evolution of systems, by highlighting the influence of the present on the future. The result here is more complete than with simulation, since it is possible to obtain all the conditions which make a future event being sensitive to the present or to the past.

Finally, in order to take into account the complexity of systems in terms of the data which interact and which must be manipulated, the ISM model has been defined, and the verification process has been extended. It is now necessary, first to take into account the non independence of data states from one time step to another, and second to widen the model concepts to a more precise notion of time. Indeed, for the moment, time is only viewed as a succession of steps. For some real applications, it may be necessary to insert some 'real time' features, as for example duration on transition firings, or delays in data updatings. In parallel, the interconnection and decomposition mechanisms must be described, and the verification process studied.

References

[1] CHENG, K. T. – KRISHNAKUMAR, A. S. (1993); Automatic Functional Test Generation using the Extended Finite State Machine Model; *30th ACM/IEEE Design Automation Conference.*

[2] GABBAY, D. – PNUELI, A. – SHELAH, S. – STAVI, J. (1980); On the temporal analysis of fairness; *7th ACM Symposium on Principles of Programming Languages.*

[3] HARTMANIS, J. – STEARNS, R. E. (1966); Algebraic Structure Theory of Sequential Machines; Prentice Hall, Englewood Cliffs, N.J.

[4] KOHAVI, Z. (1978); Switching and Finite Automata Theory; Tata McGraw Hill, Computer Science Series.

[5] LARNAC, M. – CHAPURLAT, V. – MAGNIER, J. – CHENOT, B. (1997); Formal Representation and Proof of the Interpreted Sequential Machine; *Eurocast'97*, LNCS, Springer Verlag, to appear.
[6] MAGNIER, J. (1990); Représentation symbolique et vérification formelle de machines séquentielles, Thèse d'Etat, University of Montpellier II, France.
[7] MANNA, Z. – PNUELI, A. (1982); How to cook a temporal proof system for your pet language; report No. STAN-CS-82-954, Depart. of Computer Science, Stanford University.
[8] VANDERMEULEN, E. (1996); La Machine Séquentielle Interprétée: un modèle à états pour la représentation discrète et la vérification de systèmes; PhD Thesis, University of Montpellier II.