**UNIVERSITÀ DEGLI STUDI DI CAGLIARI**
Dipartimento di Ingegneria Elettrica ed Elettronica

**ÉCOLE NORMALE SUPÉRIEURE DE CACHAN**
Laboratoire Universitaire de Recherche en Production Automatisée

# Analysis of grafcet models by automatic generation of the equivalent timed automaton

Master thesis in Electronic Engineering

Tesi di Laurea Specialistica in Ingegneria Elettronica

**By**

**Cristiano Poddie**

Advisors:

Ass. Prof. Jean-Marc Roussel, LURPA – ENS Cachan

Prof. Alessandro Giua, DIEE – Università degli Studi di Cagliari

December 2009


**UNIVERSITÀ DEGLI STUDI DI CAGLIARI**

Dipartimento di Ingegneria Elettrica ed Elettronica

**ÉCOLE NORMALE SUPÉRIEURE DE CACHAN**

Laboratoire Universitaire de Recherche en Production Automatisée


# Analysis of grafcet models by automatic generation of the equivalent timed automaton

*Dedicato alla mia famiglia*

*Dedicated to my family*

## Acknowledgements

First, I would like to express my gratitude to Monsieur Jean-Marc ROUSSEL for his trust on me supporting my candidature for the ENS – Cachan International Scolarship and for his help during the research work and writing this thesis after my comeback in Italy. This work could not be possible without his wide willingness, his teachings, his suggestions and his precious encouragement.

This research work has been carried out within LURPA laboratory at the École Normale Supérieure de Cachan which financed me with an International Scholarship for a six months research training. LURPA is a very important university automated production research laboratory and it provides its students with highly qualified and experienced staff and first choice equipments and structures. I want to thank all LURPA staff and students, all International Relations Department staff and my neighbors' door at the PdJ that ensured me a very pleasant stay at the ENS – Cachan.

I would like to express my gratitude to Prof. Alessandro Giua for his trust on me, without his suggestions I could not attend the International Scholarship concourse, and without his teachings on discrete and hybrid systems I could not face this work.

I want to thank all DIEE department staff and colleagues and all my friends, especially who constantly supported me in the difficult moments.

I would like to give my special thanks to my family who always supported me in all these years.

## Ringraziamenti

Innanzitutto vorrei esprimere la mia piena gratitudine verso M. Jean-Marc Roussel, per avermi dato la sua fiducia sostenendo la mia candidature al concorso ENS – Cachan International Scolarship e per il suo aiuto sia durante il lavoro di ricerca sia durante la stesura della tesi dopo il mio rientro in Italia. La realizzazione di questo lavoro non sarebbe stata possibile senza la sua grande disponibilità, i suoi insegnamenti e suggerimenti e senza il suo prezioso incoraggiamento.

Il lavoro di ricerca trattato in questa tesi è stato realizzato all'interno del laboratorio LURPA, presso l'École Normale Supérieure de Cachan che mi ha finanziato con una borsa di studio per un semestre di tirocinio di ricerca. Il laboratorio LURPA è un importante laboratorio specializzato in produzione automatizzata che fornisce ai suoi studenti personale altamente qualificato, ed equipaggiamenti e strutture di primo livello. Voglio ringraziare tutto lo staff e gli studenti del LURPA, lo staff del Servizio Relazioni Internazionali e i miei vicini di stanza al PdJ per avermi permesso una piacevole permanenza all'ENS – Cachan.

Vorrei esprimere la mia gratitudine verso Prof. Alessandro Giua, per avermi dato la sua fiducia, senza il suo supporto e i suoi consigli non avrei potuto partecipare al concorso che mi ha permesso di effettuare questo tirocinio, e senza i suoi insegnamenti nel campo dei sistemi discreti ed ibridi non avrei potuto completare questo lavoro.

Voglio ringraziare tutto lo staff e i colleghi del DIEE e tutti i miei amici, specialmente quelli che mi hanno dato il loro supporto nei momenti difficili che ho incontrato.

Un ringraziamento speciale va alla mia famiglia, che in tutti questi anni mi ha sempre dato tutto il supporto morale e finanziario di cui avevo bisogno e non ha mai permesso che mi mancasse niente.

**Analysis of grafcet models by automatic generation of the equivalent timed automaton**

## Abstract

The objective of this thesis is to give a contribute to solve the problem of formal verification on GRAFCET models with time requirements.
GRAFCET is a standardized graphical model used to describe the behavior of sequential logic systems. Currently, all solutions offered to engineers to perform formal verification on GRAFCET models are based on simulation techniques. However, the generation of simulation sequences is an error-prone, tedious and time-consuming task. This explains why formal verification is today an active research area. Interesting results are now available for timed automata formal verification. For this class of models, the tool UPPAAL allows to verify automatically timed properties.
With this work we allow to take advantage of results obtained on timed automata and to apply them to perform formal verification of GRAFCET models. In order to apply this technique a translation algorithm that allows to express automatically the behavior of a GRAFCET model with a timed automaton has been carried out.

## Sommario

L'obiettivo di questa tesi è dare un contributo nel risolvere il problema della verifica formale di modelli descritti in termini di GRAFCET con vincoli temporali.
Il GRAFCET è un modello grafico standardizzato usato per descrivere il comportamento di sistemi logici sequenziali. Attualmente, tutte le soluzioni offerte agli ingegneri per effettuare la verifica formale di modelli GRAFCET sono basate su tecniche di simulazione. In ogni caso la generazione di sequenze per la simulazione è un processo incline all'errore, noioso e dispendioso in termini di tempo. Questo spiega perché la verifica formale al giorno d'oggi è un'attiva area di ricerca. Attualmente sono disponibili interessanti risultati riguardo la verifica formale degli automi temporizzati. Per questa classe di modelli l'applicazione UPPAAL permette di verificare automaticamente le proprietà temporali.
Grazie ai risultati di questo lavoro, è possibile sfruttare i vantaggi ottenuti sugli automi temporizzati e applicarli per effettuare la verifica formale di modelli GRAFCET. Per utilizzare queste tecniche, è stato realizzato un algoritmo che consente di esprimere in modo automatico il comportamento di un modello GRAFCET in termini di automa temporizzato.

# Contents

## Chapter 1



## Chapter 2

# List of figures

## Chapter 1



## Chapter 2



## Chapter 3

## Chapter 4



## Chapter 5

# List of definitions

*…il n'y a pas de problème, il y a seulement des solutions à trouver…*

*J.M. ROUSSEL*

# Chapter 1

# Introduction

## Contents

*In this chapter we introduce the objectives of this work. The problem of grafcet models formal verification is presented with the support of an example that we'll use in the entire document.*

*In the second part a brief bibliography about previous research works on this field is presented, then the approach used in this work is introduced.*

## 1.1 Objective of this work

The objective of this work is to contribute to the formal verification of GRAFCET models with time requirements.

GRAFCET is a standardized graphical model used to describe the behavior of sequential logic systems [IEC 60848]. This language is widely used in several industrial domains, like railway transport, electrical power production and manufacturing industry, to specify the expected behavior of logic controllers. Defined in France at the end of ‘70s, it was standardized in France at the beginning of the ‘80s, and at the international level in 1988. Since this date, several extensions have been proposed to enhance the modeling possibilities of this model and they are included in the latest version of the standard [IEC 60848].

Currently, all solutions offered to engineers to verify if a GRAFCET model is correct are based on simulation techniques. However, the generation of simulation sequences is an error-prone, tedious and time-consuming task. This explains why the formal verification is today an active research area. Interesting results are now available for the formal verification of timed automata. For this class of models, the tool UPPAAL permits to verify automatically timed properties.

So our idea is to take advantage of results obtained on timed automata and to apply them to perform formal verification of GRAFCET models. In order to apply this technique it is necessary to have a method that allows to express automatically the behavior of a GRAFCET model with a timed automaton. The objective of this work is to develop this method.

## 1.2 Problem presentation

The formal verification problem is the problem of verifying if a system satisfies to required properties. Mainly, there are two classes of specify:

- safeness specifies, that assure that no anomalous condition can be reached;
- liveness specifies, that assure that a required condition can be reached.

To perform a formal verification of GRAFCET based models is a very complex operation as we will illustrate with the support of an example. Let us consider the grafcet in figure 1.2.1.

This model is composed of five grafcet charts that describe the system behavior. A parallel behavior is implemented by the presence of five charts and there exist also several synchronizations as several transitions that can be fired simultaneously. The grafcet is characterized by five input variables (pump1_fault, pump2_fault, blockage, low debit, high debit) and eight output variables (Reversal, pump1, pump2, upstr pump1, upstr pump2, dwnstr pump1, dnwstr pump2, out). We have transition conditions that depend on input conditions (for example transition $t_{10\_11}$ in grafcet FMP1), transition conditions that depend on step activation (for example transition $t_{50\_51}$ in grafcet Func) and transition conditions that depend on step activation time (for example transition $t_{51\_53}$ in grafcet Func).

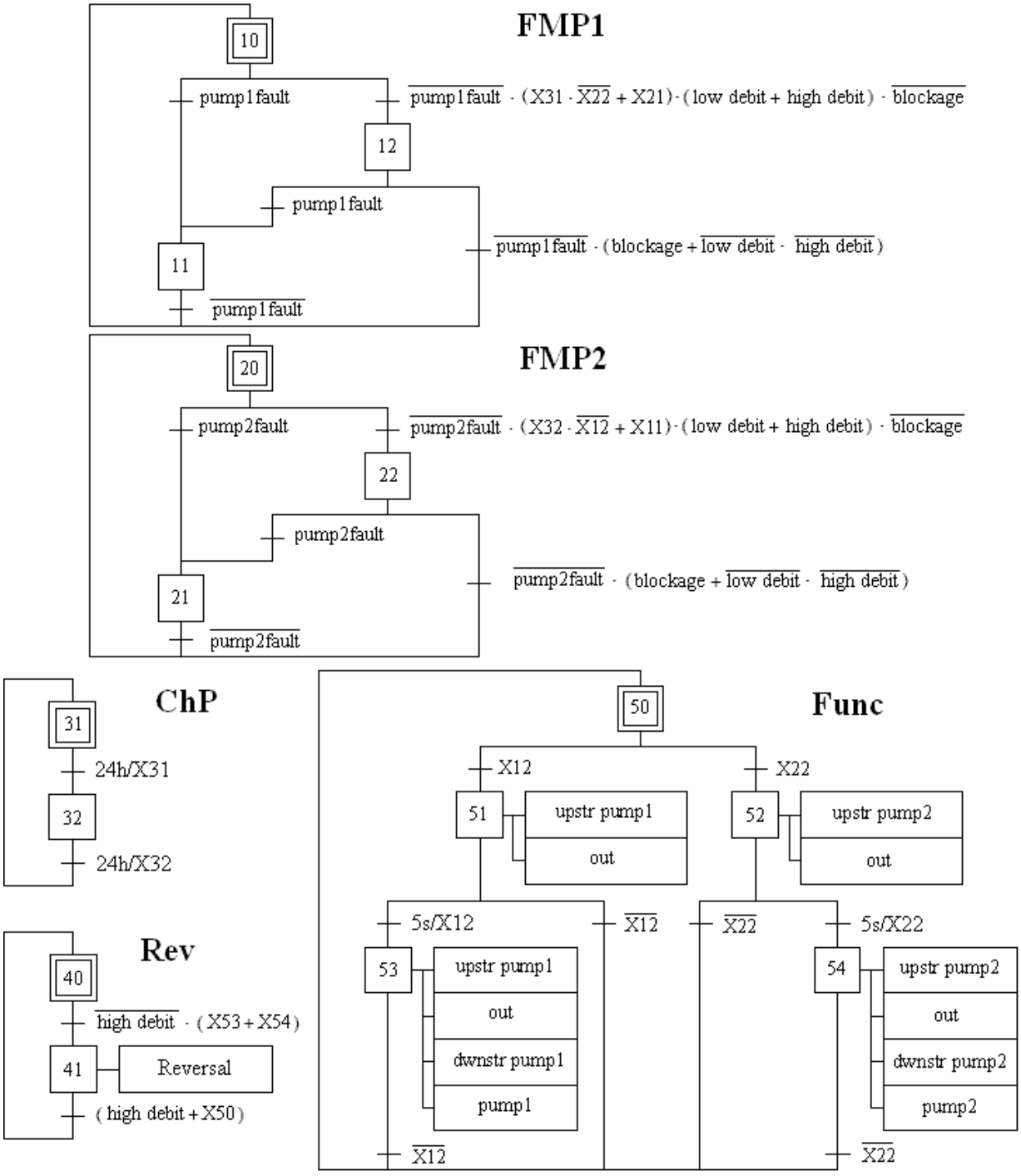


**Figure 1.2.1 Water distribution grafcet control model**

In order to present the problem of analysis of a GRAFCET[1] based model behavior let us consider a starting situation (a situation is a set of active steps) and an input configuration. We will calculate the evolution of the GRAFCET model and the reached situation.

Let us consider the following starting situation: {12,20,32,41,53}. The set of enabled transitions (a transition is said to be enabled if all its upstream steps are activated) from this situation is: $\{t_{12\_11},t_{12\_10},t_{20\_21},t_{20\_22},t_{32\_31},t_{41\_40},t_{53\_50}\}$.

Let suppose that input configuration is: pump1fault = 1, pump2fault = 0, blockage = 0, low debit = 1, high debit = 0. We suppose also that step 32 is active less than 24 hours.

The set of firing transitions is $\{t_{12\_11}\}$ and the reached situation is {11,20,32,41,53}. But the analysis is not completed yet as it still exists firing transitions.

The set of transitions that can be fired from {11,20,32,41,53} is $\{t_{20\_22},t_{53\_50}\}$ so we have another evolution that leads to situation {11,22,32,41,50} and we have to repeat the same analysis.

From situation {11,22,32,41,50} the set of transitions that can be fired is $\{t_{41\_40},t_{50\_52}\}$ so we have another evolution that leads to situation {11,22,32,40,52}. As step 52 is just activated, the set of transitions that can be fired is empty. This situation is a stable situation. We can evaluate the set of emitted outputs: {upstream pump2,out}.

This simple simulation shows how the behavior of a GRAFCET model could be complex. It is obvious that a manual analysis is impossible and engineers must be assisted.

## 1.3 Previous works

Several works have been developed to be able to express the behavior of a timed GRAFCET model with timed automata [L'Her 01], [Bauer 04].

In [L'Her 01], authors propose a method to translate a GRAFCET into a timed automaton in order to use the model-checker Kronos to perform formal verification. In this approach, a location of a timed automaton represents a possible situation of the grafcet, a specific

[1] A detailed description of GRAFCET formalism is given in chapter 2

combination of the input values and a specific combination of the temporizations.

From a technical point of view, this approach could be use only for very small grafcets due to the combinatorial explosion. For the grafcet on figure 1.2.1, we have: 32 different situations, 5 inputs, then 32 ($2^5$) different combinations of the input values, 4 temporizations, then potentially 16 ($2^4$) different combinations of temporizations.

The size of the generated timed automaton could be very important. Potentially, it could have 16384 different locations. For each location, it is also necessary to represent time evolutions or input changes. In our case, we have for each location, 31 or 32 transitions.

In [Bauer 04], authors propose a method to translate a SFC program into a timed automaton in order to use the model-checker Uppaal to perform formal verification. In this approach, a timed automaton is associated for each SFC without parallelism structure. For SFC with parallelism structure, several automata are associated. To take into account synchronization among SFC nets by step activation, authors introduce a Boolean variable in timed automata for each step. The value of this variable is fixed by the corresponding automaton.

This method could not be used directly for GRAFCET models as the behavior of a GRAFCET model and the behavior of a SFC model are different. For example, in a SFC, it is impossible to fire simultaneously two enabled transitions from the same step. In GRAFCET models, this evolution is possible.

The method proposed in [Bauer 04] is based on this characteristics. To be used for GRAFCET models, it is necessary to verify, before the translation step, that it is impossible to fire simultaneously two enabled transitions from the same step. For the grafcet on figure 1.2.1, this property is not verified for transitions $t_{50\_51}$ and $t_{50\_52}$. This grafcet could not be verified by this method.

Therefore a specific method is necessary.

## 1.4 Proposed method

The proposed method consists to calculate the whole space of state of the grafcet before the translation into a timed automaton. During this stage, all the specificities of GRAFCET evolution are taken into account, such as simultaneous firing of transitions or research of stability.

The work presented in this master thesis is an extension of a previous work [Roussel 94] developed by Jean-Marc Roussel for not-timed grafcet at the LURPA laboratory (ENS - Cachan). In his work he gives an important contribution to solve the problem of grafcet analysis first by

improving GRAFCET theoretic foundation as stability notion and by introducing an extended Boolean algebra (also dealt in [Roussel, Lesage 93]) and then by introducing a particular state machine called Graph of Accessible Situations. In this state machine each state represents a grafcet situation and each evolution represents a grafcet evolution. It is important to underline that only events that cause grafcet evolutions are taken into account and this is a key aspect since it leads to avoid combinatorial explosion.

In our work, in a first step we extend the concept of grafcet situation to the concept of grafcet configuration by integration of time constraints at a logic level, then, in a second step, time at logic level is extended to a physic level in order to obtain a timed automaton representation.

The first step allows to individuate all potentially reachable grafcet configurations and evolutions, then by analysis of timed automaton all physical timing constraints are taken into account to individuate which configurations are actually reachable.

## 1.5 Outline

In chapter 2 grafcet formalism is introduced with base elements and classical programming structures, a particular attention on stability problem is given. In chapter 3 a particular state machine (the Graph of Accessible Stable Configurations) used for grafcet representation and analysis is introduced. In chapter 4 after a brief presentation of theory notions about timed transition systems and timed automaton formalism, an algorithm that allows the translation of the Graph of Accessible Stable Configurations into a timed automaton is shown. In chapter 5 a case study of a complex grafcet analysis by using a python tool is presented with obtained results.

# Chapter 2

# The GRAFCET

## Contents

*In the first part of this chapter we present the GRAFCET formalism. All main aspects are introduced including action assignation and timed behaviors.*

*The second part is devoted to the presentation of GRAFCET based models stability analysis problem, in particular a stability criterion is given.*

# 2.1 Introduction

In 1975, in France, a commission composed of university and industrial engineers with the aim of carry out the means to describe the complex systems of industrial automation was established. The nature of the systems to be described fell into the category of discrete event systems, systems for which discrete space is not continuous, and whose evolution depends on the occurrence or otherwise of special events. The result of commission work was the definition of the GRAphe Fonctionnel de Commande Etapes-Transitions, or GRAFCET, which was adopted by the International Electrotechnical Commission in 1988, in the International Standard No 848, as a language for describing systems of industrial automation. The GRAFCET language, described by this standard, has served as foundation of SFC language of IEC standard 61131-3, but the syntax and the semantics defined by each of the two standards are nevertheless distinct because their scopes are different: GRAFCET is a specification language and SFC is a programming language. This standard is mainly for people (design engineers, realization engineers, maintenance engineers, etc) who need to specify the behavior of a system (control-command of automatic machine, safety component, etc). This specification language should also serve as a communication means between designers and users of automated systems. In fact the implementation of an automated system requires, in particular, a description relating cause and effect. To do that, the logical aspect of the desired behavior of the system musts be described. The sequential part of the system, which is accessed via Boolean input and output variables, is the logical aspect of this physical system. The behavior indicates the way in which the output variables depend on the input variables (see figure 2.1.1). The objective of the GRAFCET is to specify the behavior of the sequential part of the systems.

In this chapter we will present the main notions about GRAFCET. All information is taken from [IEC 60848], [J.Perrin, F.Binet 06] and [P.Chiacchio 04].

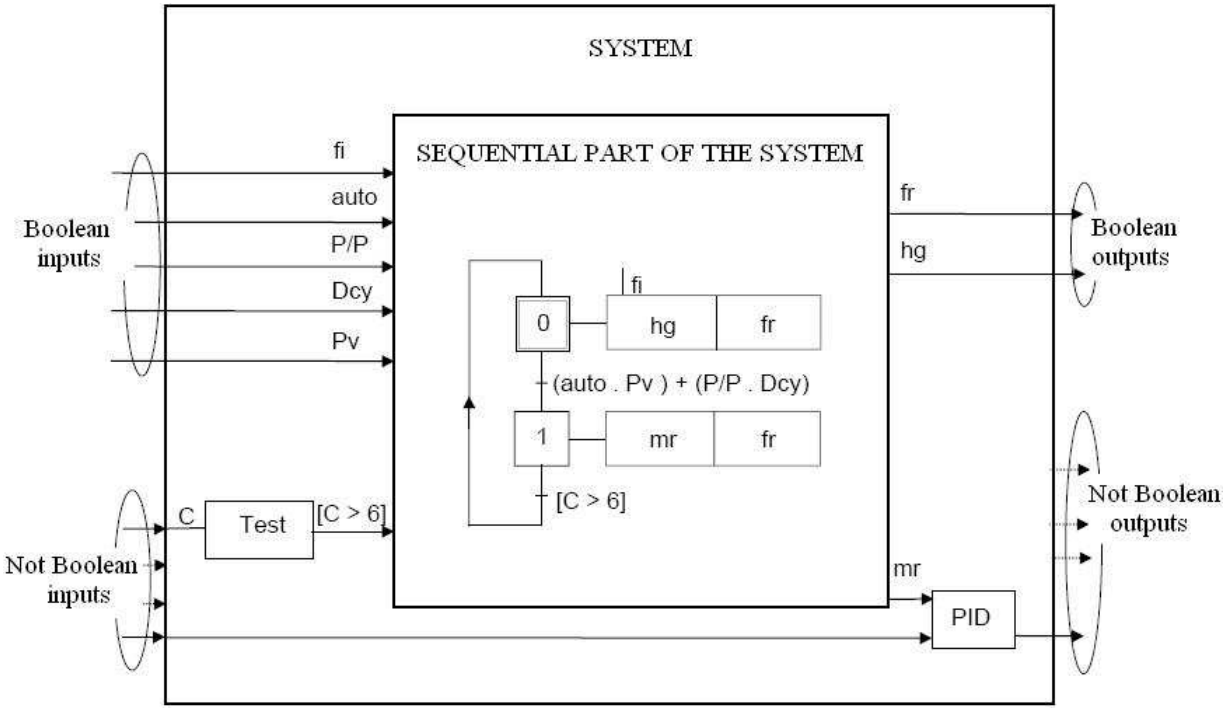


**Figure 2.1.1 A grafcet as sequential part of a system**

# 2.2 Terms and definitions

(In alphabetic order)

- **action**: language element associated with a step indicating an activity to be performed on output variables;
- **chart, graph**: graphical presentation describing the behavior of a system;
- **directed link**: language element indicating the evolution paths between steps by connecting steps to transitions and transitions to steps;
- **grafcet chart**: function chart using GRAFCET;
- **interpretation**: part of the GRAFCET enabling to link:
  - the input variables and the structure, by the means of the transition-condition;
  - the output variables and the structure, by the means of the actions;

- **situation**: name of the state of the system described by grafcet and characterized by the active steps at a given instant;

- **step**: language element used for the definition of the sequential part of the system;

- **transient evolution**: evolution characterized by the clearing of several successive transitions on the occurrence of a single input event;

- **transition**: language element indicating a possible evolution of the activity between two or more steps;

- **transition condition**: language element associated with a transition indicating the result of a Boolean expression.

## 2.3 Structure and interpretation

The GRAFCET is used for the designing of grafcet charts to provide a graphical and synthetic representation of sequential system behavior. The representation distinguishes:

- the **structure**, which allows possible evolutions between the situations to be described;

- the **interpretation**, which enables the relationship between input, output variables and the structure (evolution, assignation and allocation rules are necessary to achieve this interpretation).

The structure comprises the following basic items:

- **step**: a step is either active or inactive, the set of active steps of a grafcet chart at any given instant represents the situation of this grafcet at this instant. We represent graphically a step with a rectangle and we distinguish active steps from inactive steps by a spot.

- **transition**: a transition indicates that an evolution of the activity between two or more steps may evolve. This evolution is realized by the clearing of the transition. We represent graphically a transition by a transversal line.

- **directed link**: a directed link connects one step to a transition, or a transition to one steps.

The following elements are used for the interpretation:

- **transition-condition**: associated with each transition, the transition-condition is a logical expression which is true or false and which is composed of input variables and/or internal variables.

- **action**: the action indicates, in a rectangle, what shall be done on the output variable, either by assignation (continuous action), or allocation (stored action).

Figure 2.3.1 shows the main grafcet elements.

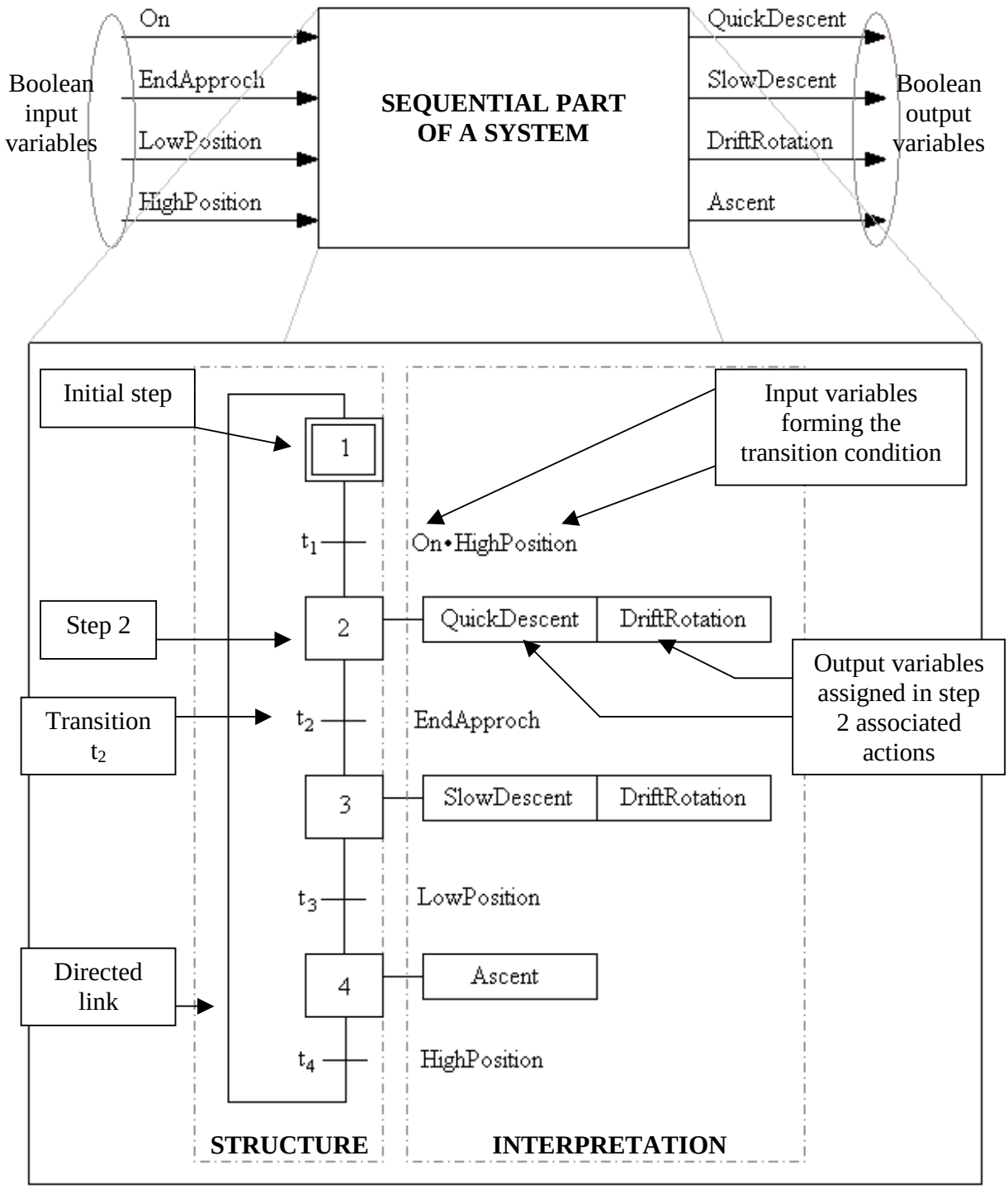


**Figure 2.3.1 Structure and interpretation elements used in a grafcet chart**

# 2.4 GRAFCET rules

## 2.4.1 Syntax rule

*Step transition and transition step alternation shall always be respected in all forms of sequence.*

Consequences:

- two steps shall never be connected directly by a directed link;
- the directed link shall only connect a step to a transition or a transition to a step.

## 2.4.2 Evolution rules

As each situation is characterized by the set of active steps at a given instant, the GRAFCET evolution rules only affect the application, on the steps, of the evolution principle between the situations of the sequential part of the system.

***Rule 1**: The initial situation, chosen by the designer, is the situation at the initial time.*

The initial situation is the situation at the initial time. Therefore it is described by the set of steps active at this time. The choice of the situation at the initial time depends on the methodology relating to the type of sequential part of the considered system. Graphically, the initial steps are indicated with a double rectangle.

***Rule 2**: A transition is said to be enabled when all immediately preceding steps linked to this transition are active. The clearing of a transition occurs when the transition is enabled and its associated transition-condition is true.*

***Rule 3**: The clearing of a transition causes simultaneously the activation of all the immediate succeeding steps and the deactivation of all the immediate preceding steps.*

***Rule 4**: Several transitions which can be fired (cleared) simultaneously are simultaneously fired.*

The evolution between two active situations implies that no other intermediate situation is possible, the change from one representation of the situation by a set of steps to another representation is instantaneous.

***Rule 5**: If during the operation, an active step is simultaneously activated and deactivated, it remains active.*

If a step is included in the description of the preceding situation and of the following one, it can only, therefore, remain active.

## 2.5 Input events and internal events

### 2.5.1 Input events

Evolution rules show that only a change in the values of the input variables may cause the evolution of the grafcet. This change called "input event" shall be defined by the preceding value and the succeeding value of all the input variables for characterizing this single event. In practice,

a set of input events is specified only by the state change characterized (rising edge[1] or falling edge[2]) by one or several Boolean input variables. We can say that "the event occurs" at the date of the change of state of the input variables which characterize it. The input event specification is implemented by a logical expression of one or several characteristic variables, usually in a transition-condition. It may also directly specify an internal event but more rarely.

### 2.5.2 Internal events

Only certain input events could occur from a given situation. The connection between a situation and input event, which may occur from this situation, is called internal event. This notion is mainly used by the designer to condition an output allocation to a set of internal events. The description of a set of internal events is performed by one of the following ways:

- **step activation**: the step activation describes the set of internal events each of which has this step activation as a consequence.
- **step deactivation**: the step deactivation describes the set of the internal events which have, for each one, this step deactivation as consequence.

## 2.6 Output modes

The actions enable links to establish the connection between the evolution of the grafcet chart and the outputs. Two output modes, continuous mode or stored mode, describe how the outputs depend on the evolution and on the system inputs.

### 2.6.1 Continuous mode (assignation on state)

In the continuous mode, the association of an action with a step indicates that an output

---

[1] The rising edge of a logical variable, indicated by the sign "↑" in front of a Boolean variable, indicates that this rising edge is only true for the change from value 0 to value 1 of the concerned variable.

[2] The falling edge of a logical variable noted by the sign "↓" in front of a Boolean variable, indicates that this falling edge is only true for the change from value 1 to value 0 of the concerned variable.

variable has a true value if the step is active and if the assignation condition is verified. The assignation condition is a logical expression of the input variables and/or the internal ones. If one of the conditions is not met and provided that no other action relating to the same output meets the conditions, the concerned output variable takes the false value. Assignation refers to imposing the value of the output variables (true or false). The set of local assignations (relating to the active steps at a given instant) defines the assignation of all the output variables for this situation.

***Assignation rule**. For a given situation, the value of the outputs relating to the continuous actions is assigned:*

*to the true value, for each output relating to the actions associated with active steps and for which the assignation conditions are verified;*

*to the false value, for the other outputs (which are not assigned to the true value).*

Figure 2.6.1 shows an example of an action assigned in continuous mode.

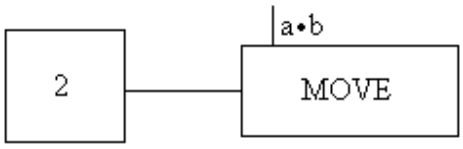


**Figure 2.6.1 Example of continuous mode action assignation**

Action **MOVE** is executed if step 2 is active and condition **a·b** is true

## 2.6.2 Stored mode (allocation on event)

In the stored mode, the association of an action to internal events is used to indicate that an output variable takes and maintains the enforced value if one of these events occurs. Explicit representations are necessary to describe the association of the actions with the events (activation step, deactivation step, ...).

The value of an output relating to a stored action remains unchanged until a new specified event modifies its value. Allocation refers to storing, at a considered instant, a determined value

to an output variable.

> ***Allocation rule**: the value of an output, relating to a stored action and associated with an event, is allocated to the indicated value, if the specified internal event occurs the value of this output is null at the initialization.*

Figure 2.6.2 shows an example of action assigned in continuous mode.

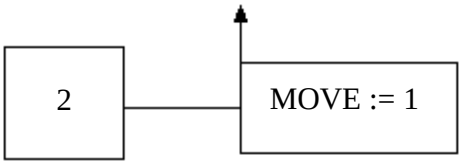


**Figure 2.6.2 Example of stored mode action assignation**

The allocation of the value 1 to the output variable **MOVE** is performed on the occurrence of one of the input events having the activation of the step 2 as consequence.

# 2.7 The temporizations

In GRAFCET standard we can associate temporizations to expressions depending on input events and internal events. A condition on a temporization, like Boolean conditions, can be used in transition-conditions and in assignation conditions. We have two forms of condition on temporization: TON and TOF.

## 2.7.1 TON

The TON condition is indicated with d/exp, where d represents the delay time and exp is an expression that can depend on input or/and internal variables. TON functioning is described in figure 2.7.1.

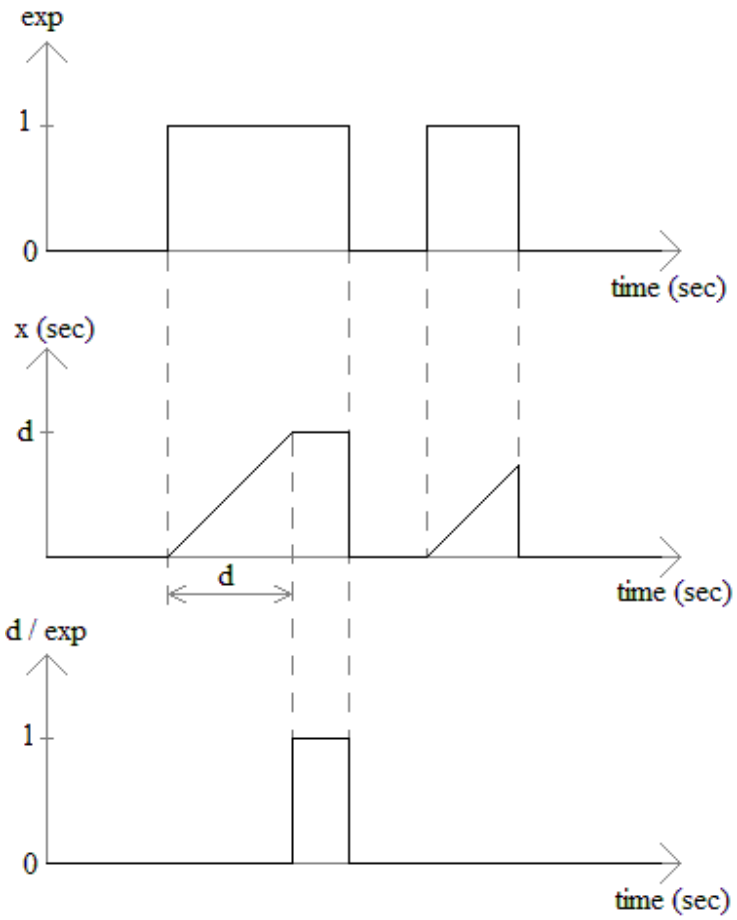


**Figure 2.7.1 A representation of TON condition**

The condition d/exp becomes true when x becomes equal to d. The clock x counts the time elapsed from becoming true of expression exp end stops when the value d is reached, finally x is reset when exp becomes false.

## 2.7.2 TOF

The TOF condition is indicated with exp/d, where d represents the delay time and exp is an expression that can depend on input or/and internal variables. TOF functioning is described in figure 2.7.2.

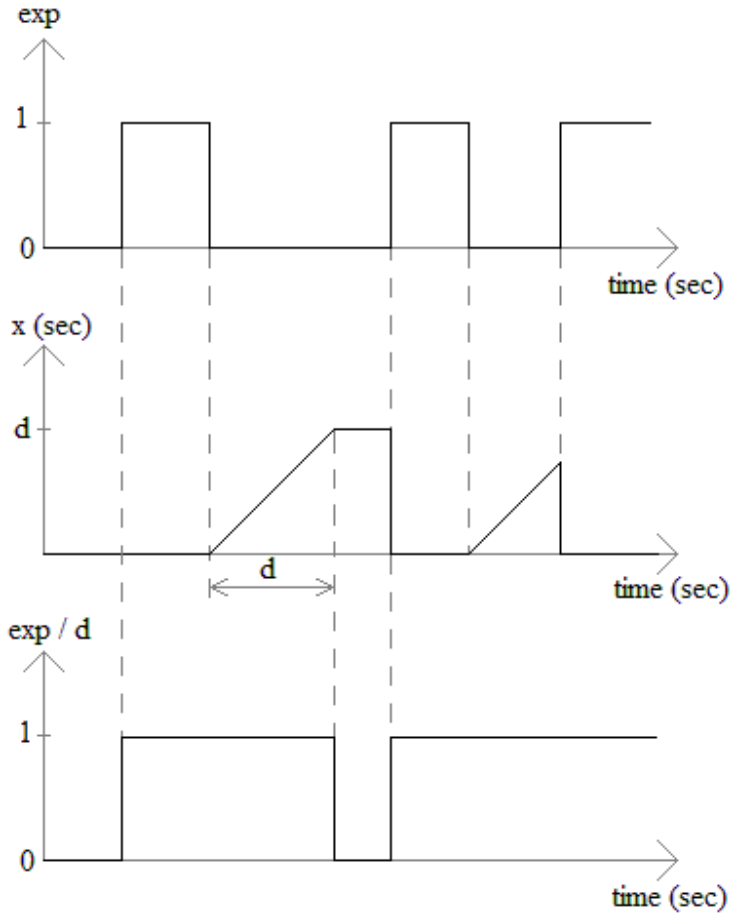


**Figure 2.7.2 A representation of TOF condition**

The condition exp/d becomes true when exp becomes true and it is still valid until x becomes equal to d. The clock x counts the time elapsed from becoming false of expression exp end stops when the value d is reached, finally x is reset when exp becomes true.

# 2.8 Classical programming structures

In this paragraph several used classical programming structures are presented. In fact in addition to simple sequence, there are other very useful particular structures.

## 2.8.1 Selection of sequences

We have a selection of sequences when a step is followed by more of one transition. If we

want to carry out a choice we have to ensure us that the different transition-conditions are mutually exclusive.

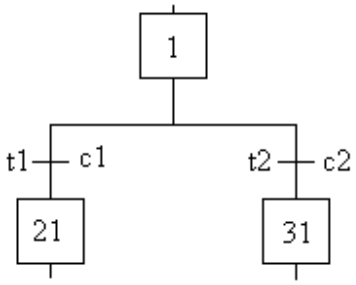


**Figure 2.8.1 Selection of sequence**

In figure 2.8.1 we have to check that $c1 \cdot c2 = 0$, in fact starting from situation {1} in which step 1 is activated the grafcet can perform the three evolutions shown in figure 2.8.2.

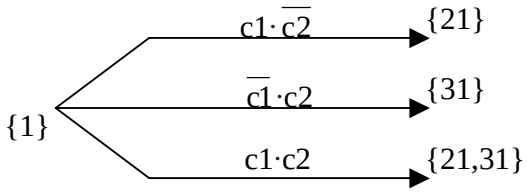


**Figure 2.8.2 Possible evolutions for grafcet in figure 2.8.1**

If c1 and c2 are not mutually exclusive for each possible combination we have to carry out a mutual exclusion to avoid that steps 21 and 31 (and so their following sequences) become active simultaneously. However there exist some cases in which one can desire that under certain conditions several sequences become active simultaneously, in these cases we talk about interpreted parallelism: the structure is the same of figure 2.8.1 but a simultaneous sequence activation is allowed.

Particular cases of selection of sequences are step skip and backward skip.

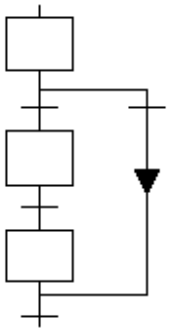

**Figure 2.8.3 Step skip**

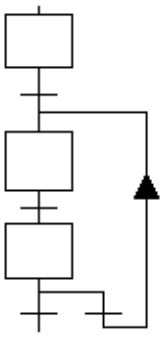

**Figure 2.8.4 Backward skip**

## 2.8.2 Activation and synchronization of parallel sequences

We have an activation of parallel sequences when a transition is followed by several steps. In figure 2.8.5 if transition t1 is fired step 12, 22 and 32 become active and they lead to three sequences that will evolve independently.

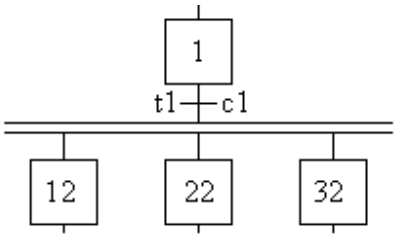


**Figure 2.8.5 Activation of parallel sequences**

Synchronization occurs when several steps precede a transition. Necessary condition to have transition firing is that all the parallel sequences are completed, so that the final steps of each

sequence are active. With reference to figure 2.8.6 the transition t1 is validated if and only if steps 1F, 2F and 3F are simultaneously active.

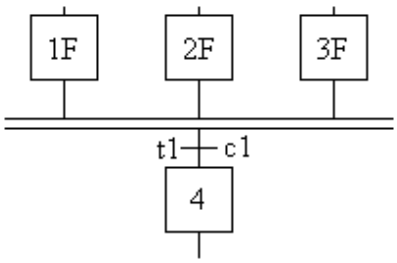


**Figure 2.8.6 Synchronization of sequences**

# 2.9 Temporal boundary of a given grafcet (UTE C 03-191)

The insulation of a system described by a grafcet establishes a description boundary that defines a portion of the universe in an internal one and an external one. This boundary insulation corresponds to a border between an internal temporal scale and an external temporal scale to the model.

From the point of view of the external time scale an event causes a change of the instantaneous state of the system outputs. From the point of view of the internal time scale, the time interval between the occurrence of a condition-transition that becomes true and the actual firing of the transition is very small but not zero.

This separation is shown in figure 2.9.1. The example is taken from [Roussel 94].

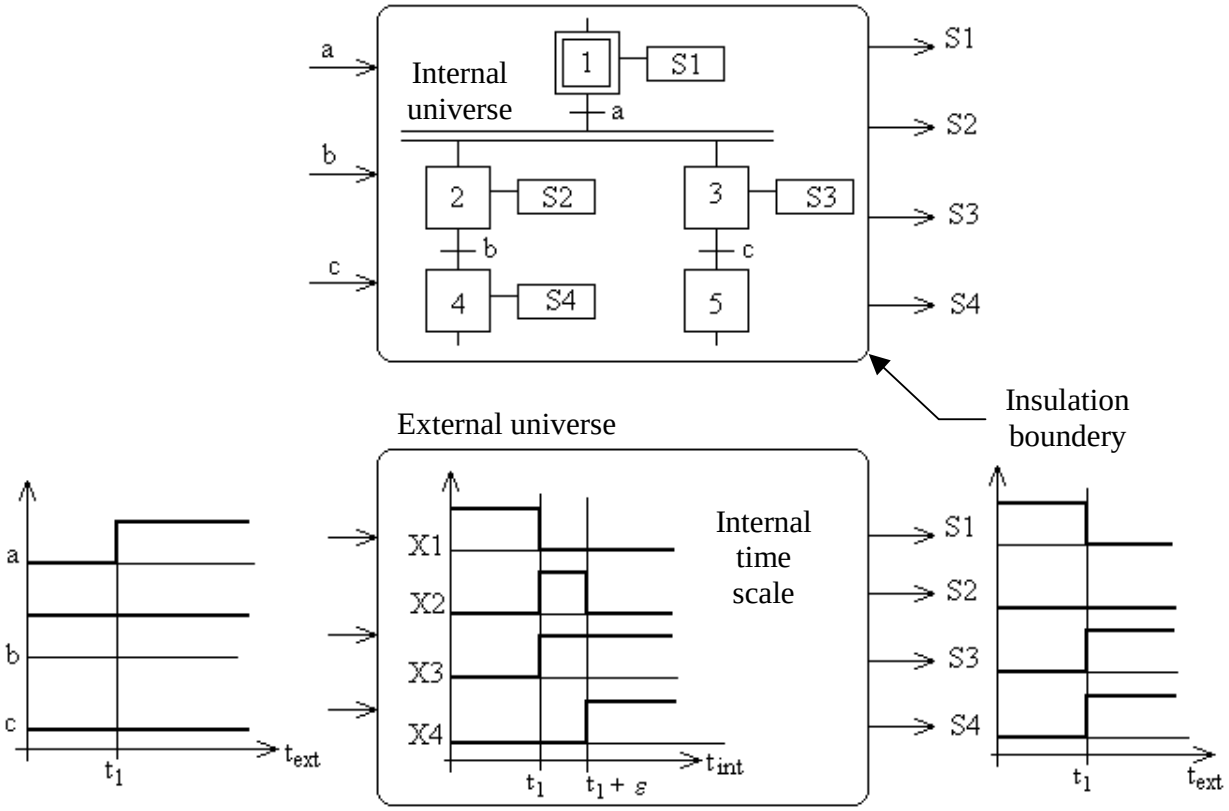


**Figure 2.9.1 Internal and external universe**

Following the introduction of dual time scale, the GRAFCET now presents the following characteristics:

- only outputs associated with steps that belong to stable situations are generated;
- only stable situations are sensible to input state changes;
- only instable situations are sensible to step activations/deactivations;

## 2.10 The stability notion

What has been introduced suggests that it is necessary to introduce a criterion of stability on the GRAFCET.

### 2.10.1 Stability of a situation

In [afcet 83], the stability of a situation is defined as follows:

*1. Situation stability*

*When a grafcet reaches a situation, this is called unstable if at least a transition is clearable, if any transition is clearable then the situation is called stable.*

By considering this definition to emphasize an important aspect is important: the stability of a situation depends not only on the situation itself but also on changes of the inputs that have allowed achieving this situation.

### 2.10.2 Total instability criterion

The two criteria mentioned above need to be completed because the essential properties required by the models is that they are cycle free, that is there are not stationary situations or totally unstable situations. It is therefore necessary to define a criterion of instability.

The first two stability criteria considered were the following.

*2. Stability criterion (1)*

*The total instability has verified when the number of evolutions between two external events is greater than the number of steps in the overall grafcet.*

*3. Stability criterion (2)*

*The total instability has verified when a grafcet reaches twice the same situation between two external events.*

In his PhD thesis [Roussel 94], Jean-Marc Roussel, shows with the following two examples that these two criteria of instability are not completely correct.

### Example 1

This example puts in a critical position the first criterion. With reference to figure 2.10.1, the grafcet consists of 7 steps but it needs 12 evolutions to reach the stable situation {10,20} from the initial situation to the occurrence of ↑m.

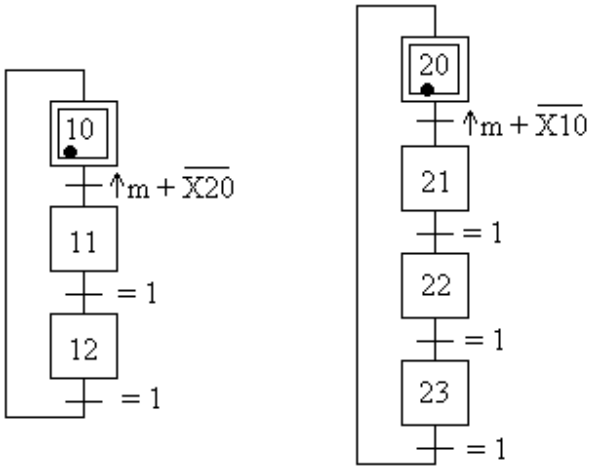


**Figure 2.10.1 Example of crisis of first stability criterion**

## Example 2

This example puts in a critical position the second criterion. With reference to figure 2.10.2, starting from initial situation, if ↑m occurs the grafcet reaches twice the unstable situation {10,21} and finally it reaches the stable situation {10,20}.

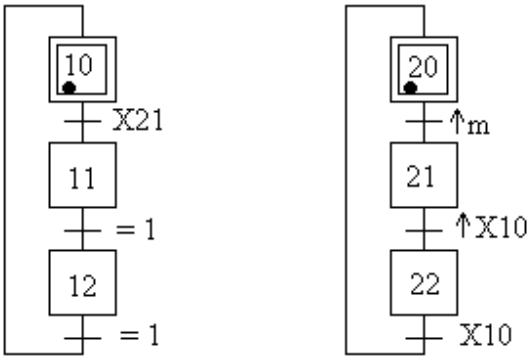


**Figure 2.10.2 Example of crisis of second stability criterion**

Then, in his PhD thesis Jean-Marc Roussel shows a new stability criterion:

### *4. Used stability criterion*

*If in occurrence of an external event a grafcet performs twice the same evolution, then we have total instability.*

A correct grafcet verification implies a correct and complete application of all GRAFCET

standard behaviors and a complete study of grafcet behavior at the internal and external time scale. We perform this analysis by construction of the Graph of Accessible Stable Configurations, that is presented in the following chapter.

# Chapter 3

# The Graph of Accessible Stable Configurations

## Contents

*This chapter illustrates all aspects of the state transition machine that we have carried out in order to study the behavior of GRAFCET based models: the Graph of Accessible Stable Configurations (GASC).*

*In the first part we present the elements of this state transition machine as state and evolution and how time is taken into account. In order to illustrate all following aspects we introduce the context notion.*

*In the second part a particular tree structure (the Tree of Accessible Configurations TAC) that allows a grafcet analysis at the internal time scale is introduced. The construction of this tree is indispensable in order to obtain the Graph of Accessible Stable Configurations.*

*In the third part the algorithms that allow to build the TAC and the GASC are shown. The stability analysis is performed with the TAC construction.*

*In the last part we apply to a grafcet portion these algorithms.*

## 3.1 Introduction

The Graph of Accessible Stable Configurations (GASC) allows a complete analysis of a grafcet behavior at the external time scale by evaluation of all possible grafcet states and evolutions. In order to be sure of correctness of our analysis we have to apply all GRAFCET standard rules. The form of GRAFCET temporization that we deal with in this work is the TON applied on grafcet steps, but a particular grafcet translation that allows to deal with TON and TOF on input variables is presented in the following of this document.

The GASC is an extension of the Graph of Accessible Situations [Roussel 94] where the situation notion is extended to configuration notion to perform time integration. It is important to underline that time notion is dealt only at a logic level then all timed events and conditions are dealt as Boolean variables. The extension of time at the physical level is performed in a second analysis step with the construction of the equivalent timed automaton.

The construction of the GASC needs an intermediate step, in fact to obtain grafcet behavior at the external time scale it is necessary to perform a first analysis at the internal time scale, so we have to build a Tree of Accessible Configurations where only root and leafs are stable configurations and all internal nodes are intermediate configurations.

In the following of this chapter this state transition machine will be introduced with all main characteristics.

## 3.2 Configuration

The first notion that we have to introduce is the configuration notion. In GRAFCET standard a situation *Sit* is defined as a set of active steps. But in order to study all possible grafcet evolutions starting from a situation, the only information about active steps is not enough. For example in figure 1.2.1, starting from situation {12,20,32,41,53}, it is necessary to know how long step 32 has been activated. Then we have to extend the notion of situation by integration of information about time.

In order to integrate a compact time representation we introduce a second set *Temp* that includes all grafcet timed conditions that are verified. A third set is necessary to a complete representation of a grafcet state and this is the set of emitted outputs *Out*. Moreover, to complete

our representation, we associate a clock[1] *h* to each step for which at least a timed expression is associated.

So we define as follows a configuration:

*5. Configuration*

*A configuration Cfg is defined as a set of active steps Sit, a set of true timed expressions Temp and a set of emitted outputs Out.*

For example, for situation {12,20,32,41,53} of grafcet in figure 1.2.1 there are four distinct configurations:

- 1:{12,20,32,41,53},{},{Reversal,upstr pump1,out,dwnstr pump1,pump1};
- 2:{12,20,32,41,53},{5s/X12},{Reversal,upstr pump1,out,dwnstr pump1,pump1};
- 3:{12,20,32,41,53},{24h/X32},{Reversal,upstr pump1,out,dwnstr pump1,pump1};
- 4:{12,20,32,41,53},{5s/X12,24h/X32},{Reversal,upstr pump1,out,dwnstr pump1,pump1}.

By looking at the figure 1.2.1 it is obvious that configuration 3 and 4 are not stable configurations: in fact if condition 24h/X32 is verified we have transition $t_{32\_33}$ firing and then an evolution leads the grafcet to a new situation (and consequently a new configuration). So it is necessary to introduce the notion of invariant condition associated with a configuration.

### 3.2.1 The configuration invariant condition

The invariant condition associated with a configuration is a Boolean expression on input and internal variables for which there are not possible evolutions: this is the stability condition for a configuration. In order to avoid grafcet evolutions all the conditions relating to enabled transitions must be false. Moreover, we have to consider all possible variations of emitted outputs set Out, then all input and internal variable variations that modify this set lead to a GASC evolution. Finally we have to consider all possible variations of true timed condition set Temp.

We can give the following definitions:

---

[1] A more detailed description about temporizations will be dealt in the following of this chapter

### *6. Situation invariant*

*The situation invariant condition is given by negation of application of OR operator between the transition-conditions cond($t_i$) of all validate transitions $t_i$. All step activation variables and timed expressions are replaced with theirs values.*

$$\mathbf{Inv(sit) = \overline{\sum_i cond(t_i)}}$$

### *7. Timed invariant*

*The timed invariant condition is given by negation of application of OR operator between all timed expression tmp associated with the activated steps that are not in the true timed condition set Temp.*

$$\mathbf{Inv(Temp) = \overline{\sum_{m,n} d_m / X_n}}$$

$$\mathbf{where\ d_m/X_n \notin Temp}$$

### *8. Output invariant*

*The output invariant condition is given by negation of application of OR operator between the assignation condition cond($O_k$) of all outputs $O_k$, associated with each step in Sit, that are not present in the Out set and the negation of assignation condition cond($O_{exk}$) of all outputs $O_{exk}$ associated with each active step in the Out set. All step activation variables and timed expressions are replaced with theirs values.*

$$\mathbf{Inv(Out) = \overline{\sum_h cond(O_h) + \overline{\sum_k cond(O_{exk})}}}$$

*9. Configuration invariant*

*The configuration invariant condition is given by application of AND operator between these three invariant conditions*

$$\mathbf{Inv(Cfg) = Inv(Sit) \cdot Inv(Temp) \cdot Inv(Out)}$$

Let us consider, for example, the following configuration of grafcet in figure 1.2.1:

Cfg = 1:{12,20,32,41,53},{},{Reversal,upstr pump1,out,dwnstr pump1,pump1};

the three defined sets are:

Sit = {12,20,32,41,53}

Temp = {}

Out = {Reversal,upstr pump1,out,dwnstr pump1,pump1}

and the invariant conditions:

$$Inv(Sit) = \overline{pump1fault \cdot (blockage + \overline{low\ debit} \cdot \overline{high\ debit}) + pump2fault + high\ debit}$$

$$Inv(Temp) = \overline{24h/X32} \cdot \overline{5s/X12}$$

$$Inv(Out) = 1$$

# 3.3 Events

As each GASC configuration is a stable configuration, we need at least an event to lead the GASC from a configuration to another one. We have an input event every time that at least an input variable changes its state and we have a timed event every time that a timed expression changes its state. All timed events lead to a change in the Temp set but not all input events lead to a change in the Sit or Out set. As temporizations are internal variables of grafcet and inputs are external variables of grafcet, since there is a boundary between internal and external grafcet universe (par. 2.9), it is impossible to have simultaneity between input and timed events (they are uncorrelated). This result can be explained with an example, let consider the following grafcet portion:

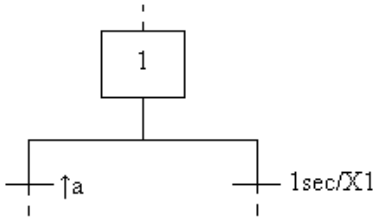


**Figure 3.3.1 Timed and input events**

It is impossible that the two events ↑a and ↑(1sec/X1) occur simultaneously: if we consider event ↑a probability we can suppose that its probability has an uniform distribution ([A.Papoulis 02]) in the time (figure 3.3.2), to assume that ↑a occurs simultaneously with occurrence of ↑(1sec/X1) implies that the area of a rectangle with base 0 is not 0 and this is not possible (in figure 3.3.2 $t_0$ is the instant of activation of step 1).

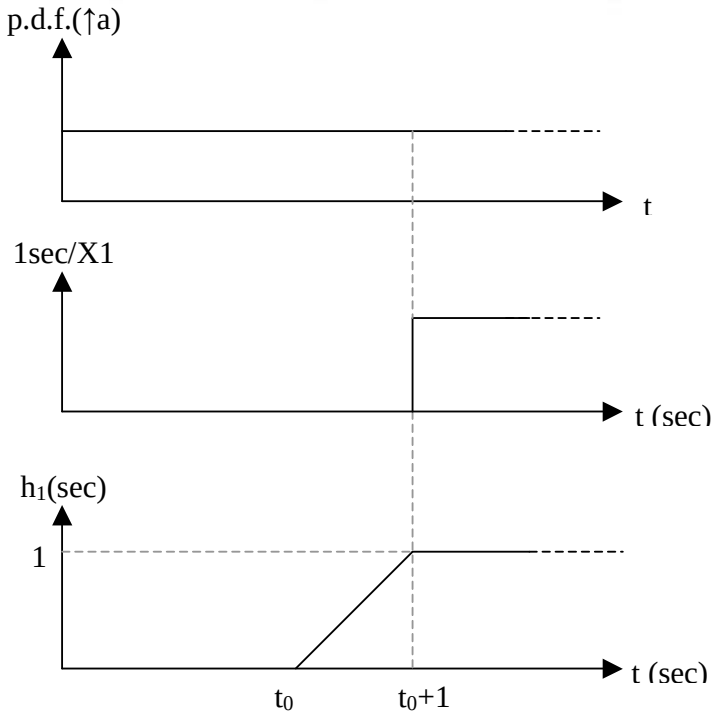


**Figure 3.3.2 Input events p.d.f. and timed events**

# 3.4 Evolution

In the Graph of Accessible Stable Configurations an evolution is caused by a change of at least an input variable state or a timed expression state that leads from a stable configuration to another one.

We distinguish evolutions on the basis of cause and consequence. The cause of an evolution can be an input event or a timed event, the consequence of an evolution, mainly, can be a configuration change that implies a variation in the Sit set (then at least a transition is cleared) or a configuration change that doesn't imply changes in Sit set. We summarize all possible combinations in table 3.4.1.

| Cause | Possible consequence | | | |
|---|---|---|---|---|
| timed event | transition clearings (a variation in Sit) | Temp set variation | Out set variation | Clock resets |
| input event | transition clearings (a variation in Sit) | Temp set variation | Out set variation | Clock resets |

**Table 3.4.1 Evolution causes/consequences**

Now, as the objective of GASC construction is a complete grafcet analysis, in our state machine we have to report all information about evolutions: in each evolution we have to include all causes and all consequences. Only by using this approach we can track all grafcet behaviors.

As shown in paragraph 1.2, the consequence of an input or timed event can be the firing of one or more transitions (simultaneously), moreover, before to reach a stable grafcet situation (and then a stable GASC configuration), a grafcet can reach one or more intermediate not stable situations.

Generally, we can say that the consequence of an input or timed event is the firing of a sequence of transition sets *SeqT* (where *T* is a set of cleared transitions).

In order to complete our representation we have to consider also clock resets: we have a clock reset Res($h_i$) if the new configuration contains a new step i for which a timed expression (and then a clock $h_i$) is associated. As for transition firing, generally, we associate with an evolution a sequence of set of clock resets *SeqR* (where *R* is a set of clock resets). The evolution cause is represented by an expression that depends on input variables and timed expressions, this

is called evolution context[1].

*10. Evolution*

*An evolution is characterized by a starting configuration StartCfg, an arrival configuration ArrCfg, the evolution context Cevo, a sequence of cleared transition sets SeqT and a sequence of clock reset sets SeqR.*

As it's not possible to have timed events and input events simultaneously, we study separately evolutions caused by input events and timed events: this is a very useful simplification.

# 3.5 Grafcet temporizations and timed events

An efficient and simple way of dealing with GRAFCET temporization is the key to perform a low complex analysis of grafcet models. We have decided to deal only with a grafcet temporization form: TON applied on a single step activation variable Xi. We made this choice because grafcet clocks can be stopped in particular situations, but in timed automaton formalism it is not possible to stop a clock. This problem can be solved in the case of TON applied on a single step variable activation but it can't in all other cases.

## 3.5.1 Time representation

In the GASC we have decided to represent the time at a "logic level". All timed expressions will be dealt as Boolean conditions that can be true or false.

These considerations allow a translation of a grafcet configuration into a timed automaton state. Physically in the equivalent timed automaton the clock is always active but this is not a problem because in all automaton states for which in the grafcet the relating clock $h_i$ is stopped, there are not evolution conditions that depend on clock $h_i$ (because all relating conditions are replaced with false).

---

[1] See paragraph 3.6

### 3.5.2 Several temporizations associated with the same step

A necessary observation is that several timed expressions can be associated with a step. Now, it's obvious that it is not necessary to associate several clocks with the same step, because they depend on the same activation step variable.

Let consider an ordered M-dimensional set *Tcond(n)* (read Timed Conditions on n) of timed conditions associated with a step n $<d_1/X_n,d_2/X_n \ldots d_m/X_n,d_{m+1}/X_n \ldots d_M/X_n>$ where $d_1<d_2<\ldots<d_M$: each condition $d_m/X_n$ can't be true if the previous condition $d_{m-1}/X_n$ is false.

For each step n we consider an ordered subsets of Tcond(n): the ordered set of timed conditions on n that are still false *FTcond(n)* (read False Timed Conditions on n). Formally:

$$\mathbf{FTcond(n) = \forall\ d_m/X_n \mid (d_m/X_n \in\ Tcond(n)) \cap (d_m/X_n \notin Temp)}$$

Then, for each GASC configuration analysis, we associate with each step n in Sit a set FTcond(n) and we define for each configuration a set *Wtemp* (read Waiting Temporizations) that includes all first elements of all associated FTcond(n).

This procedure can be clarified with the support of example in figure 3.5.1.

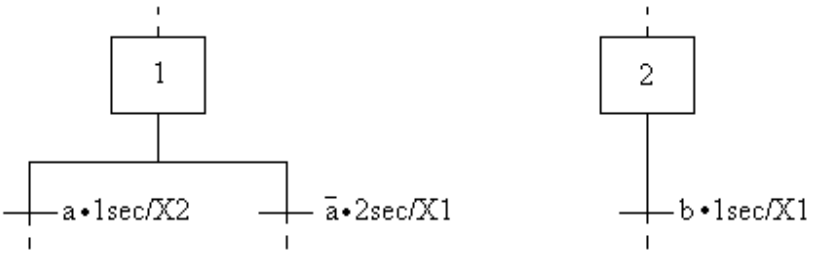


**Figure 3.5.1 Grafcet and timed events**

Let consider the configuration 1:{1,2},{},{}: the defined sets are:

Tcond(1) = <1sec/X1,2sec/X1> , Tcond(2) = <1sec/X2>

FTcond(1) = <1sec/X1,2sec/X1> , FTcond(2) = <1sec/X2>

Wtemp = {1sec/X1,1sec/X2}

Only timed expressions in Wtemp can cause a timed evolution and then only these timed expressions must be considered for the given configuration analysis.

## 3.6 Context

Starting from a given stable configuration we can have the same evolution by different variations of input/internal variable state, so we can compact the representation with an expression in II that summarize all possible variations that cause the same evolution.

### 3.6.1 Not timed context

This notion includes the concept of event and condition, and it is tightly related to the Boolean algebra extension II introduced in [Roussel, Lesage 93]. When it is not specified, with context, we intend not timed context. In this case we study only input and internal variable variations while all timed conditions don't change.

*11. Context*

*We call context the II combinatorial expressions that we associate with situations, evolutions and transition sets.*

Each context characterizes a precise set of internal and external variable variations.
We associate with each reached situation Sit a residual context and a stability context.

*12. Residual context*

*A residual context Cres describes the set of input and internal variable variations and timed conditions that allows reaching this situation.*

*13. Stability context*

*A stability context Csta is the part of residual context for which the reached situation is stable.*

We associate with each transition set T a minimal context and a maximal context.

*14. Minimal context*

*A minimal context Cmin describes the set of input and internal variable variations and timed conditions that allows to clear this set of transitions (necessary condition for clearing).*

*15. Maximal context*

*A maximal context Cmax is the part of the minimal context for which only this set of transitions is cleared (necessary and sufficient condition for clearing).*

We associate with each evolution an evolution context.

*16. Evolution context*

*An evolution context Cevo describes the set of input and internal variable variations and timed conditions that allows the occurrence of this evolution.*

## Example

In order to show these definitions we use the example in figure 3.6.1.

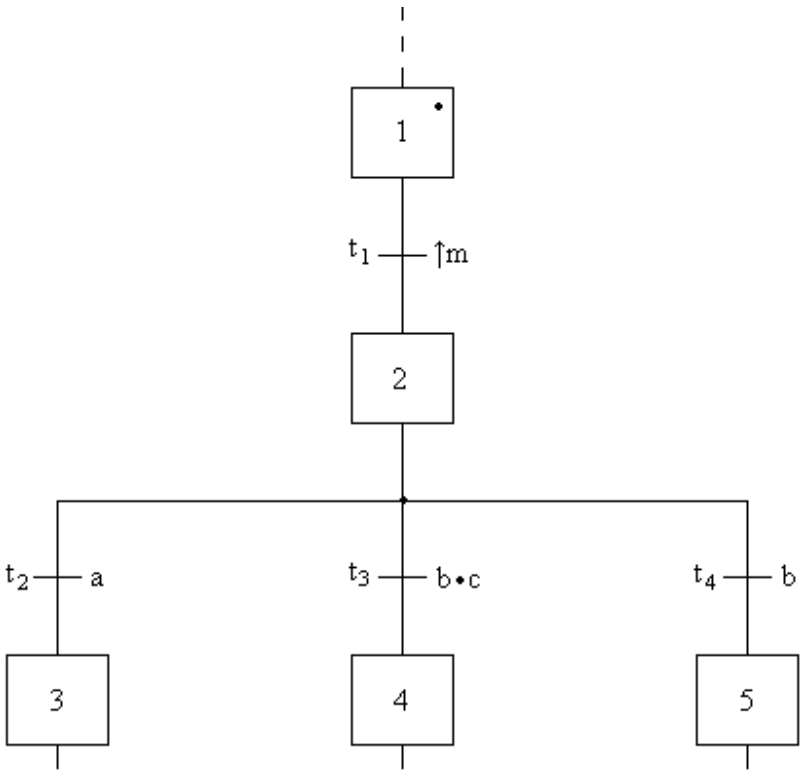


**Figure 3.6.1 Gracfet example for the illustration of context notion**

In figure 3.6.2 we represent the sets of variations that correspond to the analysis of possible evolutions starting from situation {2} that is reached by situation {1} in occurrence of event ↑m.

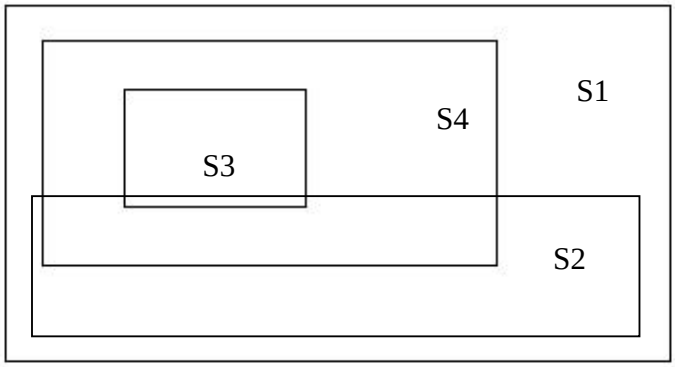


**Figure 3.6.2 Illustration of context notion**

We have four sets of input/internal variable variations:

- S1: set of input/internal variable variations for which the situation {2} is reached;
- S2: set of input/internal variable variations for which the transition t2 is cleared after reaching situation {2};
- S3: set of input/internal variable variations for which the transition t3 is cleared after reaching situation {2};
- S4: set of input/internal variable variations for which the transition t4 is cleared after reaching situation {2};

Formally, we define these four sets as follows:

$S1 = \{x \mid \bar{x} + \uparrow m = 1^*\}$;

$S2 = \{x \mid \bar{x} + \uparrow m \cdot a = 1^*\}$;

$S3 = \{x \mid \bar{x} + \uparrow m \cdot b \cdot c = 1^*\}$;

$S4 = \{x \mid \bar{x} + \uparrow m \cdot b = 1^*\}$;

Where 1* is the neutral element of AND operator in II.

In the following table we show the several contexts that we have defined giving their expression in II and the relating sets of input variations, starting from the reached situation {2}.

| Context | II expression | Represented set |
|---|---|---|
| Residual context of Sit {2} | $\uparrow m$ | S1 |
| Stability context of Sit {2} | $\uparrow m \cdot \overline{(a + b)}$ | S1 – (S2 ∪ S3 ∪ S4) |
| Minimal context of t2 | $\uparrow m \cdot a$ | S2 |
| Maximal context of t2 | $\uparrow m \cdot a \cdot \bar{b}$ | S2 – (S3 ∪ S4) |
| Minimal context of t3 | $\uparrow m \cdot b \cdot c$ | S3 |
| Maximal context of t3 | 0 | ø |
| Minimal context of t4 | $\uparrow m \cdot b$ | S4 |
| Maximal context of t4 | $\uparrow m \cdot \bar{a} \cdot b \cdot \bar{c}$ | S4 – (S2 ∪ S3) |
| Minimal context of {t2, t3} | $\uparrow m \cdot a \cdot b \cdot c$ | S2 ∩ S3 |
| Maximal context of {t2, t3} | 0 | ø |
| Minimal context of {t2, t4} | $\uparrow m \cdot a \cdot b$ | S2 ∩ S4 |
| Maximal context of {t2, t4} | $\uparrow m \cdot a \cdot b \cdot \bar{c}$ | (S2 ∩ S4) – S3 |
| Minimal context of {t3, t4} | $\uparrow m \cdot b \cdot c$ | S3 ∩ S4 |
| Maximal context of {t3, t4} | $\uparrow m \cdot \bar{a} \cdot b \cdot c$ | (S3 ∩ S4) – S2 |
| Minimal context of {t2, t3, t4} | $\uparrow m \cdot a \cdot b \cdot c$ | S2 ∩ S3 ∩ S4 |
| Maximal context of {t2, t3, t4} | $\uparrow m \cdot a \cdot b \cdot c$ | S2 ∩ S3 ∩ S4 |

**Table 3.6.1 Context illustration**

## 3.6.2 Timed context

As we define a context for not timed expressions, we associate separately a context for timed expressions.

### *17. Timed context*

*We call timed context the II combinatorial expressions that we associate with timed evolutions.*

Each timed context characterizes a precise set of timed variable variations and input and internal variable conditions.

We associate with each reached configuration Cfg (after a timed evolution) a timed residual context and a timed stability context.

### *18. Timed residual context*

*A timed residual context TCres describes the set of timed condition variations and input and internal variable conditions that allows reaching this configuration.*

### *19. Timed stability context*

*A timed stability context TCsta is the part of timed residual context for which the reached configuration is stable.*

We associate with each possible Temp set variation a timed minimal context and a timed maximal context.

### *20. Timed minimal context*

*A timed minimal context TCmin describes the set of timed expression variations and input and internal variable conditions that allows a timed evolution (timed evolution necessary condition).*

### *21. Timed maximal context*

*A timed maximal context TCmax is the part of the timed minimal context for which only this timed evolution is possible (timed evolution necessary and sufficient condition).*

We associate with each timed evolution a timed evolution context.

*22. Timed evolution context*

*A timed evolution context TCevo describes the set of timed expression variations and input and internal variable conditions that allows the occurrence of this timed evolution.*

## Example

In order to show these definitions we use the example in figure 3.5.1. Let consider the configuration 1:{1,2},{},{}, the set of waiting temporizations is: Wtemp = {1sec/X1,1sec/X2}. In figure 3.6.3 we represent the sets of variations that correspond to the analysis of all possible timed evolutions starting from the analyzed configuration.

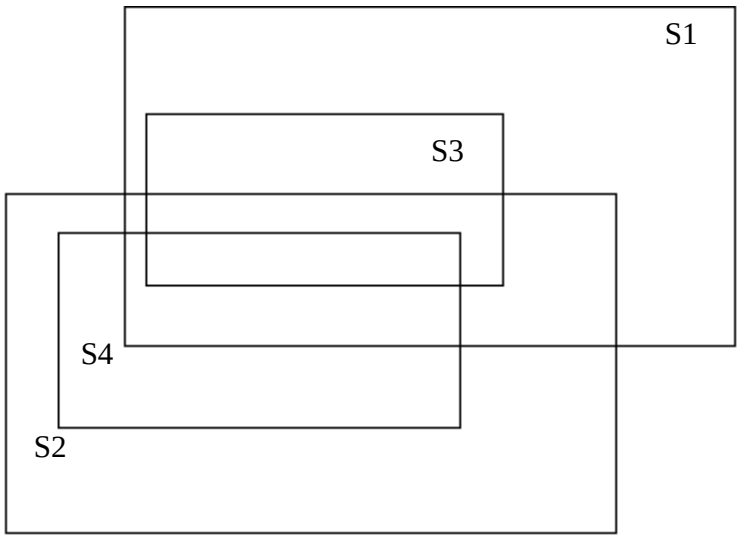


**Figure 3.6.3 Illustration of timed context notion**

We have four sets of timed expression variations and input and internal variables conditions:

- S1: timed expression variations set for which the timed expression {1sec/X1} is added to Temp;
- S2: timed expression variations set for which the timed expression {1sec/X2} is added to Temp;
- S3: timed expression variations set and input conditions for which there are not other possible evolutions after timed evolution caused by clock $h_1$;

- S4: timed expression variations set and input conditions for which there are not other possible evolutions after timed evolution caused by clock $h_2$.

Let us define the following configurations: 1:{1,2},{1sec/X1},{}, 2:{1,2},{1sec/X2},{}, 3:{1,2},{1sec/X1,1sec/X2,},{}.

In the following table we show the several contexts that we have defined giving their expression in II and the relating set of timed expressions variations and input and internal variables conditions.

| **Timed context** | **II expression** | **Represented set** |
|---|---|---|
| Timed minimal context of Cfg 2 | 1sec/X1 | S1 |
| Timed maximal context of Cfg 2 | $1sec/X1 \cdot \overline{1sec/\ X2}$ | S1 – S2 |
| Timed minimal context of Cfg 3 | 1sec/X2 | S2 |
| Timed maximal context of Cfg 3 | $\overline{1sec/\ X1} \cdot 1sec/X2$ | S2 – S1 |
| Timed minimal context of Cfg 4 | $1sec/X1 \cdot 1sec/X2$ | S1 ∩ S2 |
| Timed maximal context of Cfg 4 | $1sec/X1 \cdot 1sec/X2$ | S1 ∩ S2 |
| Timed residual context of Cfg 2 | $1sec/X1 \cdot \overline{1sec/\ X2}$ | S1 – S2 |
| Timed stability context of Cfg 2 | $\bar{b} \cdot 1sec/X1 \cdot \overline{1sec/\ X2}$ | S3 – S2 |
| Timed residual context of Cfg 3 | $\overline{1sec/\ X1} \cdot 1sec/X2$ | S2 – S1 |
| Timed stability context of Cfg 3 | $\bar{a} \cdot \overline{1sec/\ X1} \cdot 1sec/X2$ | S4 – S1 |
| Timed residual context of Cfg 4 | $1sec/X1 \cdot 1sec/X2$ | S1 ∩ S2 |
| Timed stability context of Cfg 4 | $\bar{a} \cdot \bar{b} \cdot 1sec/X1 \cdot 1sec/X2$ | S3 ∩ S4 |

**Table 3.6.2 Timed context illustration**

# 3.7 The Tree of Accessible Configurations

In order to build the Graph of Accessible Stable Configurations a grafcet behavior study at the internal time scale is necessary.

The Tree of Accessible Configurations (TAC) is a representation of a grafcet behavior at the internal time scale. Starting from a given stable configuration the TAC gives all possible evolutions that we can have in a grafcet in occurrence of all possible timed and input events.

In the TAC we have four main elements: root, internal nodes, leafs and directed links.

## 3.7.1 The root

The root is the starting stable configuration that we want to analyze. The root is characterized by:

- the configuration description Cfg: Sit, Temp, Out;
- the invariant condition Inv(Cfg);
- the set of waiting temporizations Wtemp;
- the set of enabled transitions EnT.

Starting from these parameters we can calculate a set of parameters that we use for TAC construction.

For each transition $t_i$ in EnT we evaluate:

***The actual transition condition (for a root)****: the actual transition condition cond'($t_i$) is given by the transition condition cond($t_i$) after application of the following replacements: all timed expressions $d_m/X_n$ are replaced with theirs values (if $d_m/X_n \in$ Temp it is replaced with true, else it is replaced with false), all step activation variables $X_n$ are replaced with theirs values (if $X_n \in$ Sit it is replaced with true, else it is replaced with false), all falling and rising edges of step activation variables $X_n$ are replaced with false.*

Then we can evaluate:

***The set of clearable transitions (for a root)****: the set of clearable transitions CT is composed of all enabled transitions $t_i$ for which the actual transition condition cond'($t_i$) is not false.*

The following parameter defines all possible not timed evolutions.

***The simultaneous clearable transitions sets (for a root):*** *the simultaneous clearable transitions sets SCT, are the sets of transitions that can be fired simultaneously. These sets are given by sets of clearable transitions for which the resulting maximal contexts are not false and they are characterized by an evolution context Cevo. We calculate as follows the Cevo:*

$$\mathbf{Cevo(SCT_i) = Cmax(SCT_i) \cdot \overline{\sum_{m,n} d_m / X_n}}$$

$$\mathbf{where\ d_m/X_n \in Wtemp}$$

The following parameter defines all possible timed evolutions.

***The simultaneous timed evolutions sets****: the simultaneous timed evolutions sets STevo, are the sets of timed evolutions that can be performed simultaneously. These sets are given by sets of timed evolutions for which the resulting timed maximal contexts are not false and they are characterized by a timed evolution context TCevo. We evaluate as follows the TCevo:*

$$\mathbf{TCevo(STevo_i) = TCmax(STevo_i) \cdot \overline{\sum_j Cevo'(SCT_j)}}$$

*Where Cevo'($SCT_j$) is the evolution context Cevo($SCT_j$) after replacement of all timed expressions with theirs values.*

### 3.7.2 Internal nodes

Internal nodes represent generic reached configurations. The internal nodes are characterized by:

- the configuration description Cfg: Sit, Temp, Out;
- the upstream configuration UpCfg;
- the (timed) residual context (T)Cres, that is given by the (timed) evolution context (T)Cevo that allows to reach this configuration;
- the sequence of transitions sets SeqT;
- the sequence of clock resets sets SeqR;
- the set of enabled transitions EnT.

All evolutions starting from a reached configuration are performed at the internal time scale, then there are not possible input or timed events. The first consequence is that at this level for each transition condition all timed expressions must be replaced with their values and all input rising and falling edges are replaced with false.

We can calculate a new set of parameters that we use for the following step of TAC

construction.

The first set of parameters that we have to evaluate is the set of internal events. We have two sets of internal events.

***The set of step falling edges****: the set of step falling edges FS is the set of steps that are deactivated in the analyzed evolution. We evaluate this set as follows:*

$$\mathbf{FS = \forall\, X_n \mid (X_n \in Sit(UpCfg)) \cap (X_n \notin Sit(Cfg))}$$

***The set of step rising edges****: the set of step rising edges RS is the set of steps that are activated in the analyzed evolution. We evaluate this set as follows:*

$$\mathbf{RS = \forall\, X_n \mid (X_n \in Sit(Cfg)) \cap (X_n \notin Sit(UpCfg))}$$

Then we have to establish which transitions can be fired. In order to perform this operation we have to transform the (timed) residual context and all transition conditions as follows.

***The actual residual context:*** *the actual residual context Cres' is given by the (timed) residual context (T)Cres after application of the following replacements: each falling edge of input and timed expressions is replaced with the negation of the expression and each rising edge of input and timed expressions is replaced with the expression.*

For each transition $t_i$ in EnT we evaluate:

***The actual transition condition (for an internal node)**: the actual transition condition cond'($t_i$) is given by:*

$$\mathbf{cond'(t_i) = cond(t_i) \cdot Cres'}$$

*Where in cond'($t_i$) the following replacements are performed: all timed expressions $d_m/X_n$ are replaced with theirs values, all step activation variables $X_n$ are replaced with theirs values, each step falling edge $\downarrow X_n$ is replaced with true if $X_n \in FS$ and with false otherwise, each step rising edge $\uparrow X_n$ is replaced with true if $X_n \in RS$ and with false otherwise.*

Then we can evaluate:

***The set of clearable transitions (for an internal node)**: the set of clearable transitions CT is composed of all enabled transitions $t_i$ for which the actual transition condition (for the internal node) cond'($t_i$) is not false.*

The following parameter defines all possible evolutions at the internal time scale.

***The simultaneous clearable transitions sets (for an internal node)**: the simultaneous clearable transitions sets SCT, are the sets of transitions that can be fired simultaneously. These sets are given by sets of clearable transitions for which the resulting maximal contexts are not false and they are characterized by an evolution context Cevo. We calculate as follows each evolution context:*

$$\mathbf{Cevo(SCT_i) = Cmax(SCT_i) \cdot Cres}$$

Then we evaluate the stability context Csta:

$$\mathbf{Csta = Cres \cdot \overline{\sum_j C_{max}(SCT_j)}}$$

If the stability context Csta is not false we create a fictitious “stability” evolution that is characterized by the evolution context:

$$\mathbf{Cevo(Stab) = Csta}$$

The obtained configuration is marked as leaf and at this level the emitted outputs are evaluated.

### 3.7.3 Leafs

The leafs represent stable configurations. A leaf is characterized by:

- the configuration description Cfg: Sit, Temp, Out;
- the stability context Csta;
- the sequence of transitions sets SeqT that allows reaching the configuration;
- the sequence of clock resets sets SeqR met in all internal evolutions;

### 3.7.4 Directed links

A direct link connects a root to e node, a node to a node or a node to a leaf. Each directed link is characterized by:

- the upstream node UpNode;
- the downstream node DnNode;
- the (timed) evolution context (T)Cevo;
- the sequence of transitions sets SeqT;
- the sequence of sets of clock resets SeqR.

We evaluate the set SeqT as follows:

$$\mathbf{SeqT = <SeqT, T>}$$

Where SeqT is the sequence of transitions sets evaluated until the given evolution and T is the set of cleared transition associated with the given evolution.

We evaluate the set SeqR as follows:

$$\mathbf{SeqR = <SeqR, R>}$$

Where SeqR is the sequence of sets of clock resets evaluated until the given evolution and R is the set of clock resets performed in the given evolution.
The set R is evaluated as follows:

**for each step $X_i$ in Sit(DnCfg):**

**{if ($X_i \notin$ Sit(UpCfg))**

**{if (a clock $h_i$ is associated with $X_i$) then $R = R \cup h_i$; }}**

## 3.7.5 TAC construction

For the TAC construction we use an algorithm. Before to show this algorithm let us define all used variables.

*Nodes* is the set of unexplored nodes, *Tnodes* is the set of TAC nodes, *UpNode* is an upstream node, *DnNode* is a downstream node, *Root* is the tree root, *UpCfg* is an upstream configuration, *DnCfg* is a downstream configuration, *An_cfg* is the analyzed configuration.

The set *links*, associated with nodes, contains the information about all possible evolutions that can be performed at the internal time scale. Each element of links is a set {UpNode,DnNode,Cevo,SeqT,SeqR}.

The set *Slinks*, associated with stable configurations, contains the information about all possible evolutions that can be performed at the external time scale. Each element of Slinks is a set {UpCfg,DnCfg,Cevo,SeqT,SeqR}.

A tree structure is individuated by the Root, the Tnodes set, the Leafs set the links set and the Slink set.

**TAC construction algorithm (pseudo-code)**

**Function BuildTree(An_cfg,grafcet) {**

*// Association of a root to the analyzed configuration with a function that evaluate all root parameters*

**Root = new root(An_cfg);**

**evaluate all SCT(Root);**

**evaluate all STevo(Root);**

**for ∀ SCT {** *// Evaluation of all possible not timed evolutions*

**evaluate(DnNode,Cevo,T,R);** *// Evaluation of all evolution parameters*

**links = links ∪ {Root,DnNode,Cevo,T,R};** *// Add the internal time scale evolution*

**Nodes = Nodes ∪ DnNode;** *// Add the reached node* **}**

**for ∀ STevo{** *// Evaluation of all possible timed evolutions*

**evaluate(DnNode,Cevo,,R);**

**links = links ∪ {root,DnNode,Cevo,,R};**

**Nodes = Nodes ∪ DnNode;}**

**for ∀ n ∈ Nodes{** *//Analysis of each unexplored node*

**Nodes = Nodes - n;** *// delete the node n from the set of unexplored nodes*

**evaluate all SCT(n);** *// Evaluation of all possible not timed evolutions starting from n*

**for ∀ SCT{**

**evaluate(DnNode,Cevo,T,R);**

**if(T∉ SeqT) {** *// Check if total instability condition is verified*

**SeqT = SeqT(n) ∪ T;**

**SeqR = SeqR(n) ∪ R;**

**links = links ∪ {n,DnNode,Cevo,SeqT,SeqR};**

**Tnodes = Tnodes ∪ n;** *// add n to the Tree of Accessible Configurations*

**Nodes = Nodes ∪ DnNode;** *//add DnNode to the unexplored nodes* **}**

**else{** *// If there is a totally instable situation the error is reported*

**SeqT = SeqT(n) ∪ T;**

**SeqR = SeqR(n) ∪ R;**

**links = links ∪ {n,ERROR,Cevo,SeqT,SeqR};**

```
                TNodes = TNodes ∪ ERROR; }
            if(Csta(n)) { // If the stability context is not false the node is a leaf
                out_evaluation(n); //evaluation of emitted outputs
                L = new leaf(n); // creation of a leaf with the node n
                links = links ∪ {n,L,Csta,SeqT,SeqR}; //add the stability evolution
                leaf = leafs ∪ L; // add the leaf to the set of leafs
                //add an evolution between root configuration and leaf configuration:
                Slinks = Slinks ∪ {cfg(Root),cfg(L),Csta,SeqT,SeqR};
                }
            }
Tree = {Root,Tnodes,Leafs,links,Slinks}
Return Tree;
}
```

### 3.7.6 Individuation of totally instable situations

In paragraph 2.10.2 we introduced the criterion of total instability. The construction of the TAC allows us to find totally instable situations.

> ***Individuation rule:*** *we have a totally instable situation if, for a given evolution, we meet twice the same set of transitions.*

We associate with this kind of evolution a reached node that we call ERROR, this reached node allows the analyst to individuate totally instable situations.

### 3.7.7 A particular case: a leaf configuration is equal to the root configuration

There are particular cases in which a leaf of the tree coincides with the root of the tree. This is not an instable situation, but anyway this is a situation that requires attention, because we have an evolution that leads to the starting configuration, then, potentially, we have a useless evolution.

## 3.8 The GASC construction

The construction of the GASC is performed with the application of algorithm reported in the paragraph 3.8.1;

### 3.8.1 GASC construction algorithm

Before to show this algorithm let us define all used variables.

*UScfg* is the set of unexplored stable configurations, *Loc* (Locations) is the set of GASC pairs configuration/invariant, *Glink* is the set of GASC evolutions.

We define the initial configuration as 1:{Sit1},{},{Out1}, where Sit1 is the set of initial steps and Out1 is the set of initial activated outputs.

**GASC construction algorithm (pseudo-code)**

```
Function BuildGASC(grafcet) {
UScfg = cfg1;
for ∀ An_cfg ∈ UScfg { //An_cfg is the analyzed configuration
        UScfg = UScfg – An_cfg; //Delete the configuration from the set of unexplored configurations
        Tree = BuildTree(An_cfg,grafcet); //Construction of the TAC
        //add all unexplored stable configurations to the set UScfg:
        for ∀ L ∈ Leafs(Tree)
                if (cfg(L) ∉ EScfg) UScfg = UScfg ∪ cfg(n);
        Invariant = Tree.Inv(Root); //Evaluate the invariant of the Tree root
        //Add the analyzed configuration and the relating invariant to the GASC:
        Loc = Loc ∪ {An_cfg,Invariant}
        //add all evolutions to the evolution set Glinks:
        for ∀ lnk ∈ Slinks(Tree)
                Glinks = Clinks ∪ lnk;
        }
GASC = {Loc,Glinks}
//This function evaluate the initial configuration and all possible starting evolutions:
GASC = GASC.InitialConfiguration(grafcet);
return GASC;
}
```

### 3.8.2 Evaluation of the initial configuration

The initial grafcet situation can be different from the situation in which only initial steps are activated. Let us consider the example in figure 1.2.1: if at the starting time the condition pump1fault is true, the initial situation is {11,20,31,40,50}.

In order to consider this kind of situations we decided to introduce a fictitious configuration 0: INIT, {}, {}. Starting from this initial configuration, all not timed evolutions relating to the configuration 1 are analyzed and if it is necessary initial evolutions are associated. The way to evaluate the initial evolutions is the following:

- indicate with 0 the new initial configuration;
- evaluate Inv'(1) as Inv(1) where all time depending expressions are replaced with false;
- link configurations 0 and 1 with an evolution characterized by:
  - an empty transition set;
  - an evolution context given by the resulting condition invariant Cevo(INIT) = Inv'(1);
  - a set of clock reset operations depending on Sit(1) steps;
- for each not timed evolution of 1 make the following replacements in Cevo($SCT_i$):
  - replace all input edges with false;
  - replace all time depending expressions with false;
- if the resulting Cevo'($SCT_i$) is not false, associate an evolution that links 0 with the arrival configuration relating to $SCT_i$ characterized by the $SCT_i$ sequence of transitions sets SeqT, the evolution condition Cevo'($SCT_i$) and the $SCT_i$ sequence of reset operations SeqR.

### 3.8.3 Dealing with TON and TOF input conditions

A representation of TON and TOF on input variables is impossible by using timed automata. The only way to deal with this kind of problems is a grafcet transformation before its GASC generation. We translate the grafcet in a form that allows the analysis by means of GASC.

## TON on input conditions

In figure 3.8.1 Cond(d/exp) is a generic Boolean expression that depends on expression d/exp, where d is a time constant and exp is a Boolean expression of input variables.

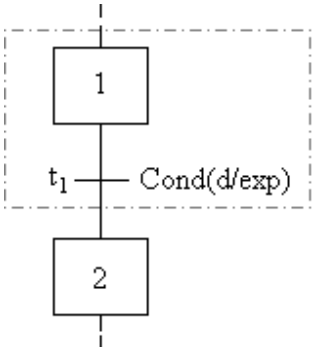


**Figure 3.8.1 A grafcet with a TON on input condition**

The solution is the construction of a grafcet with a structure like figure 3.8.2: this grafcet "simulates" expression d/exp behavior:

- if step F is activated the condition d/exp is false;
- if step W is activated the condition exp is true for less than a time d so d/exp is false;
- if step T is activated the condition exp is true for at least a time d and the condition d/exp is true.

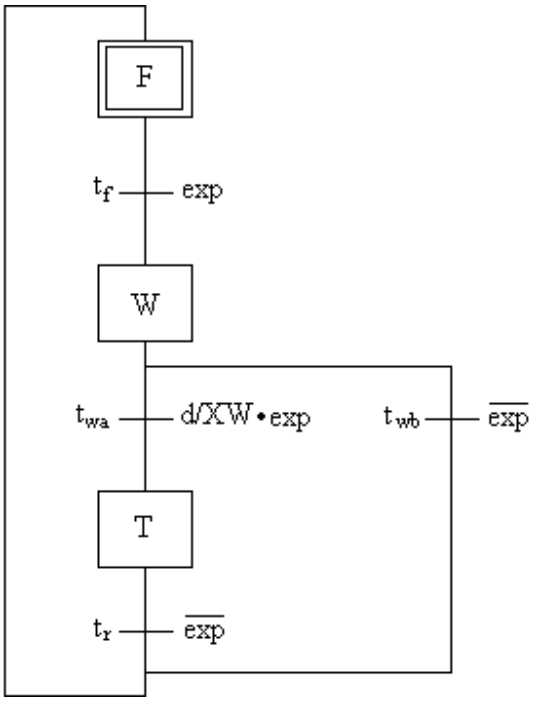


**Figure 3.8.2 The grafcet solution for TON problem**

In the starting grafcet all expressions d/exp are replaced with:

$$\mathbf{d/exp = XT \cdot exp + XW \cdot exp \cdot (d/XW)}$$

## TOF on input conditions

In figure 3.8.3 Cond(exp/d) is a generic Boolean expression that depends on expression exp/d, where d is a time constant and exp is a Boolean expression of input variables.

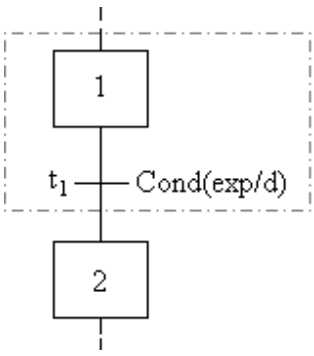


**Figure 3.8.3 A grafcet with a TOF on an input condition**

The solution is the construction of a grafcet with a structure like figure 3.8.4: this grafcet "simulates" expression exp/d behavior:

- if step F is active the condition exp/d is false;
- if step T is active the condition exp is true and then condition exp/d too is true;
- if step W is active the condition exp is false for less than a time d and the condition exp/d is still true.

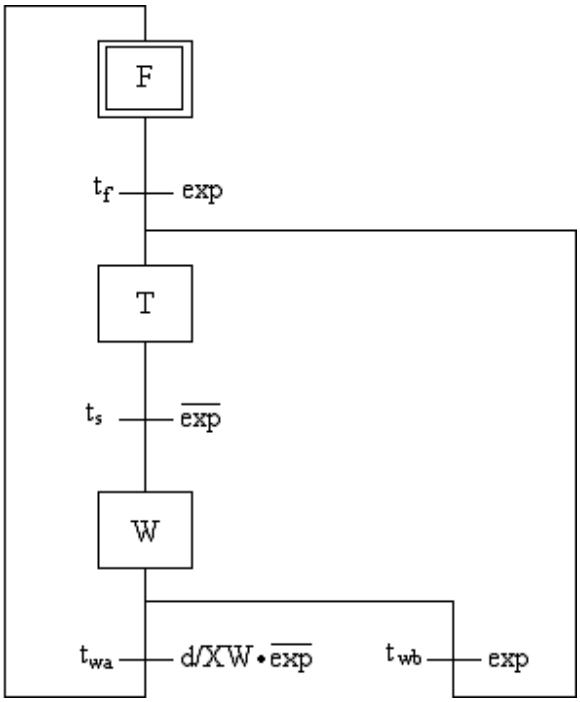


**Figure 3.8.4 The grafcet solution for TOF problem**

In the starting grafcet all expressions exp/d are replaced with:

$$\mathbf{exp/d = exp + XT + XW \cdot \overline{d / XW}}$$

## 3.9 GASC formal definition

We can give a formal definition for a GASC.

*23. Graph of Accessible Stable Configurations*

*A GASC is a tuple (Cfg,$cfg_0$,H,TEXP,EXP,SEQT,SEQR,INV) where:*

- *Cfg is a finite set of control states (configurations);*
- *$cfg_0 \in Cfg$ is the initial configuration;*
- *H is a finite set of clocks $h_i$;*
- *TEXP is a set of timed expressions $d_n/X_n$;*
- *EXP is a set of Boolean expressions on grafcet input variables;*
- *SEQT is a set of SeqT;*
- *SEQR is a set of SeqR;*
- *INV is a set of invariants Inv(cfg);*
- *$E \subseteq Cfg \times SEQT \times TEXP \times EXP \times SEQR \times Cfg$ is a finite set of evolutions;*
- *$evo = \langle cfg, SeqT, Texp, exp, SeqR, cfg' \rangle \in E$ represents an evolution from cfg to cfg'.*

  *We also write $cfg \xrightarrow{SeqT,Texp\cdot exp,SeqR} cfg'$ for evo;*

## 3.10 An example of application

We conclude this chapter with the application of the algorithm to the grafcet portion in figure 3.10.1.

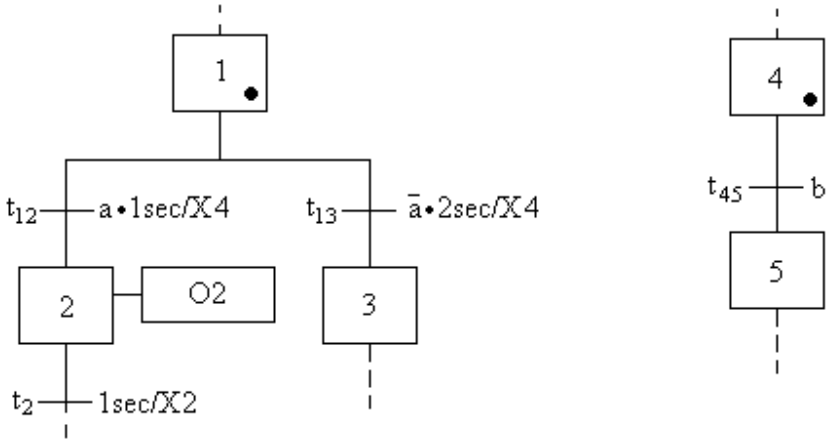


**Figure 3.10.1 The analyzed grafcet portion**

Let us consider the stable configuration 1: {1,4},{},{}. We have to evaluate the Tree of Accessible Configurations of configuration 1. The obtained TAC and GASC are shown in figure 3.10.2 and 3.10.3.

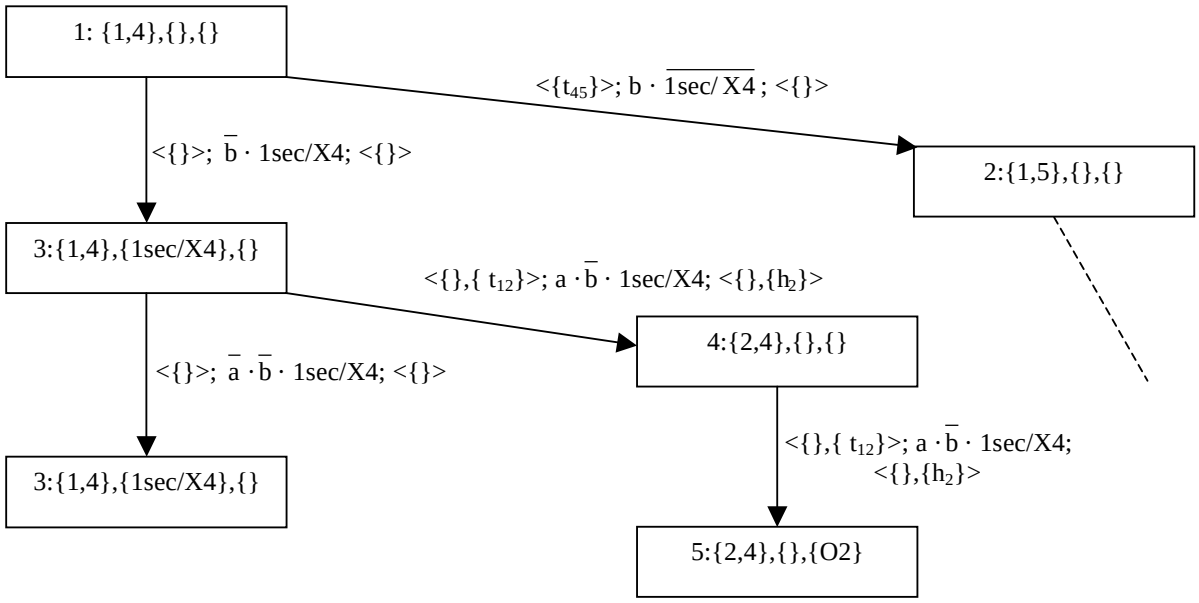


**Figure 3.10.2 TAC graphical representation**

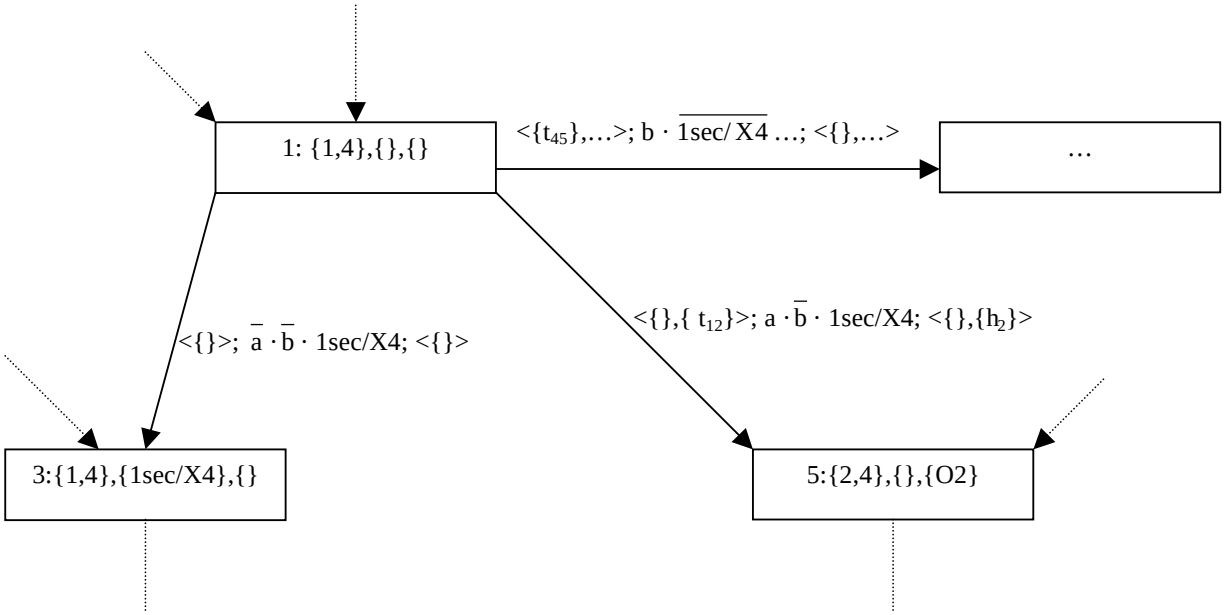


**Figure 3.10.3 GASC graphical representation**

In order to have a more simple representation the invariants are not reported, but normally they must be indicated.

Now we show how the Tree and the Graph have been evaluated.

## 3.10.1 Root analysis

Root description:

$$\text{Sit} = \{1,4\};\ \text{Temp} = \{\};\ \text{Out} = \{\};\ \text{Wtemp} = \{1\text{sec/X4}\};\ \text{EnT} = \{t_{12}, t_{13}, t_{45}\}$$

Configuration invariants:

$$\text{Inv(Sit)} = \bar{b},\quad \text{Inv(Temp)} = \overline{1\text{sec/ X4}} \cdot \overline{2\text{sec/ X4}},\quad \text{Inv(Out)} = 1$$

$$\text{Inv(Cfg)} = \bar{b} \cdot \overline{1\text{sec/ X4}} \cdot \overline{2\text{sec/ X4}}$$

Actual transition conditions:

$$\text{cond'}(t_{12}) = 0;\ \text{cond'}(t_{13}) = 0;\ \text{cond'}(t_{45}) = b$$

Clearable transitions set:

$$\text{CT} = \{t_{45}\}$$

Evaluation of all not timed evolutions:

The only set of transitions for which the maximal context is not false is $SCT_1 = \{t_{45}\}$

$$Cmax(SCT_1) = b$$

$$Cevo(SCT_1) = b \cdot \overline{1sec/X4}$$

Evaluation of all timed evolutions:

The only possible timed evolution is $STevo_1 = \{1sec/X4\}$

$$TCmax(STevo_1) = 1sec/X4$$

$$TCevo(STevo_1) = 1sec/X4 \cdot \bar{b}$$

## 3.10.2 Direct links starting from the root

Not timed evolutions:

$SCT_1$:

UpNode = Root = 1:{1,4},{},{}

$$Cevo = b \cdot \overline{1sec/X4}$$

SeqT = <{$t_{45}$}>; SeqR = <{}>

DnNode: (Cfg = 2:{1,5},{},{}; UpCfg = 1:{1,4},{},{}; Cres = $b \cdot \overline{1sec/X4}$ ; SeqT = <{$t_{45}$}>; SeqR = <{}>)

Timed evolutions:

$STevo_1$:

UpNode = Root = 1:{1,4},{},{}

$$TCevo = 1sec/X4 \cdot \bar{b}$$

SeqT = <{}>; SeqR = <{}>

DnNode: ( Cfg = 3:{1,4},{1sec/X4},{}; UpCfg = 1:{1,4},{},{}; TCres = $1sec/X4 \cdot \bar{b}$ ; SeqT = <{}>; SeqR = <{}>)

## 3.10.3 Analysis of an internal node

Analyzed node:

(Cfg = 3:{1,4},{1sec/X4},{}; UpCfg = 1:{1,4},{},{}; TCres = $1sec/X4 \cdot \bar{b}$ ; SeqT = <{}>; SeqR = <{}>)

$$EnT = \{t_{12}, t_{13}, t_{45}\}$$

Evaluation of node parameters:

$$FS = \{\}, RS = \{\}$$

$$Cres' = 1sec/X4 \cdot \bar{b}$$

$$cond'(t_{12}) = a \cdot 1sec/X4 \cdot 1sec/X4 \cdot \bar{b} = a \cdot \bar{b}$$

$$cond'(t_{13}) = \bar{a} \cdot 2sec/X4 \cdot 1sec/X4 \cdot \bar{b} = 0$$

$$cond'(t_{45}) = b \cdot 1sec/X4 \cdot \bar{b} = 0$$

$$CT = \{t_{12}\}$$

Evaluation of all possible evolutions at the internal time scale:

The only set of transitions for which the maximal context is not false is $SCT_1 = \{t_{12}\}$

$$Cmax(SCT_1) = a \cdot \bar{b}$$

$$Cevo(SCT_1) = a \cdot \bar{b} \cdot 1sec/X4 \cdot \bar{b} = a \cdot \bar{b} \cdot 1sec/X4$$

Evaluation of the stability context:

$$Csta = 1sec/X4 \cdot \bar{b} \cdot \overline{\overline{a \cdot \bar{b}}} = 1sec/X4 \cdot \bar{a} \cdot \bar{b}$$

## 3.10.4 Update the set of direct links

Evolutions:

$$SCT_1:$$

UpNode: (Cfg = 3:{1,4},{1sec/X4},{}; UpCfg = 1:{1,4},{},{}; TCres = $1sec/X4 \cdot \bar{b}$ ; SeqT = <{}>; SeqR = <{}>)

$$Cevo = a \cdot \bar{b} \cdot 1sec/X4$$

$$SeqT = <\{\},\{t_{12}\}>, SeqR = <\{\},\{h_2\}>$$

DnNode:( Cfg = 4:{2,4},{},{}; UpCfg = 3:{1,4},{1sec/X4},{};Cres = $a \cdot \bar{b} \cdot 1sec/X4$; SeqT = <{},{ $t_{12}$}>; SeqR = <{},{$h_2$}>)

Stability evolutions:

Leaf = (Cfg = 3:{1,4},{1sec/X4},{}; TCsta = $1sec/X4 \cdot \bar{a} \cdot \bar{b}$ ; SeqT = <{}>; SeqR = <{}>)

UpNode = (Cfg = 3:{1,4},{1sec/X4},{}; UpCfg = 1:{1,4},{},{}; TCres = $1sec/X4 \cdot \bar{b}$ ; SeqT = <{}>; SeqR = <{}>)

$$TCevo = 1sec/X4 \cdot \bar{a} \cdot \bar{b}$$

$$SeqT = <\{\}>, SeqR = <\{\}>$$

DnNode: Leaf

### 3.10.5 Analysis of a second internal node

Analyzed node:

$$(Cfg = 4{:}\{2,4\},\{\},\{\};UpCfg = 3{:}\{1,4\},\{1sec/X4\},\{\};\ Cres = a\cdot\bar{b}\cdot 1sec/X4;\ SeqT = <\{\},\{t_{12}\}>,\ SeqR = <\{\},\{h_2\}>)$$

$$EnT = \{t_2,t_{45}\}$$

Evaluation of node parameters:

$$FS = \{X2\},\ RS = \{X1\}$$

$$Cres' = a\cdot\bar{b}\cdot 1sec/X4$$

$$cond'(t_2) = 1sec/X2\cdot a\cdot\bar{b}\cdot 1sec/X4 = 0$$

$$cond'(t_{45}) = b\cdot a\cdot\bar{b}\cdot 1sec/X4 = 0$$

$$CT = \{\}$$

There are not possible evolutions.

Evaluation of the stability context:

$$Csta = a\cdot\bar{b}\cdot 1sec/X4$$

Stability evolution:

$$Leaf = (Cfg = 5{:}\{2,4\},\{\},\{O2\};\ Csta = a\cdot\bar{b}\cdot 1sec/X4;\ SeqT = <\{\},\{t_{12}\}>;\ SeqR = <\{\},\{h_2\}>)$$

$$UpNode = (Cfg = 4{:}\{2,4\},\{\},\{\};\ UpCfg = 3{:}\{1,4\},\{1sec/X4\},\{\};\ Cevo = a\cdot\bar{b}\cdot 1sec/X4;\ SeqT = <\{\},\{t_{12}\}>;\ SeqR = <\{\},\{h_2\}>)$$

$$Cevo = a\cdot\bar{b}\cdot 1sec/X4$$

$$SeqT = <\{\},\{t_{12}\}>,\ SeqR = <\{\},\{h_2\}>$$

DnNode: Leaf

# Chapter 4

# The Equivalent Timed Automaton

## Contents

*The first part of this chapter is devoted to introduce all main theoretical notions about timed transition systems and timed automata, then a particular means for timed automata analysis called region graph is presented.*

*In the second part an algorithm that allows to translate a Graph of Accessible Stable Configurations into a corresponding equivalent timed automaton is presented with all results obtained in terms of study of reachability of a configuration.*

*The chapter is concluded with two examples: the first one illustrates how all introduced algorithms have to be applied, the second one illustrates how the obtained equivalent timed automaton can be analyzed with the support of UPPAAL.*

# 4.1 Introduction

It's simple to observe that the form of the GASC is very similar to the form of a timed automaton: we have a set of locations, each location is characterized by an invariant and the evolutions are caused by input and timed events.

This is not a casual result: we have carried out the GASC in order to obtain a GRAFCET representation as close as possible to timed automaton formalism. For example a more simple choose can be to use a representation that allows to stop clocks, but in this case the more similar formalism is the automaton chronometer class that presents undecidability problems.

# 4.2 Timed automata

Timed automata[1] have been proposed by R. Alur and D. Dill in the 1990s [Alu 90], [Alu 94] as a model for real-time systems. A timed automaton is a classical finite automaton which can manipulate clocks, evolving continuously and synchronously with absolute time. Each transition is labeled by a constraint over clock values (also called guard), which indicates when the transition can be fired, and a set of clocks to be reset when the transition is fired. Each location is constrained by an invariant, which restricts the possible values of the clocks for being in the state.

## 4.2.1 Some notation

Let $X$ be a finite set of clocks. A (clock) valuation $v$ over $X$ is a function $v : X \rightarrow \mathbb{R}$ which associates to each clock $x$ its value $v(x) \in \mathbb{R}$. We denote by $\mathbb{R}^X$ the set of clock valuations over $X$. Let be $\tau \in \mathbb{R}$, we write $v + \tau$ for the clock valuation associating with clock $x$ the value $v(x) + \tau$. If $r$ is a subset of $X$, $[r \leftarrow 0]v$ is the valuation $v'$ such that $v'(x) = 0$ if $x \in r$, and $v'(x) = v(x)$ otherwise. We write $C(X)$ for the set of clock constraints over $X$. We note by $C'(X)$ the restriction of $C(X)$ to

---

[1] [1] See [Alur 90], [Alur 94], [P.W.Kopke 95], [A.Puri 96], [A. Di Febbraro, A. Giua 02],

positive Boolean combinations only containing constraints of the form $x \leq c$ or $x < c$. We interpret clock constraints over clock valuations: a valuation $v$ satisfies the atomic constraint $x \bowtie c$ whenever $v(x) \bowtie c$. When a valuation $v$ satisfies a constraint $g$, we write $v \models g$.

### 4.2.2 Syntax

We define a timed automaton as follows:

*24. Timed automaton*

*A timed automaton is a tuple $(L, \ell_0, X, Inv, T, \Sigma)$ where:*

- *$L$ is a finite set of control states (locations);*
- *$\ell_0 \in L$ is the initial location;*
- *$X$ is a finite set of clocks;*
- *$T \subseteq L \times C(X) \times \Sigma \times 2^X \times L$ is a finite set of transitions;*
- *$e = \langle \ell, g, \sigma, r, \ell' \rangle \in T$ represents a transition from $\ell$ to $\ell'$, $g$ is the guard of $e$, $r$ is the set of clocks that is reset by $e$, and $\sigma$ is the action of $e$. We also write $\ell \xrightarrow{g,\sigma,r} \ell'$ for $e$;*
- *$Inv: L \rightarrow C'(X)$ associates with each location an invariant;*
- *$\Sigma$ is an alphabet of actions.*

### 4.2.3 Timed transition system, bisimulation and quotient

In this paragraph we give the main notions that help to understand how a reachability problem study on timed automata can be performed.

*25. State transition system*

*A state transition system (or transition system) is a tuple $T = (S, \Sigma, \rightarrow, S_0, S_F)$ where:*

- *$S$ is a set of states (possibly infinite);*
- *$\Sigma$ is an alphabet of actions;*
- *$\rightarrow \subseteq S \times \Sigma \times S$ is the transition relation: if $(s, \sigma, s') \in \rightarrow$, where $s, s' \in S$ and $\sigma \in \Sigma$, we write $s \xrightarrow{\sigma} s'$;*
- *$S_0$ is the set of initial states;*
- *$S_F$ is the set of final states (it can be omitted).*

## *26. Timed transition system*

*A timed transition system is a tuple $S = (S, \Sigma, \rightarrow, s_0)$ where:*

- *$S$ is a set of states (possibly infinite);*
- *$s_0 \in S$ is the initial state;*
- *$\Sigma$ is an alphabet of actions;*
- *$\rightarrow \subseteq S \times (\Sigma \cup \mathbb{R}) \times S$ is the transition relation. Moreover, the relation $\rightarrow$ satisfies the three following conditions:*

  *I* *if $s \xrightarrow{0} s'$, then $s = s'$;*

  *II* *if $s \xrightarrow{\tau} s'$ and $s' \xrightarrow{\tau'} s''$ with $\tau, \tau' \in \mathbb{R}$ then $s \xrightarrow{\tau+\tau'} s''$;*

  *III* *if $s \xrightarrow{\tau} s'$ with $\tau \in \mathbb{R}$ then for all $0 \leq \tau' \leq \tau$, there exists $s''$ such that $s \xrightarrow{\tau'} s''$ and $s'' \xrightarrow{\tau-\tau'} s'$.*

## *27. Bisimulation*

*Let $T = (S, \Sigma, \rightarrow, S_0, S_F)$ be a transition system. An equivalence relation $\sim \subseteq S \times S$ between the states of $T$ is a bisimulation if:*

- *$(s \sim \hat{s}) \wedge (s \in S_0) \quad \Rightarrow \quad (\hat{s} \in S_0)$;*
- *$(s \sim \hat{s}) \wedge (s \in S_F) \quad \Rightarrow \quad (\hat{s} \in S_F)$;*
- *$(s \sim \hat{s}) \wedge (s \xrightarrow{\sigma} s') \quad \Rightarrow \quad \exists s' \in S \mid \quad (\hat{s} \xrightarrow{\sigma} \hat{s}') \wedge (s' \sim \hat{s}')$.*

## *28. Predecessors set*

*Let $T = (S, \Sigma, \rightarrow, S_0, S_F)$ be a transition system. Given a subset $S' \subseteq S$ and a generator $\sigma$, we define the set of $\sigma$–predecessors of $S'$:*

$$Pre_\sigma(S') = \{ s \in S \mid (\exists s' \in S')\ s \xrightarrow{\sigma} s' \}$$

**Proposition 1.** Let $T = (S, \Sigma, \rightarrow, S_0, S_F)$ be a transition system. Necessary and sufficient condition so that $\sim \subseteq S \times S$ be a bisimulation is that the following conditions are verified:

- the set $S_0$ is the union of equivalence classes of $\sim$;

- the set $S_F$ is the union of equivalence classes of ~;
- for each equivalence class $\pi \in \Pi_{\sim}$ and for each $\sigma \in \Sigma$, the set $Pre_\sigma(\pi)$ is the union of equivalence classes of ~.

*29. Quotient*

*Let $T = (S,\Sigma,\rightarrow,S_0,S_F)$ be a transition system and let $\sim \subseteq S \times S$ be a bisimulation. We define quotient $T/\sim$ the transition system $T/\sim = (S/\sim,\Sigma,\rightarrow,S_0/\sim,S_F/\sim)$ where:*

- *The states set $S/\sim = \Pi_{\sim}$ coincides with the equivalence classes of ~;*
- *$S_0/\sim = \{\pi \in S/\sim \mid \pi \subseteq S_0\}$;*
- *$S_F/\sim = \{\pi \in S/\sim \mid \pi \subseteq S_F\}$;*
- *the transition $\rightarrow$ is defined as follows: $\forall \pi,\pi' \in S/\sim,\ \forall \sigma \in \Sigma \quad \pi \xrightarrow{\sigma} \pi'$ if $\pi \subseteq Pre_\sigma(\pi')$.*

We can underline two important results that allow solving our GASC reachability problem after a translation into a timed automaton.

**Result 1.** Let suppose that, in the quotient, $\pi'$ is reachable by $\pi$ by means of generator $\sigma$. We can say that in the original system starting from each state in $\pi$ it is possible to reach at least a state in $\pi'$ by means of generator $\sigma$.

**Result 2.** Let suppose that, in the quotient, $\pi'$ is not reachable by $\pi$ by means of generator $\sigma$. We can say that in the original system there are not states in $\pi$ from which it is possible to reach at least a state in $\pi'$ by means of generator $\sigma$.

If we can evaluate the quotient of a timed automaton, if each timed automaton location represents a GASC configuration and each automaton transition represents a GASC evolution, then we can evaluate all reachable configurations of the analyzed GASC.

### 4.2.4 Region graph

A state of a timed automaton is a pair $(\ell,v) \in L \times \mathbb{R}^X$ where $\ell$ is the location and $v$ is the clock valuation. The semantic of a timed automaton is given as a timed transition system with action transitions (labeled with elements of $\Sigma$) and delay transitions (labeled with real numbers representing the delay). Classically, an execution in a timed transition system TTS is a sequence

of consecutive transitions. A state $s \in S$ is said to be reachable in S if there exists an execution from $s_0$ to s.

### *30. Semantic of a timed automaton*

*Let $A = (L,\ell_0,X,Inv,T,\Sigma)$ be a timed automaton. The semantic of A is defined as the timed transition system $S_A = (S,s_0,\rightarrow,\Sigma)$ where:*

- $S = L \times \mathbb{R}^X$;
- $s_0 = (\ell_0,v_0)$ *where* $v_0(x) = 0$ *for each* $x \in X$;
- *the transition relation* $\rightarrow$ *is composed of:*

  *I* *action transitions:* $(\ell,v) \xrightarrow{\sigma} (\ell',v')$ *iff there exists* $\ell \xrightarrow{g,\sigma,r} \ell' \in T$ *such that* $v \vDash g$, $v' = [r \leftarrow 0]v$ *and* $v' \vDash Inv(\ell')$;

  *II* *delay transitions: if* $\tau \in \mathbb{R}$, $(\ell,v) \xrightarrow{\tau} (\ell,v+\tau)$ *iff* $v + \tau \vDash Inv(\ell)$.

Given a timed automaton, let us define for each clock $x_i$ a value $M_i$ that is the biggest integer that $x_i$ is confronted with in the guards. We can give the following definition.

### *31. Regions*

*Let consider the equivalence relation* $\approx \subseteq X \times X$ *between timed automaton continuous states: given two states* $x = (x_1, x_2,\ldots,x_n)$ *and* $x' = (x'_1, x'_2,\ldots,x'_n)$, *then* $x \approx x'$ *if:*

- $\forall i$ $\quad \lfloor x_i \rfloor = \lfloor x'_i \rfloor$ $\quad$ *or* $\quad (\lfloor x_i \rfloor > M_i) \wedge (\lfloor x'_i \rfloor > M_i)$;
- $\forall i \mid x_i \leq M_i$ $\quad \langle x_i \rangle = 0 \Leftrightarrow \langle x'_i \rangle = 0$;
- $\forall i,j \mid x_i \leq M_i,\ x_j \leq M_j$ $\quad \langle x_i \rangle = 0 \Leftrightarrow \langle x'_i \rangle = 0$;

*where* $\lfloor x \rfloor$ *is the integral part of* $x$ *and* $\langle x \rangle \in [0,1)$ *is its fractional part.*

**Proposition 2.** Let consider the equivalence relation $\approx \subseteq S \times S$ between the $S_A$ states defined as follows: given two states $(\ell,v)$ and $(\ell',v')$, then $(\ell,v) \approx (\ell',v')$ if $\ell \approx \ell'$ and $v \approx v'$. This relation is a finite bisimulation on S.

**Proposition 3.** The number of equivalence classes of relation ≈ defined on a timed transition system associated with an n-dimensional timed automaton is less or equal than $N=|L|\cdot(\prod_{i=1}^{n}(M_i+1))\cdot n!\cdot 2^n$.

A quotient of a timed automaton is also called region graph, practically a region graph is a finite automaton. In figure 4.2.1 an example of timed automaton and region graph is shown.

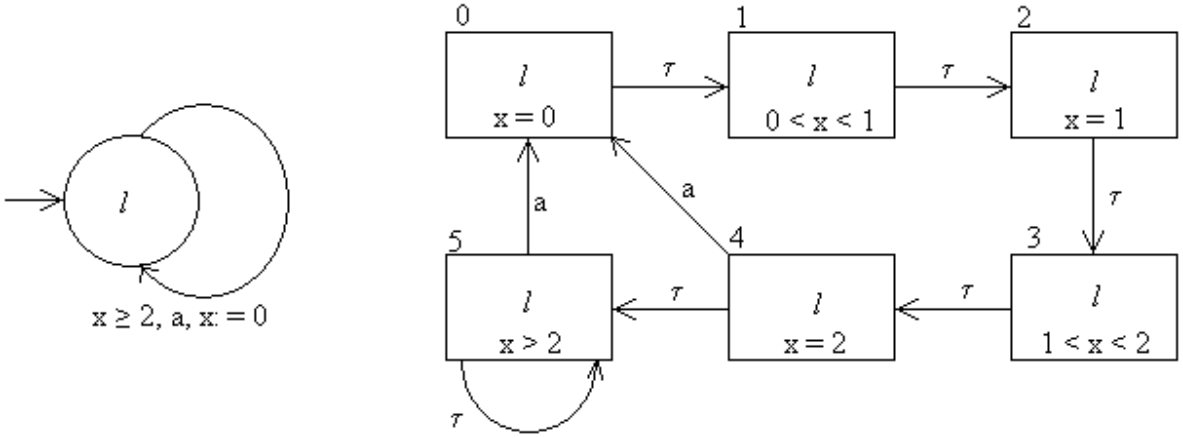


**Figure 4.2.1 An example of region graph**

# 4.3 Timed automata for GASC analysis

We translate the GASC into a timed automaton in order to solve a problem of configuration reachability. The translation of a grafcet model into a GASC implies to introduce some simplification: we deal with time at a logic level. As the time is dealt at a logic level it is obvious that we can't perform all possible controls on timed expressions, the result is that in the GASC there are certain configurations that is physically impossible to reach. The GASC gives a necessary condition of reachability: if a grafcet configuration is not reachable by its GASC, then the analyzed grafcet can't reach this configuration. Vice versa we can't say that if a configuration is reachable by the GASC then it is reachable by the analyzed grafcet. With the translation of the GASC into a timed automaton we can individuate all not physically reachable configurations and then delete them.

### 4.3.1 Timed automaton construction

Let us call $\ell_1$ the equivalent timed automaton initial location, we obtain the equivalent timed automaton with the following steps:

- for each GASC clock $h_i$ in H associate a timed automaton clock $x_i$ in X;
- associate the initial timed automaton location $\ell_1$ to the initial GASC configuration $cfg_1$, all clocks are reset in the initial location;
- for each configuration $cfg_i$ (with $i \neq 1$) in Cfg associate a timed automaton location $\ell_i$;
- for each Boolean expression $exp_i$ in EXP associate an action $\sigma$ in $\Sigma$;
- for each timed automaton location $\ell_i$ and GASC configuration $cfg_i$:
  - for each timed expression $\overline{d_m / X_n}$ in Inv(Temp) associate with the location $\ell_i$ an invariant $Inv_{ij}$: $x_n \leq d_m$;
- for each GASC evolution $evo_i$ $\langle cfg_i, SeqT_i, Texp_i, exp_i, SeqR_i, cfg_i' \rangle$ in E associate a timed automaton transition $e_i$ $\langle \ell_i, g_i, \sigma_i, r_i, \ell'_i \rangle$ in T where:
  - for each timed expression $d_m / X_n$ in $Texp_i$ associate with the guard $g_i$ a condition $x_n \geq d_m$, for each timed expression $\overline{d_m / X_n}$ in $Texp_i$ associate with the guard $g_i$ a condition $x_n < d_m$;
  - if there is at least an expression $d_m/X_n$ in $Texp_i$ then there are not associated actions ($\sigma_i$='') else the associated action $\sigma_i$ is $exp_i$;
  - for each GASC clock $h_j$ in $SeqR_i$ associate a timed automaton clock $x_j$ reset;

It's obvious that in certain situations the equivalent timed automaton is a nondeterministic timed automaton.

### 4.3.2 Analysis of the equivalent timed automaton and GASC simplification

The analysis of the equivalent timed automaton is performed by construction of the region graph. As there is a 1-1 correspondence between automaton locations and GASC configurations, if a given location is not reachable by timed automaton then the corresponding configuration is not physically reachable by the analyzed grafcet.

The GASC simplification is performed as follows: let Reach be the set of locations $\ell_i$ reachable in the timed automaton, for each configuration $cfg_i$ in Cfg if the corresponding location

$\ell_i$ is not in Reach then $cfg_i$ is deleted from the GASC.

### 4.3.3 Equivalent timed automaton complexity and analysis simplification

As a grafcet can be the representation of very complex systems, the analyzed model can be very complex and it can be characterized by a very large number of temporizations. The consequence is that the equivalent timed automaton can be characterized by a large number of clocks and then a very large number of clock regions. But it's obvious that for each location not all clocks and guard are relevant in order to evaluate the system behavior, then a technique of active-clock reduction can be useful.

A method that generalizes active-clock reduction is proposed in [G.Behrmann, P.Bouyer 03]. On this work, authors propose a location-based finite zone abstraction which computes an abstraction based on the relevant guards for a particular state of the model (as opposed to all guards), then they propose a location-based bisimulation. The result is a region graph simplification.

## 4.4 An example of application

In this paragraph we show a step by step example of algorithm application. The simplicity of this example allows to have an overall view of the results of this work.

Let consider the grafcet in figure 4.4.1.

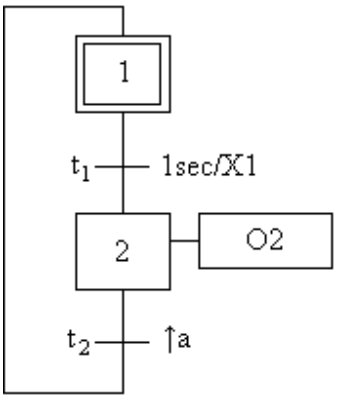


**Figure 4.4.1 A final example**

### 4.4.1 TAC construction

The initial configuration is 1: {1},{},{}

**Root 1:{1},{},{} analysis**

Root description:

$$Sit = \{1\}; Temp = \{\}; Out = \{\}; Wtemp = \{1sec/X1\}; EnT = \{t_1\}$$

Configuration invariants:

$$Inv(Sit) = 1, \quad Inv(Temp) = \overline{1sec/X1}, \quad Inv(Out) = 1$$

$$Inv(Cfg) = \overline{1sec/X1}$$

Actual transition conditions:

$$cond'(t_1) = 0;$$

Clearable transitions set:

$$CT = \{\}$$

There are not possible not timed evolutions.

Evaluation of all timed evolutions:

$$\text{The only possible timed evolution is } STevo_1 = \{1sec/X1\}$$

$$TCmax(STevo_1) = 1sec/X1$$

$$TCevo(STevo_1) = 1sec/X1$$

**Direct links starting from the root**

Timed evolutions:

$$STevo_1:$$

$$UpNode = Root = 1:\{1\},\{\},\{\}$$

$$TCevo = 1sec/X1$$

$$SeqT = <\{\}>; SeqR = <\{\}>$$

DnNode: N2 ( Cfg = 2:{1},{1sec/X1},{}; UpCfg = 1:{1},{},{}; TCres = 1sec/X1; SeqT = <{}>; SeqR = <{}>)

**Internal node N2 analysis**

Analyzed node:

(Cfg = 2:{1},{1sec/X1},{}; UpCfg = 1:{1},{},{}; TCres = 1sec/X1; SeqT = <{}>; SeqR = <{}>)

$$EnT = \{t_1\}$$

Evaluation of node parameters:

$$FS = \{\}, RS = \{\}$$

$$Cres' = 1sec/X1$$

$$cond'(t_1) = 1sec/X1 = 1$$

$$CT = \{t_1\}$$

Evaluation of all possible evolutions at the internal time scale:

The only set of transitions for which the maximal context is not false is $SCT_1 = \{t_1\}$

$$Cmax(SCT_1) = 1$$

$$Cevo(SCT_1) = 1 \cdot 1sec/X1 = 1sec/X1$$

Evaluation of the stability context:

$$Csta = 1sec/X1 \cdot 0 = 0$$

**Update the set of direct links**

Evolutions:

$SCT_1$:

UpNode: (Cfg = 2:{1},{1sec/X1},{}; UpCfg = 1:{1},{},{}; TCres = 1sec/X1; SeqT = <{}>; SeqR = <{}>)

$$Cevo = 1sec/X1$$

$$SeqT = <\{\},\{t_1\}>, SeqR = <\{\},\{\}>$$

DnNode:

N3 ( Cfg = 3:{2},{},{}; UpCfg = 2:{1},{1sec/X1},{}; Cres = 1sec/X1; SeqT = <{},{$t_1$}>; SeqR = <{},{}>)

**Internal node N3 analysis**

Analyzed node:

(Cfg = 3:{2},{},{};UpCfg = 2:{1},{1sec/X1},{}; Cres = 1sec/X1; SeqT = <{},{$t_1$}>, SeqR = <{},{}>)

$$EnT = \{t_2\}$$

Evaluation of node parameters:

$$FS = \{X2\}, RS = \{X1\}$$

$$Cres' = 1sec/X1$$

$$cond'(t_2) = 0$$

$$CT = \{\}$$

There are not possible evolutions.

Evaluation of the stability context:

$$Csta = 1sec/X1$$

This node is a leaf.

Stability evolution:

Leaf: (Cfg = 4:{2},{},{O2};UpCfg = 3:{2},{},{}; Csta = 1sec/X1; SeqT = <{},{$t_1$}>, SeqR = <{},{}>)

UpNode = (Cfg = 3:{2},{},{}; UpCfg = 2:{1},{1sec/X1},{}; Cevo = 1sec/X1; SeqT = <{},{$t_1$}>; SeqR = <{},{}>)

$$Cevo = 1sec/X1$$

$$SeqT = \langle\{\},\{t_1\}\rangle,\ SeqR = \langle\{\},\{\}\rangle$$

DnNode: Leaf

**Root 4:{2},{},{O2} analysis**

Root description:

$$Sit = \{2\};\ Temp = \{\};\ Out = \{O2\};\ Wtemp = \{\};\ EnT = \{t_2\}$$

Configuration invariants:

$$Inv(Sit) = \overline{\uparrow a},\quad Inv(Temp) = 1,\quad Inv(Out) = 1$$

$$Inv(Cfg) = \overline{\uparrow a}$$

Actual transition conditions:

$$cond'(t_2) = \uparrow a;$$

Clearable transitions set:

$$CT = \{t_2\}$$

Evaluation of all not timed evolutions:

The only set of transitions for which the maximal context is not false is $SCT_1 = \{t_2\}$

$$Cmax(SCT_1) = \uparrow a$$

$$Cevo(SCT_1) = \uparrow a$$

**Direct links starting from the root**

Not timed evolutions:

$SCT_1$:

UpNode = Root = 3:{2},{},{O2}

Cevo = ↑a

SeqT = <{$t_2$}>; SeqR = <{$h_1$}>

DnNode: N2 ( Cfg = 1:{1},{},{}; UpCfg = 4:{2},{},{O2}; Cres = ↑a; SeqT = <{$t_2$}>; SeqR = <{$h_1$}>)

**Internal node N2 analysis**

Analyzed node:

(Cfg = 1:{1},{},{}; UpCfg = 4:{2},{},{O2}; Cres = ↑a; SeqT = <{$t_2$}>; SeqR = <{$h_1$}>)

EnT = {$t_1$}

Evaluation of node parameters:

FS = {X2}, RS = {X1}

Cres' = a

cond'($t_1$) = 1sec/X1 = 0

CT = {}

There are not possible evolutions.

Evaluation of the stability context:

Csta = ↑a

This node is a leaf.

Stability evolution:

Leaf: (Cfg = 1:{1},{},{};UpCfg = 4:{2},{},{O2}; Csta = ↑a; SeqT = <{$t_2$}>; SeqR = <{$h_1$}>)

UpNode = (Cfg = 1:{1},{},{}; UpCfg = 4:{2},{},{O2}; Cres = ↑a; SeqT = <{$t_2$}>; SeqR = <{$h_1$}>)

Cevo = ↑a

SeqT = <{$t_2$}>, SeqR = <{$h_1$}>

DnNode: Leaf

All stable configurations have been analyzed.

Graphical representation:

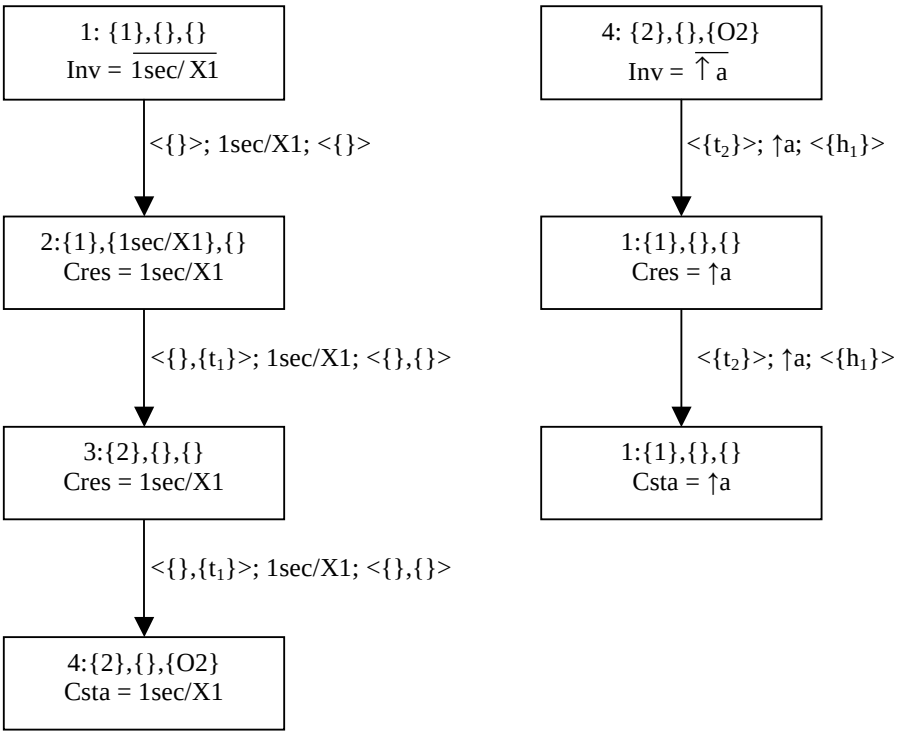


**Figure 4.4.2 Trees of Accessible Configurations**

## 4.4.2 GASC construction

Configurations:

$$cfg_1 = 1: \{1\},\{\},\{\},\ cfg_2 = 4: \{2\},\{\},\{O2\}$$

$$Cfg = \{cfg_1, cfg_2\}$$

The initial configuration is $cfg_1$.

Clocks:

$$H = \{h_1\}$$

Set of timed expressions:

$$TEXP = \{1sec/X1, \overline{1sec/X1}\}$$

Set of expressions on grafcet input variables:

$$EXP = \{\uparrow a, \overline{\uparrow a}\}$$

Set SEQT of SeqT associated with evolutions:

$$SeqT_1 = <\{\},\{t_1\}>,\ SeqT_2 = <\{t_2\}>$$

$$SEQT = \{SeqT_1, SeqT_2\}$$

Set SEQR of SeqR associated with evolutions:

$$SeqR_1 = <\{\},\{\}>,\ SeqR_2 = <\{h_2\}>$$

$$SEQR = \{SeqR_1, SeqR_2\}$$

Set INV of invariants Inv:

$$Inv_1 = \overline{1sec/\ X1},\ Inv_2 = \overline{\uparrow a}$$

$$INV = \{Inv_1, Inv_2\}$$

Set E of evolutions evo:

$$evo_1 = \langle cfg_1, 1sec/X1, , SeqT_1, SeqR_1, cfg_2\rangle,\ evo_2 = \langle cfg_2, , \uparrow a, SeqT_2, SeqR_2, cfg_1\rangle,$$

$$E = \{evo_1, evo_2\}$$

Evaluation of the initial configuration/evolution:

As there are not possible transitions starting from configuration $cfg_1$, starting from the fictitious configuration $cfg_0$ there is only an empty evolution that leads to $cfg_1$ where clock $h_1$ is reset.

Graphical representation:

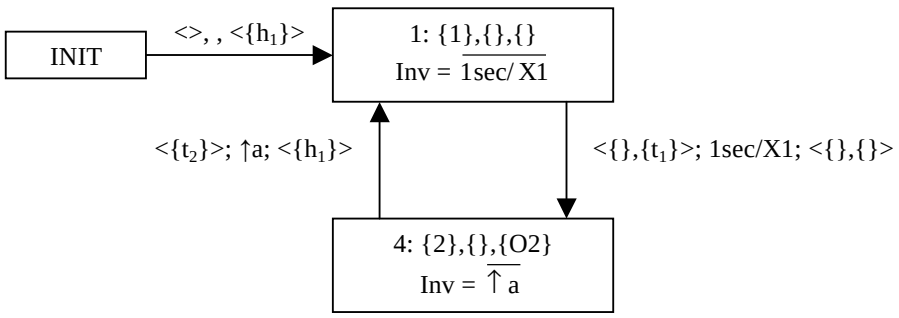


**Figure 4.4.3 Graph of Accessible Stable Configurations**

## 4.4.3 Equivalent timed automaton construction

Clocks:

$$x_1 \text{ corresponds to clock } h_1$$

$$X = \{x_1\}$$

Locations:

$\ell_1$ corresponds to configuration $cfg_1$ and $\ell_2$ corresponds to configuration $cfg_2$

$$L = \{\ell_1, \ell_2\}$$

The initial configuration is $\ell_1$.

Alphabet:

$$\Sigma = \{ \uparrow a , \overline{\uparrow a} \}$$

Set of invariants Inv:

$$Inv_{11} = \{x_1 \leq 1\}$$

$$Inv = \{Inv_{11}\}$$

Set T of transitions $e_i$:

$$e_1 = \langle \ell_1, g_1, \sigma_1, r_1, \ell_2 \rangle,\ e_2 = \langle \ell_2, g_2, \sigma_2, r_2, \ell_1 \rangle,$$

Where $g_1 = \{x_1 \geq 1\}$, $g_2 = \{\}$, $\sigma_1 =$ '', $\sigma_2 =$ '↑a', $r_1 = \{\}$, $r_2 = \{x_1\}$

Graphical representation

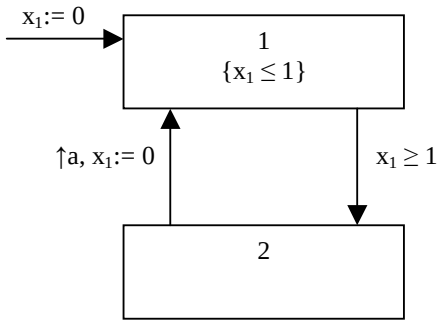


**Figure 4.4.4 Equivalent timed automaton**

## 4.4.4 Region graph construction

The maximum integer that $x_1$ is confronted with is 1. We can see that in location 2 the clock $x_1$ doesn't influence automaton behavior, moreover, before to return in location 1 this clock "passes through" a reset.

The region graph of this timed automaton is:

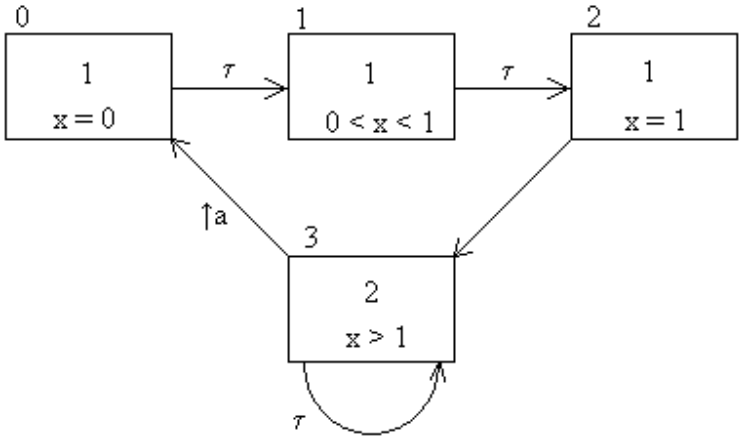


**Figure 4.4.5 Region graph of the equivalent timed automaton**

All timed automaton locations (1 and 2) are reachable, then all GASC configurations are physically reachable. In this case, this is an obvious result but if there are three or four clocks the reachability study problem becomes complex and it is impossible to perform an analysis without a region graph study.

# 4.5 An UPPAAL aided grafcet analysis

The GASC translation into a timed automaton allows to apply timed automata automatic verification tools to a grafcet model. In this paragraph we show how a grafcet verification can be performed with the support of UPPAAL.

Let us consider the grafcet in figure 4.5.1.

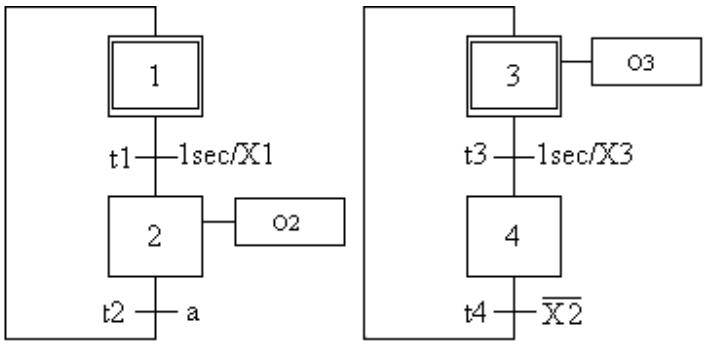


**Figure 4.5.1 grafcet analyzed with UPPAAL support**

Let us suppose that the following properties have to be verified:

- (A) safeness specify: outputs O2 and O3 can't never be activated simultaneously;
- (B) liveness specify: output O2 can be emitted.

A third property that has to be verified is the absence of deadlocks.

The GASC that represents grafcet behavior is represented in figure 4.5.2.

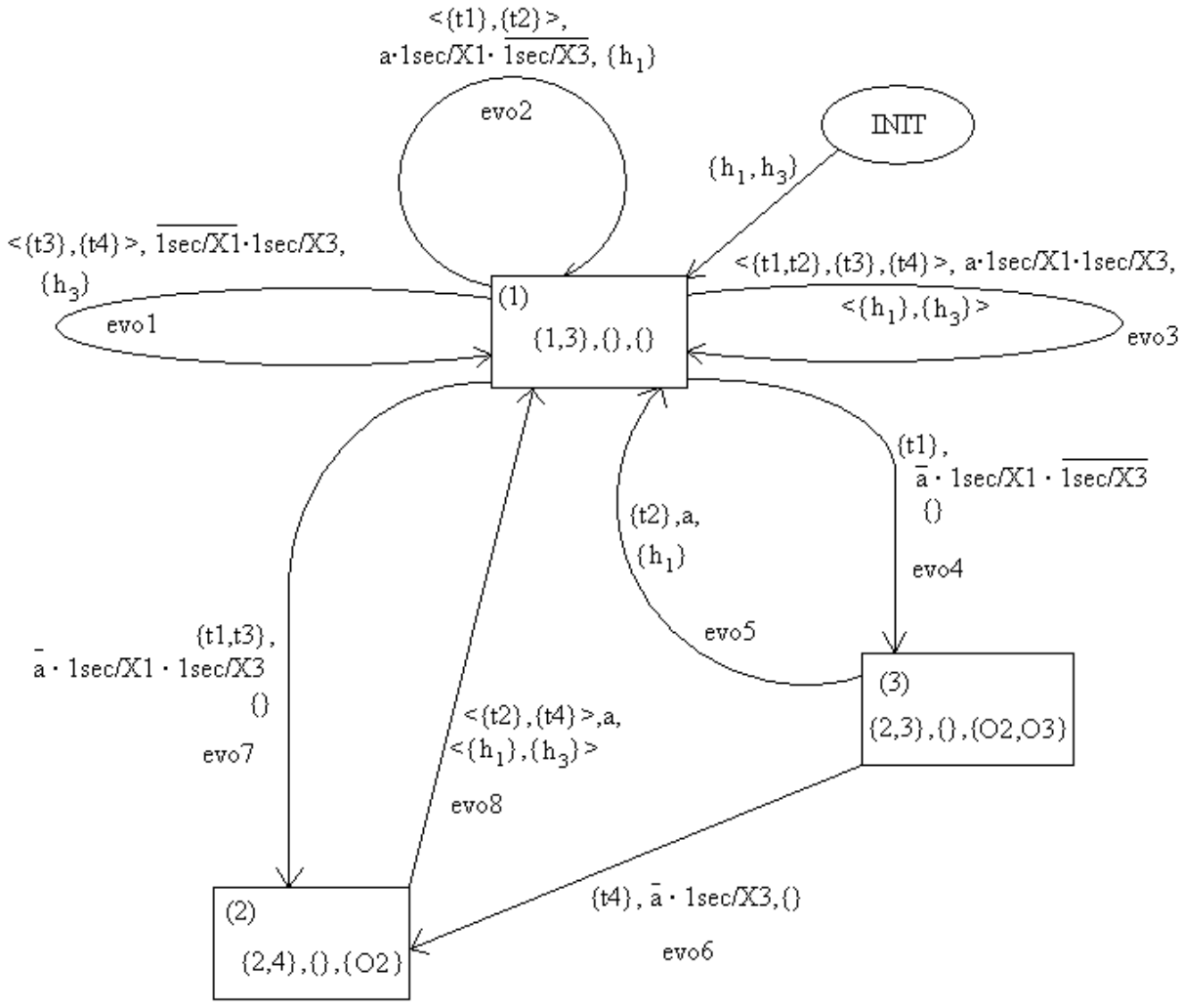


**Figure 4.5.2 GASC of grafcet in figure 4.5.1**

Each evolution is marked with an id (evo1 is the evolution 1, evo2 is the evolution 2…)

We can see that property (A) is not verified in the GASC configuration (3). Potentially we

have the following result: property (A) is not verified and property (B) is verified. However, we can't say "*property (A) is not verified and property (B) is verified*" as a configuration reachability in the GASC is only a necessary condition.

Then we have to translate our GASC into an equivalent timed automaton and then to perform a verification on obtained automaton.

The equivalent timed automaton representation in UPPAAL is shown in figure 4.5.3.

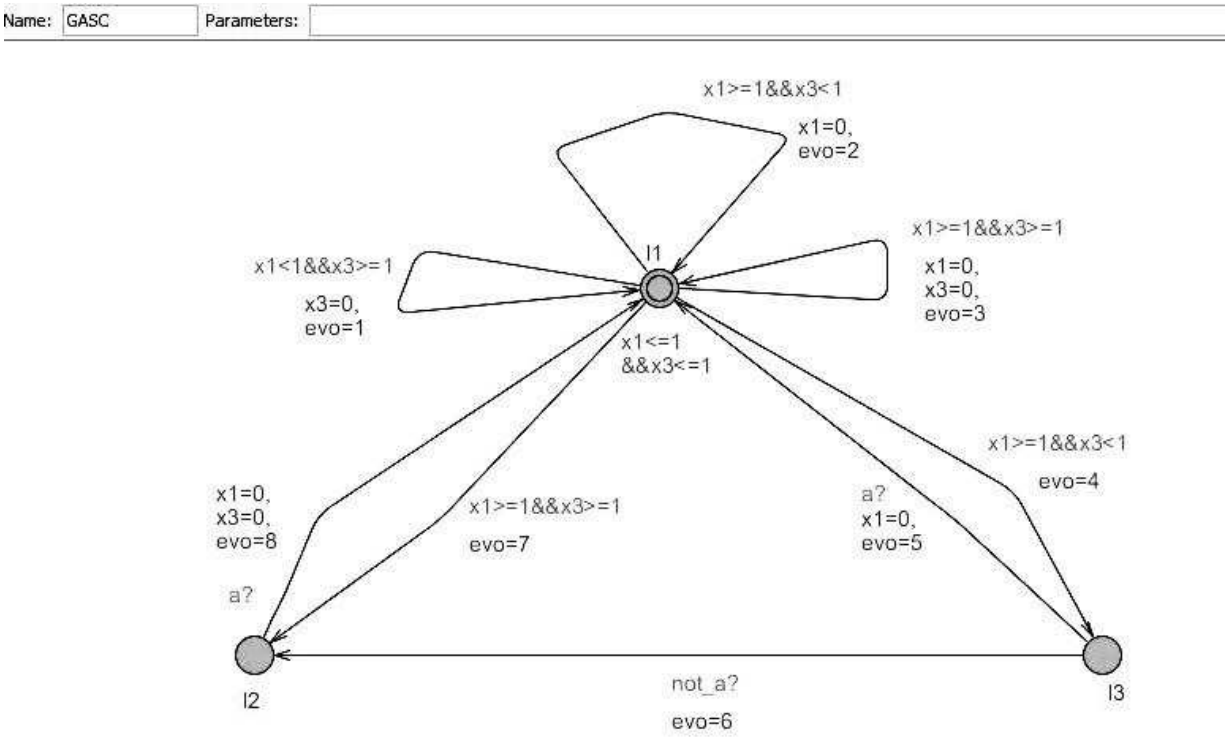


**Figure 4.5.3 Equivalent timed automaton UPPAAL representation**

We associate a location li to each configuration i and a transition evo = j to each GASC evolution evoj.

We can see that the obtained automaton is a nondeterministic timed automaton. We have nondeterminism between evolutions 2 and 4 and between evolutions 3 and 7. This is not a problem, as we only want to know if a location is reachable and if a transition can be cleared.

Now we have to "ask" to UPPAAL which locations are reachable, which transitions can be cleared and if the system is deadlock free. In UPPAAL we have a language that allows to verify these properties. The results are shown in figure 4.5.4.

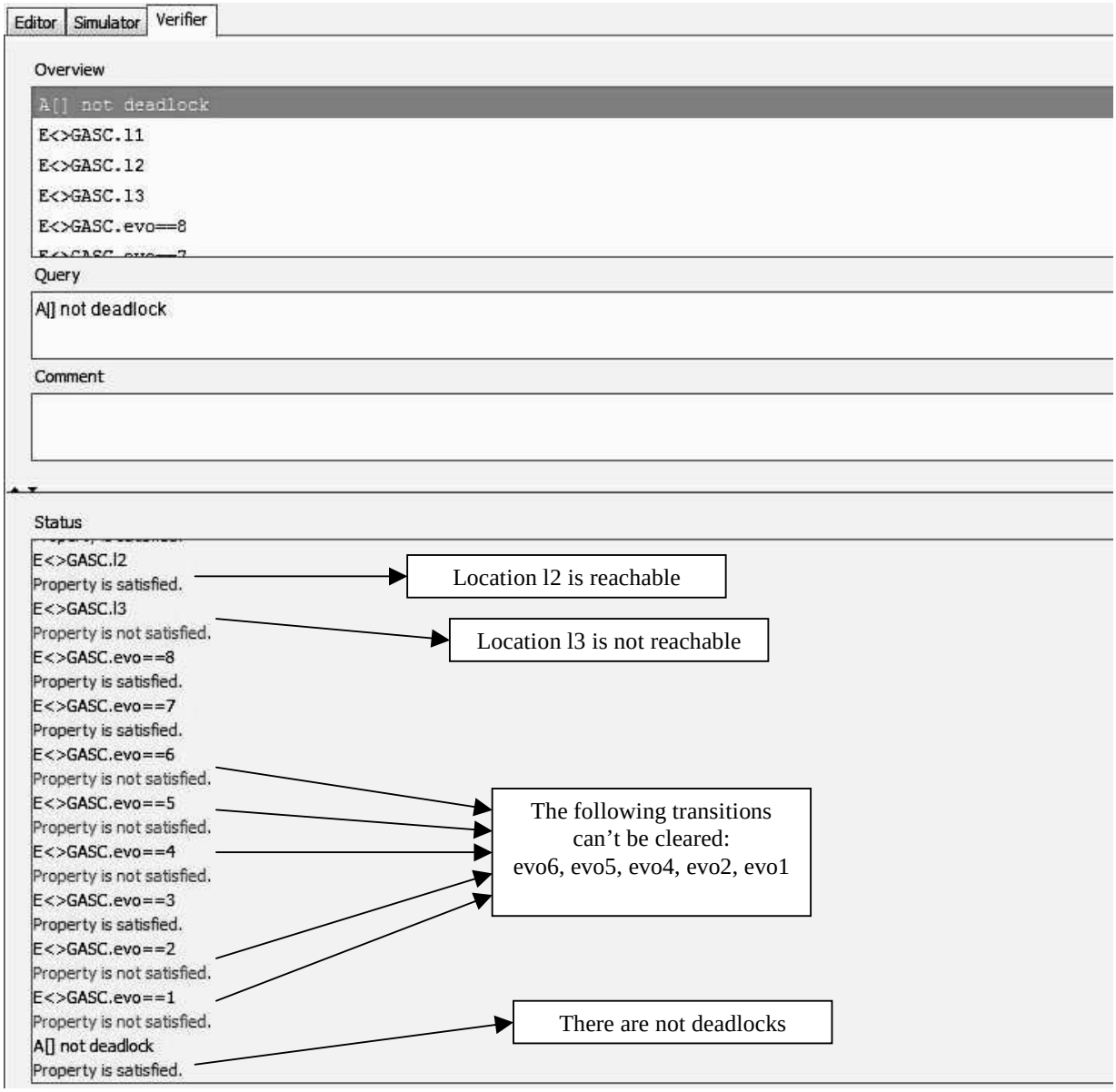


**Figure 4.5.4 UPPAAL verification**

Then we obtain the simplified GASC in figure 4.5.5.

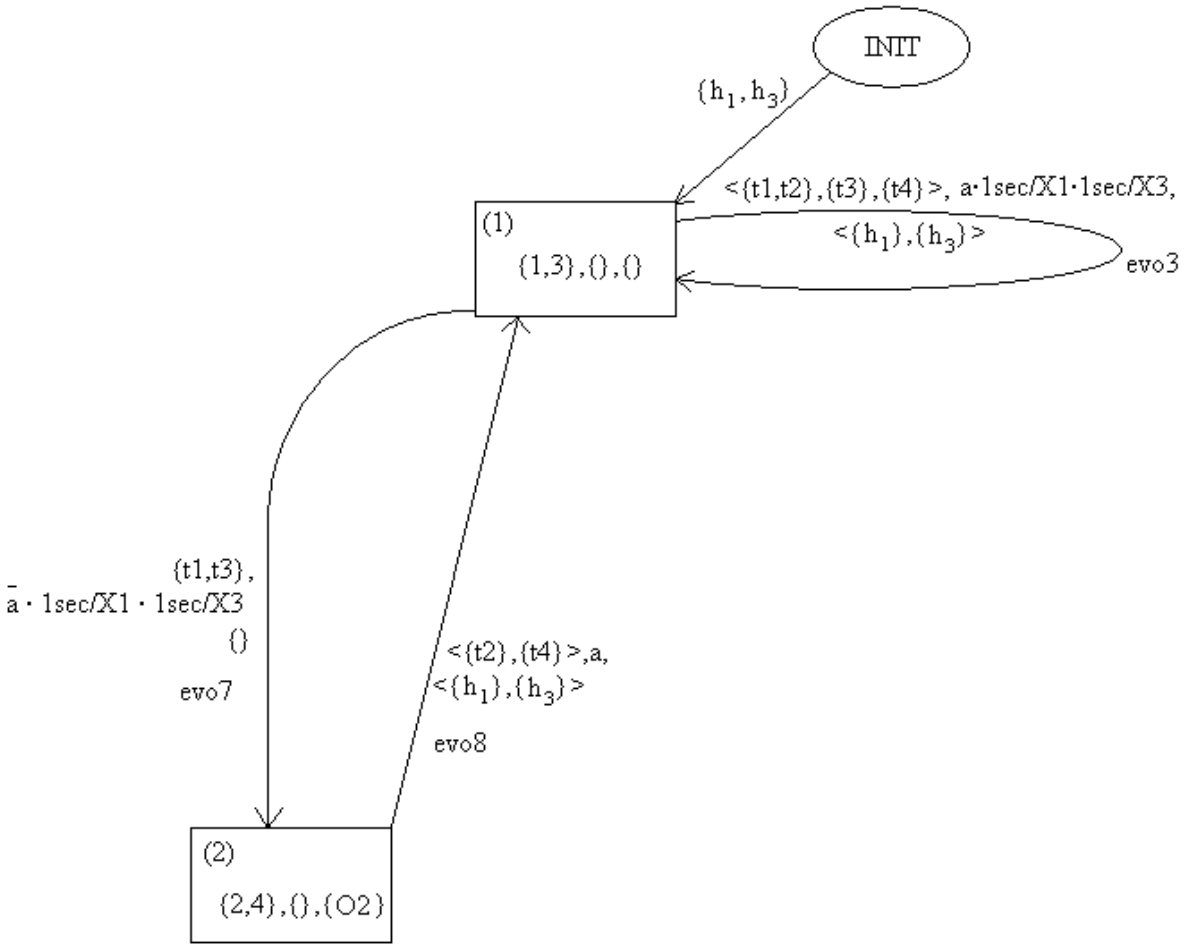


**Figure 4.5.5 Simplified GASC**

The simplified GASC gives a necessary and sufficient reachability condition. Then all specifies are verified: there are not deadlocks, at least a configuration which contains the output O2 is reachable, no configurations which contain O2 and O3 are reachable.

# Chapter 5

# Case study

## Case study

*In this chapter we present a case study. The analyzed grafcet represents a problem of water distribution.*

*The complexity of this grafcet doesn't allow a manual analysis. The GASC evaluation is performed by a python tool that has been carried out by using the algorithm presented in this work. This tool is used at the LURPA laboratory in order to perform formal verification on grafcet models.*

## 5.1 The python tool

The results obtained in this work have been implemented into a python code. This tool evaluates the Tree of Accessible Configurations and the Graph of Accessible Stable Configurations of a grafcet model. Given a grafcet all possible reachable configurations and all evolutions are reported. Also instability cases are reported.

## 5.2 Water distribution problem

The analyzed grafcet is the model presented in the first chapter. The case study concerns the distribution of a water provision used by several production channels. The system is composed of a tank, two pumps, six valves and distribution devices. The request is to generate the start and stop signals for the pumps and the closing and open signals for the valves. We have to take into account eventual pump faults, circuit problems and different requests of distribution. Each pump is characterized by an upstream valve and a downstream valve. The output and reversal valve are common valves (figure 5.2.1).

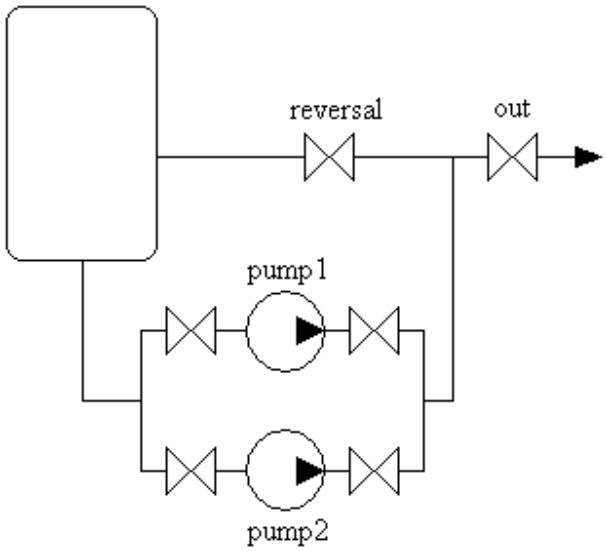


**Figure 5.2.1 Water distribution problem**

This problem was studied in [Roussel 94] and it was proposed by a specialist industrial in water distribution, we report the same grafcet solution with a modification: as in [Roussel 94] time was not taken into account, all timed expressions were replaced with Boolean variables, in this case we don't have this limitation.

### 5.2.1 System description

#### Distribution requests

Two request type: a low supply "low debit" request and a high supply "high debit" request. Only a high supply request at time can be covered and a high and low supply request can be covered simultaneously.

#### The pumps

We have two pumps "pump1" and "pump2" that work alternately. Normally a pump works only 24 hours and each day a commutation is necessary in order to have an equilibrate work distribution. This commutation is performed after that all currently requests have been covered.

A pump fault can be verified: in this case the system is stopped and all valves are closed, then when a new request is received the other one (if there are not other faults) is used. When the pump is repaired, after that all in progress requests are covered, a new commutation is performed.

#### The valves

For each pump we have an upstream valve and a downstream valve, the associated variables are: "upstream pump1", "downstream pump1", "upstream pump2", and "downstream pump2".

#### The general fault

We can individuate a general fault in our system and this is reported with signal "blockage". In this case the system is stopped and all valves are closed. All requests are ignored until the problem is solved.

#### System functioning

Let suppose that a request is received and at least a pump has not faults. If the pump is off, when a request is received the upstream valve is opened and after five seconds the pump is switched on, the downstream valve is opened and then the water distribution starts.

If the pump is working the request is covered instantaneously. When a request of end

distribution is received, the distribution is stopped and if this is the last request then the pump is switched off and the valves are closed. A request is represented with the signals “high debit” and/or “low debit”: if the Boolean value is true then we have a request, the notion of end request is not necessary.

## 5.2.2 The grafcet

The grafcet is the same that we showed in figure 1.2.1, FMP1 and FMP2 describe pump1 and pump2 functioning, ChP generates the pump switching, Rev generates the signal to open reversal valve and Func imposes system behavior.

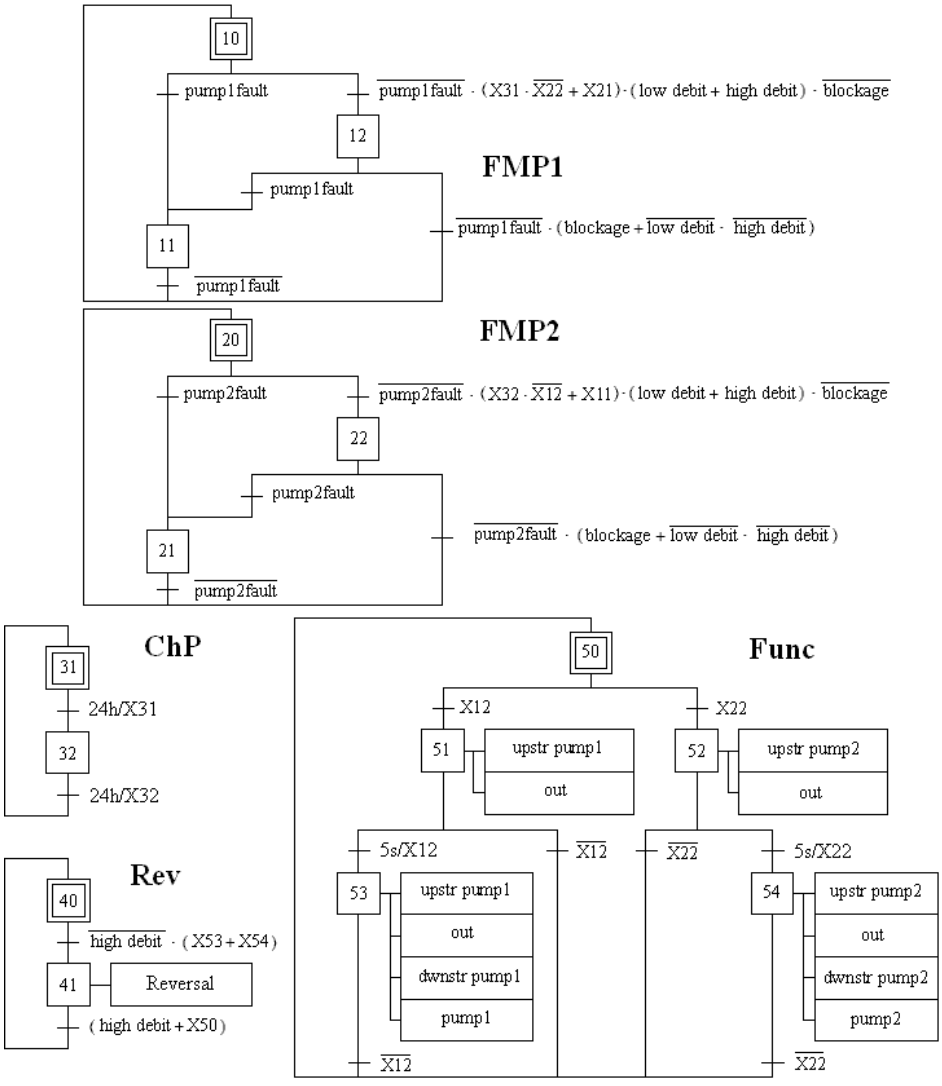


**Figure 5.2.2 Analyzed grafcet**

This grafcet has been analyzed with the python tool, we show the main results.

### 5.2.3 Python tool results

The python tool gives the following results:

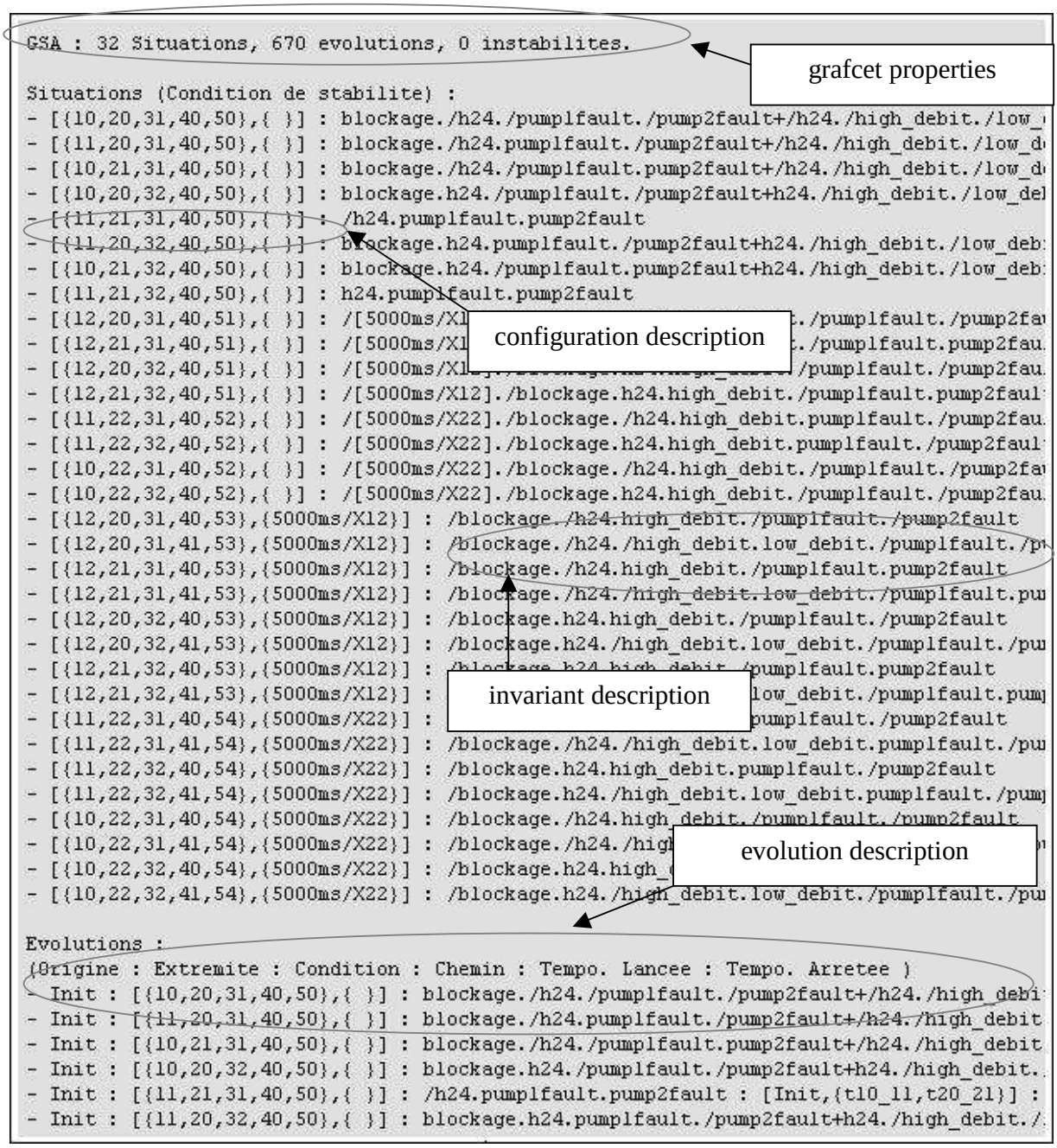


**Figure 5.2.3 Python tool results**

With the Python tool we can know in a few second: the number of (potentially) reachable configurations, the number of possible evolutions, if there are instable situations, for each reachable configuration we have its description and its invariant and for each evolution we know the sequence of transitions sets and the evolution cause.

In this case we have 32 configurations, 670 evolutions and there are not instability situations.

# Chapter 6

# Concluding remarks

# 6.1 Our results

With this work an important contribution has been given to solve the problem of GRAFCET model formal verification. Let us summarize all steps that have to be applied to perform a complete analysis.

- Logic level time:
  - complete grafcet analysis at the internal time scale and construction of the Tree of Accessible Configurations – instability detection;
  - grafcet behavior at the external time scale description with the construction of the Graph of Accessible Stable Configurations – necessary condition for reachability.
- Physical level time:
  - equivalent timed automaton construction – physical time implementation;
  - equivalent timed automaton analysis – reachability study;
  - Graph of Accessible Stable Configurations simplification – necessary and sufficient condition for reachability.

The more delicate step is the TAC construction as GRAFCET behavior interpretation is performed at this level: the presence of an error at this level compromises all analysis results. The GASC representation, as it has been defined, allows to individuate all possible reachable configurations with a detailed description and all evolution causes and consequences. This is a very important aspect because if a non desired behavior is detected, the analyst individuates:

- the input configuration/event and timed condition that generate the unexpected behavior;
- which transition sets and in which order are cleared;
- the starting configurations from which the unexpected behavior can be reached.

An other important aspect is that all this information is obtained with the minimum possible computational complexity.

## 6.2 Further investigations

The results obtained in this work will be improved with further investigations. As not all grafcet possibilities are considered, as stored actions, this work is widely open to new investigation possibilities. As a python tool for GASC has been carried out, a second python tool for equivalent timed automaton construction will be implemented, probably with a direct intarface with UPPAAL tool.

# Bibliography

(In alphabetic order)

[A. Di Febbraro, A. Giua 02]

A. Di Febbraio, A. Giua. *Sistemi ad eventi discreti*. The McGraw Hill Companies, 2002.

[A.Papoulis 02]

A. Papoulis, S. U. Pillai. *Probability, Random Variables and Stochastic Processes*. The McGraw Hill Companies, 2002.

[A.Puri 96]

A. Puri, P. Varaiya. “Decidable Hybrid Systems”. *Computer and Mathematical Modeling*, Vol. 23, N. 11/12, p. 191-202, 1996.

[afcet 83]

Groupe AFCET Systèmes Logiques. *Les interprétations algébriques et algorithmiques et les temporisations du GRAFCET*. Summary document edited by AFCET, 1983.

[Alu 90]

Alur R., Dill D., “Automata for Modeling Real-Time Systems”. *Proc. 17th International Colloquium on Automata, Languages and Programming (ICALP’90)*, vol. 443 of *Lecture Notes in Computer Science*, Springer, p. 322–335, 1990.

[Alu 94]

Alur R., Dill D., “A Theory of Timed Automata”, *Theoretical Computer Science*, vol. 126, num. 2, p. 183–235, 1994.

[Bauer 04]

N. Bauer, S. Engell and others, “Verification of PLC programs given as Sequential Function Charts”, *Integration of Software Specification Techniques for Application in Engineering*, Springer-Verlag, 2004.

[G.Behrmann, P.Bouyer 03]

G. Behrmann, P. Bouyer, E. Fleury, Kim G. Larsen, “Static Guard Analysis in Timed Automata Verification”. *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 2619 of *Lecture Notes in Computer Science*, Springer, p. 254–270, 2003.

[IEC 60848]

IEC 60848 Ed. 2, Specification language GRAFCET for sequential function charts, International Electrotechnical Commission, 2001.

[J.Perrin, F.Binet 06]

J.Perrin, F.Binet, J.-J. Dumery, C. Merlaud, J.-P Trichard. *Automatique et informatique industrielle*, Clarisse Darras edition 1990.

[L’Her 01]

D. L’Her, P. Le Parc, L. Marcé, “Proving Sequential Function Chart Program using automata”, *Theoretical Computer Science 267 – Automata Implementation, selected papers from the Third International Workshop on Implementing Automata*, WIA’98, J.-M Champarnaud Rouen, D. Maurel, D. Ziadi, ed Rouen (France), p. 141-155, 2001.

[P.Bouyer, F.Laroussinie 08]

P. Bouyer,F. Laroussinie, “Model checking in timed automata”. In S. Merz and N. Navet, *Modeling and Verification of Real-Time Systems*, pages 111-140. ISTE Ltd. - John Wiley & Sons, Ltd., January 2008.

[P.Chiacchio 04]

P. Chiacchio, F. Basile. *Tecnologie informatiche per l’automazione*. The McGraw Hill Companies, 2004.

[P.W.Kopke 95]

Thomas A. Henzinger, Peter W.Kopke, Anuj Puri and Pravian Varaya. “What’s decidable about Hybrid Automata?”. *27-th Annual ACM Symposium on Theory of Computing (STOCS)*, p. 372-382, 1995.

[Roussel, Lesage 93]

J.M. Roussel, J.J Lesage. “Une algèbre de Boole pour l’approche événementielle des systèmes logiques”. *APII*, Editions HERMES, Vol. 27 n.5, p. 541-561, 1993.

[Roussel 94]

J.M. Roussel, J.J Lesage. “Analyse de Grafcets par Génération Logique de l’Automate Équivalent”. *APII*, Editions HERMES, Vol. 27 n.5, p. 541-561, 1993.