4.5.8 Firing invariants

A T-invariant of a coloured PN is a vector $\underline{S}$. Its components are functions and it verifies the system of equations $W . \underline{S} = 0$. For the same coloured PN of Figure 4.5a, the T-invariant $\underline{S} = (Id, Id)$ is obtained. Two possible firing sequences, namely $S_1 = T_1/\langle a\rangle . T_2/\langle a\rangle$ and $S_2 = T_1/\langle b\rangle . T_2/\langle b\rangle$ correspond to this characteristic vector. As from the initial marking M_0, the firing of transitions T_1 and T_2 with respect to the same colour leads to M_0. The firing sequences S_1 and S_2 are stationary repetitive sequences.

[EXERCISE 4.10]

4.5.9 Searching for properties

We have seen that there are three major categories of methods for searching for the properties of a non-coloured PN. It goes without saying that these can be used on non-coloured PNs obtained by unfolding, but this often results in overlarge PNs. Can they be extended to coloured PNs?

Although the construction of the graph of markings is possible, generally speaking it is not very realistic, in view of the size it would reach.

The reduction methods for coloured PNs preserving certain properties require strong properties on the functions which intervene in the coloured PN (e.g. orthonormal functions). They can rarely be applied in the general case.

The use of linear algebra is possible, as we have shown in the above sections. Nevertheless, this is rather complicated to use. The systematic search for properties is extremely difficult. On the other hand, *the verification of properties which we know exist is much easier*. Using the specifications, we can for example intuitively find the marking invariants. Verification that these invariants exist permits the description to be validated.

4.5.10 Extensions

All the extensions of non-coloured PNs may also apply to coloured PNs. We can have coloured PNs which are synchronized, timed, interpreted, stochastic, continuous or hybrid. What is associated with a *place* in a non-coloured PN, is associated with a pair (*place*, *token colour*) in a coloured PN, e.g. a timing of a P-timed PN. Whatever is associated with a *transition* in a non-coloured PN, is associated with a pair (*transition, firing colour*) in a coloured PN, e.g. an event for a synchronized PN. The description may become a little awkward, but in theory no problems arise.

Livro :"Petri Nets and Grafect", R. David, H. Halla, Prentice Hall, 1992 (na biblioteca)

5

Grafcet

When Coster invented the mobile character printing press a few decades before Gutenberg made decisive improvements to it, it was essential to modify to some extent handwritten graphical representation so that each letter had a clearly defined shape and was separated from those next to it, even in the same word. This did not call into question either the spelling or meaning of the words. Today both forms, namely 'handwritten' and 'printed' still exist, each being useful in its own right. When ADEPA proposed the representation by Grafcet at standardization, it proposed a 'printed' graphical representation more suited to automatic editing than the 'handwritten' graphical representation proposed by AFCET in the original definition of Grafcet (Figure 5.1). These two forms can exist side by side in the same way as the two forms of writing. We shall ensure that we use both graphical representations fairly extensively.

We shall write Grafcet (with a capital G) when speaking of the tool in general, and grafcet (with a small g) when referring to a particular logic controller model.

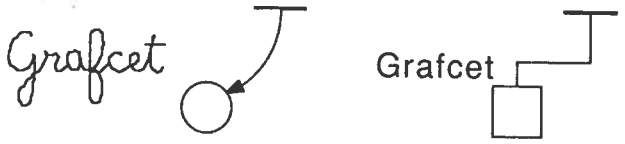

Figure 5.1 Two graphical representations for the same contents.

5.1 BASIC ELEMENTS

Grafcet is designed to represent logic controllers, i.e. systems in which the information has an 'all or nothing' character. Boolean algebra is used (we shall use italics for Boolean variables). A Boolean variable, for example a, can only assume two values, *0* or *1*.

The **complement** of the variable a will be written as a' (the notation $\bar{a}$ is also encountered).

If $a = 0$ then $a' = \bar{a} = 1$, and if $a = 1$ then $a' = \bar{a} = 0$.

The logic **sum** (function OR) of the two variables a and b is written as $a + b$.

The logic **product** (function AND) is written as $a.b$ or ab.

Let us briefly review some properties. Below, the relations given on the right-hand side are the duals of those given on the left-hand side.

$a + 0 = a,\ a + 1 = 1$	$a.1 = a,\ a.0 = 0$
$a + a' = 1,\ a + a = a$	$a.a' = 0,\ a.a = a$
$a(b + c) = ab + ac$	$a + bc = (a + b)(a + c)$
$(a + b)' = a'.b'$	$(a.b)' = a' + b'$

A grafcet is a graph having two types of *nodes*, i.e. *steps* and *transitions* (a grafcet contains *at least one step and one transition*). *Directed arcs* either connect a step to a transition or a transition to a step.

5.1.1 Steps

A step is represented as shown in Figure 5.2, by a circle in the original graphical representation and by a square in the standardized graphical representation (it looks like a *place* in a Petri net, but the word *step* is usually used). A **step** may have two states: it may either be **active** (this is represented by a token in the step) as is the case of Step 2 in Figure 5.2, or **inactive** as is the case of Step 1 in Figure 5.2. The steps which should be active when the system is started up are represented by a double circle or a double square. These are known as **initial** steps (Step 3 in Figure 5.2). **Actions** are associated with the steps, these being the outputs of the grafcet (this will be specified in Section 5.4).

5.1.2 Transitions

A transition represented as shown in Figure 5.3 and is a bar in the original graphical representation. There are a number of cases to be considered in standardized graphical representation. The transition symbol is also a bar but this latter must be preceded by a double bar when two or more arcs join this transition (transitions (2)

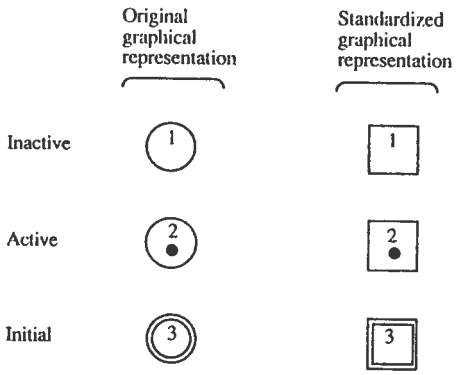

Figure 5.2 Representation of a step.

and (4)) and followed by a double bar when two or more arcs leave this transition (transitions (3) and (4)). A **receptivity** R_i will be associated with each **transition** (i). The receptivity is a function of the grafcet input variables, possibly of the internal state (this will be specified in Section 5.4).

5.1.3 Directed links (or arcs)

The representation of **directed links** (or **arcs**) is illustrated in Figure 5.4. A directed link must always run from a step to a transition or from a transition to a step. In the original graphical representation, the arc direction must be indicated by an arrow, whereas in the standardized graphical representation only the vertical parts running from bottom to top must be marked by an arrow. When two or more directed links join the same step, in the standardized graphical representation, they are grouped together as shown in Figure 5.4 for arcs (3) → 2 and (4) → 2 (note that a similar simplification is admitted in the original Grafcet). When two or more directed links leave the same step, they have a common departure point in the standardized graphical representation as is shown in Figure 5.4 for arcs 3 → (5) and 3 → (6) (this simplification was not admitted in the original Grafcet).

Remark 5.1 In a grafcet, a step may have no input and/or no output transitions. Likewise, a transition may have no input and/or no output steps. A transition without input steps is known as a **source transition** and a transition without output steps as a **sink transition**.

On the other hand, a *directed link must always have a departure node* (transition or step) and *an arrival node* (step or transition). □

[EXERCISE 5.1]

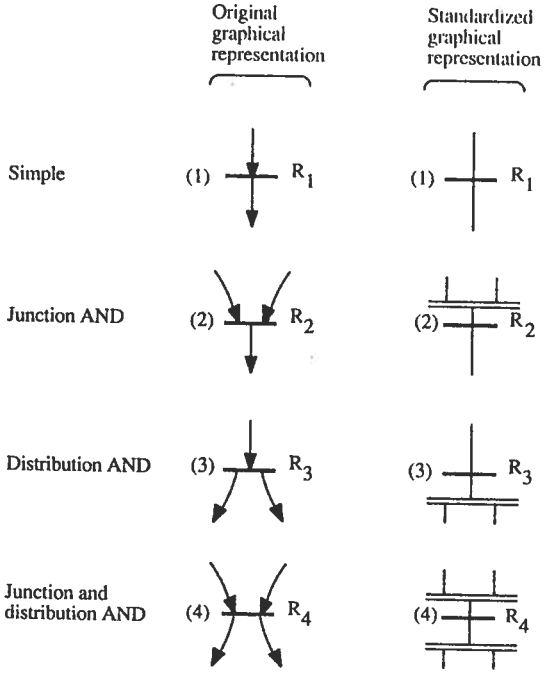

Figure 5.3 Representation of a transition.

5.2 EVOLUTION OF THE STATE

An active step contains one and only one token, and an inactive step contains no tokens. All the active steps at any given moment, i.e. the marking, define the **situation** at this moment. A situation corresponds to a system state. The evolution of the situation is carried out by firing of transitions.

Figure 5.5. represents a grafcet having four steps and three transitions. Steps 2 and 3 are active. The situation is represented by the set of active steps, i.e. {2, 3}. Action *A*, associated with Step 3, is carried out since Step 3 is active. In the standardized graphical representation, an action is written in a rectangle to the right of the step. We shall now introduce the variable written as X_i (in the original definition of Grafcet, this variable was written as ϕ_i).

Definition 5.1 The variable X_i is a Boolean variable which is equal to *1* when and only when Step i is active. □

In Figure 5.5 we have $X_1 = X_4 = 0$, and $X_2 = X_3 = 1$.

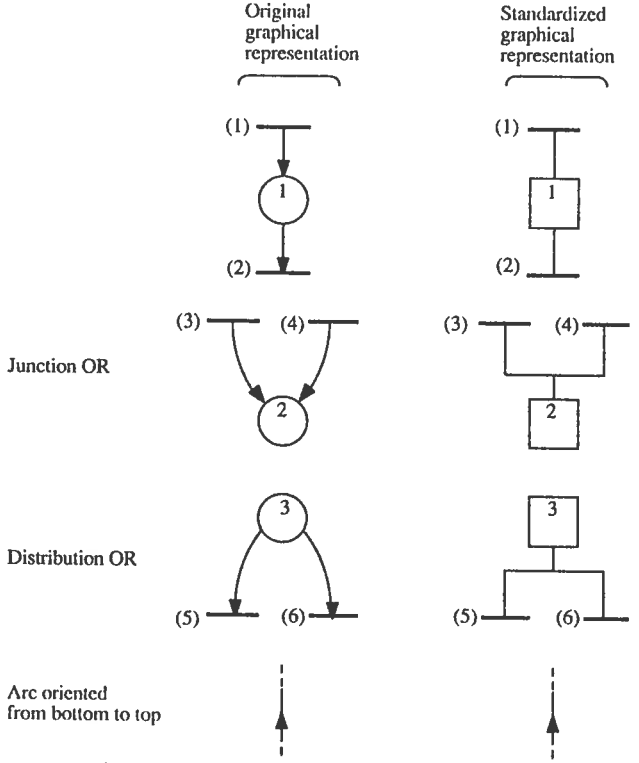

Figure 5.4 Representation of directed links.

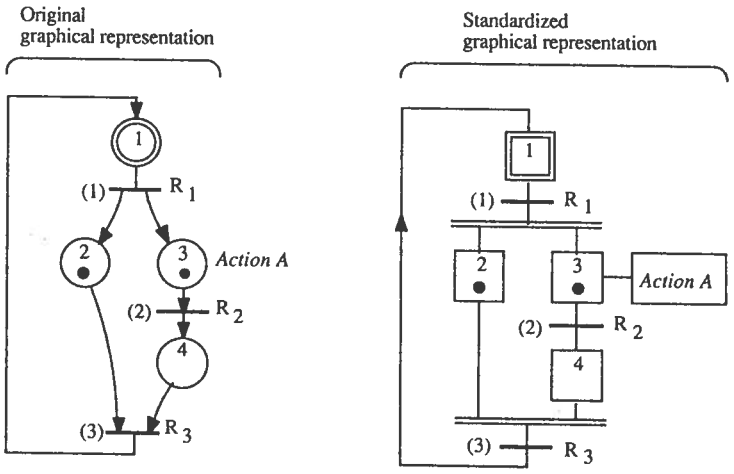

Figure 5.5 Example of a grafcet. R_i is the receptivity associated with the transition (i). Action A is carried out when Step 3 is active.

5.2.1 Fireable transition

A transition is **fireable** if and only if both the following conditions are met.

1. All the steps preceding the transition are active (the transition is said to be **enabled**).
2. The **receptivity** of the transition is true.

For example, in Figure 5.5, transitions (1) and (3) are not enabled. Transition (2) is enabled and it is thus fireable if the receptivity R_2 is true.

5.2.2 Firing of a transition

Firing of a transition consists in inactivating all the steps upstream of the transition and activating all the steps downstream. These operations (activation and inactivation) are indissociable and are carried out simultaneously.

Figure 5.6 illustrates the firing of a transition. The dotted lines above Steps 1 and 2 and beneath Steps 3 and 4 mean that the grafcet may be extended beyond the part which is represented.

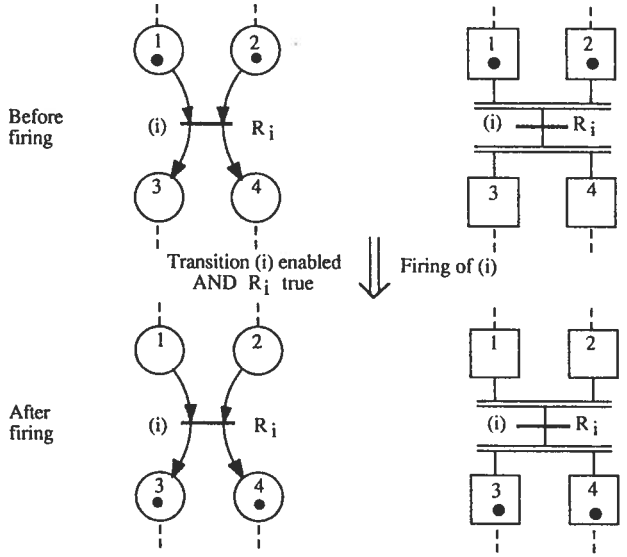

Figure 5.6 Firing of a transition.

A firing has a *zero duration* (it is sometimes said that the duration of firing of a transition is considered to be infinitely small in order to facilitate understanding of the interpretation).

[EXERCISE 5.2]

5.2.3 Firing rules

Rule 1. All fireable transitions are immediately fired (see Section 5.5).
Rule 2. Several simultaneously fireable transitions are simultaneously fired.
Rule 3. When a step must be simultaneously activated and inactivated, it remains active.

These rules are essential for understanding and interpreting Grafcet. The first and third are easy to understand and require no commentary for the time being. The second is extremely important, firstly because its application results in an interpretation different from that of a Petri net (however, we have not yet reached this point which will be dealt with in Section 5.5.5), and secondly because it may be the origin of description errors, a point which will be extensively treated below.

In certain figures such as Figure 5.7, we superimpose the original and the

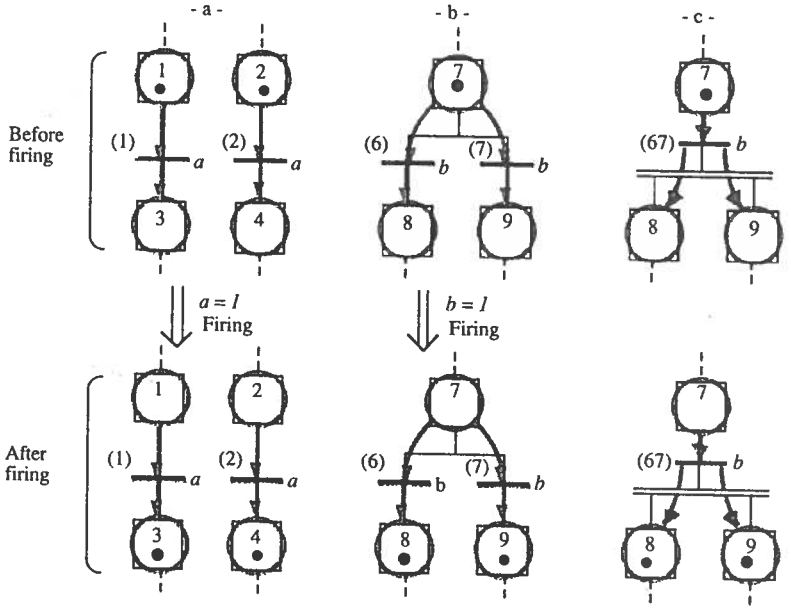

Figure 5.7 Illustration of simultaneous firing.

standardized graphical representations. This will enable us to ensure a continuity with the graphical representation and concepts which are used when dealing with Petri nets.

Comments on Rule 2

Figure 5.7 presents two examples of simultaneous firing. In Figure 5.7a, transitions (1) and (2), belonging to the same grafcet, are enabled. When the Boolean variable a assumes the value *1*, both these transitions are simultaneously fireable and are thus simultaneously fired according to Rule 2. This evolution presents no problems since the enabling conditions for transitions (1) and (2) are independent. In fact, if transition (1) were fired before transition (2), transition (2) would remain enabled, and vice versa.

Figure 5.7b presents a considerably different case. Although transitions (6) and (7) are also simultanously fired when variable b assumes value *1*, we observe that there is no independence between the enablings of these two transitions. In fact, if transition (6) were fired before transition (7), Step 8 alone would be active and transition (7) would no longer be enabled. This cannot occur since $R_6 = R_7 = b$. However, we will encounter cases where $R_6 \neq R_7$. The final result may then depend on the order in which the following three events take place: Step 7 becomes active, receptivity R_6 becomes true, and receptivity R_7 becomes true. Note that Figure 5.7b may be replaced by Figure 5.7c whose evolution is exactly the same but without simultaneous firing.

Figure 5.8b presents a case where it is not known whether only one or both transitions will be fired.

Only transition (1) will be fired if either of the following two cases apply: $a = 1$ and $b = 0$ when Step 1 becomes active, or $a = b = 0$ when Step 1 becomes active and a assumes value *1* before b.

Both transitions will be fired simultaneously if either of the following two cases

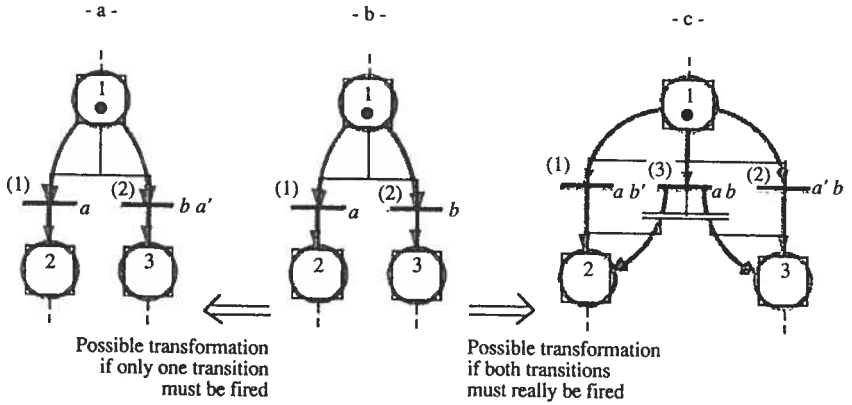

Figure 5.8 How to avoid a 'conflict' between two transitions.

apply: $a = b = 1$ when Step 1 becomes active, or $a = b = 0$ when Step 1 becomes active and a assumes value *1* at the same time as b (this may occur if a and b are not independent, e.g. $a = cd$ and $b = ce$ simultaneously assume value *1* if $d = e = 1$ and c changes from *0* to *1*).

The term **conflict** is used to describe the case when enabling of transitions (1) and (2) depends on a common step and the receptivities R_1 and R_2 may be simultaneously true. This term is not strictly accurate to the extent that functioning of the grafcet is deterministic in all cases, but this word (borrowed from the Petri nets) is a useful one. Although Figure 5.8b is quite admissible in a grafcet, we do not recommend it be used since it may stem from an error on the part of the designer. We shall examine two cases where the designer may have ended up with this graphical representation.

First interpretation of Figure 5.8b

Step 1 corresponds to the availability of a section of railway track, T, on which two tracks T_a and T_b converge. When a vehicle approaches T on track T_a, variable a assumes value *1*. Since Step 1 is active, transition (1) is fired. The situation in which Step 2 is active corresponds to the use of section T by the vehicle coming from T_a. If a vehicle coming from T_b had arrived previously, transition (2) would then have been fired. Since the simultaneous arrival of a vehicle on each of tracks T_a and T_b is considered to be impossible, the designer could have drawn up this grafcet thinking simply that the first to arrive would use track T. But what would happen if a vehicle arrives on T_a and then another on T_b before section T becomes available? We then have $a = b = 1$ when Step 1 becomes active, i.e. simultaneous firing of transitions (1) and (2), meaning that both vehicles use the same section of track. This accident can easily be avoided by giving priority to one of the two vehicles and this is what is shown in Figure 5.8a. If $a = b = 1$, we have the receptivity $a'b = 0$, i.e. priority is given to the vehicle coming from T_a.

Second interpretation of Figure 5.8b

Let us take the example of a furnace whose availability is represented by Step 1. This furnace may treat a batch of parts coming from a workshop A (Step 2) or a batch of parts coming from a workshop B (Step 3). The presence of these batches is indicated by variables a and b, respectively. However, if both batches are present simultaneously, one from each workshop, the furnace is large enough to treat them both at the same time. This may be explained by Figure 5.8c. The possibility $a = b = 1$ is associated with a transition which simultaneously activates Steps 2 and 3. We now have three receptivities which are mutually exclusive, i.e. ab', $a'b$ and ab. There is therefore no conflict, since only one transition can be fired at any one time.

Recommendation 5.1 **Avoid all 'conflict'** on a grafcet. That is to say, if enabling two transitions is conditioned by the same step, it must be ensured that the receptivities associated with these two transitions cannot be simultaneously true. This can always be achieved as shown by Figures 5.7 and 5.8. This means that the

designer must decide whether what he wants is the Figures 5.8a or 5.8c, so as not to make mistakes. □

[EXERCISE 5.3]

5.3 EXAMPLES OF DESCRIPTIONS BY GRAFCETS

In order to make the writing less cumbersome, the same symbol could be used to refer to a sensor or actuator and its associated Boolean variable. The devices will be written in roman characters and the associated *Boolean variables* in *italics*. For example, the Boolean variable *m* will be associated with the push button m. We have $m = 1$ when push button m is pushed. Certain sensors may be *held* in position, for example a push button may remain locked in the *down* position until released. Others, however, may not, i.e. in the example of the push button, we have $m = 1$ only for the time when pressure is maintained, a time which may well be of short duration.

5.3.1 Example 1: Loading a truck

We shall give a simple example (Figure 5.9). A truck may move between points A and B. At A, an operator may ask for the truck to be loaded. The truck proceeds up to point B. Upon arrival, it is loaded by opening a hopper. When loading is complete, the hopper is closed and the truck returns to A where its load is made use of. It will set off again when the operator asks for a fresh loading. In the initial state, the truck is in the stand-by position at point A. The **'functional' grafcet** defining this functioning is represented in Figure 5.9a.

The initial state corresponds to the situation where Step 1 is active. In this situation, there is no action. Transition (1) is thus the only one enabled and is *receptive* to the *loading request*. When the *loading request* receptivity becomes true, transition (1) becomes fireable and is therefore immediately fired. This firing corresponds to the inactivation of Step 1 and the activation of Step 2. In this new situation where Step 2 is active, the action *movement to the right* is carried out, i.e. the *movement to the right* lasts as long as Step 2 remains active. It is the event *arrival at right* which will end this movement by causing transition (2) to fire. At this point Step 3 will become active and the *loading* action will begin. And so on.

In the **functional description** given in Figure 5.9a, the receptivities and actions are indicated using everyday language. The user is free to choose the expressions or symbols which suit him or her. On the other hand, the user must associate quantities which may cause a change in state with the transitions, and the resulting actions with the steps. The grafcet shown in Figure 5.9a represents the desired functioning of the system, but is not yet the description of a logic controller. Indeed

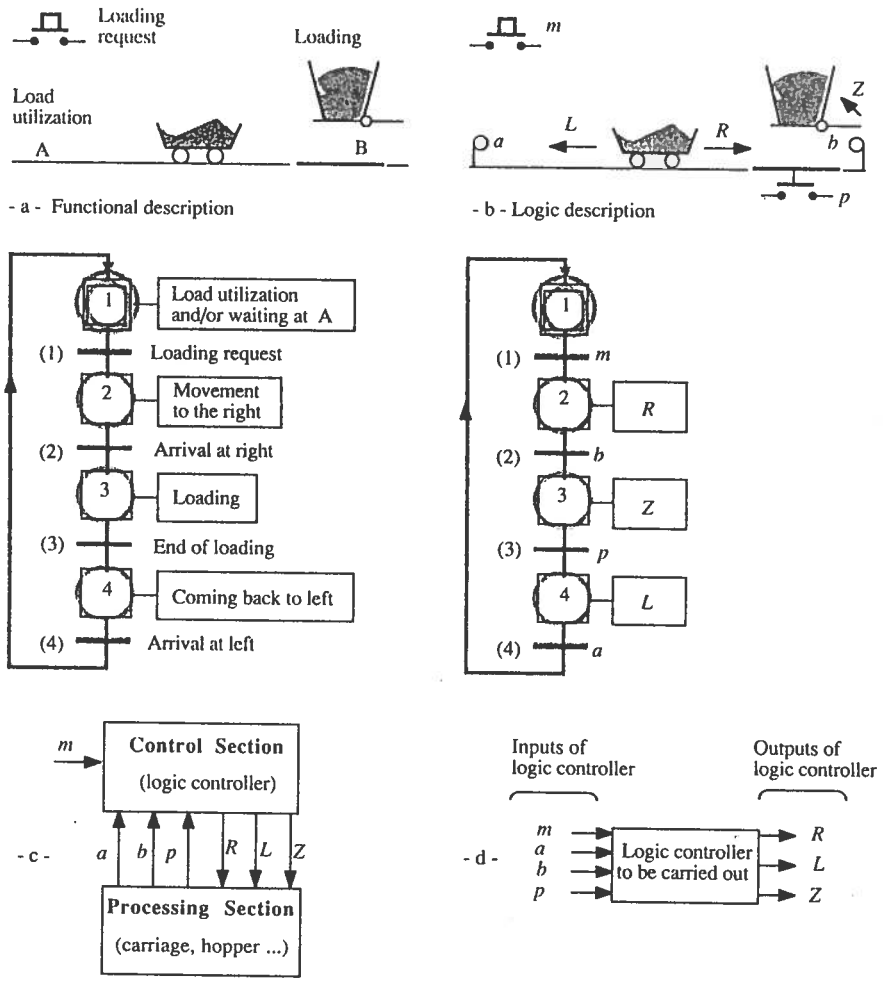

Figure 5.9 Example 1.

the entire set-up could be envisaged as manual. It could be thought that a worker uses the loading at A. Then when the truck is empty, the worker pushes it up to B, opens the hopper, etc. We have thus described the *desired behaviour quite independently of the fact that this behaviour is ensured or not by a logic controller*. In order to describe a **logic controller** in charge of the truck movements and opening the hopper, we shall associate Boolean variables to the system inputs and outputs. Let *m* be the Boolean quantity associated with a loading request, *a* and *b* the variables associated with the presence of the truck at points A and B, respectively, and *p* a variable of value *1* when the truck is filled. These are the inputs of the logic

controller to be described. The outputs are as follows: $R = 1$ when there is movement to the right, $L = 1$ when the truck moves to the left, and $Z = 1$ when the hopper is opened. Using these specifications, the **'logic' grafcet** of Figure 5.9b can be described, from which the logic controller can be implemented. Action R associated with Step 2, for example, means that $R = 1$ for all the time that this step is active.

The complete system may be broken down into two sections as shown in Figure 5.9c: the processing section (made up of the process to be controlled: truck, hopper) and the control section (logic controller to be implemented, described by a grafcet). The logic controller inputs are m, a, b and p, the first one coming from outside the system and the three others being given by sensors placed on the processing section. The logic controller outputs are R, L, and Z which activate the processing section (there may also be outputs to the world outside the system described). Comparison of Figures 5.9b and d clearly shows that the logic controller **inputs** are **associated with the transitions**, whereas its **outputs** are **associated with the steps**.

5.3.2 Example 2: Divider by two

This example is presented in Figure 5.10. The system has an input a and an output S. Figure 5.10a presents an example of a timing diagram. The *initial time* is marked by a *hachured line*. At the initial state $a = S = 0$. Each time that variable a changes from state *0* to state *1*, output S changes state. This system is described by the grafcet in Figure 5.10b, in which the receptivities are the Boolean variables a and a' which are logic **conditions**. That is to say that receptivities R_1 and R_3 are true **if** $a = 1$, while receptivities R_2 and R_4 are true **if** $a = 0$. Output $S = 1$ when either Step 2 or 3 is active. Since these steps **follow one another**, i.e. Step 3 is activated when Step 2 is inactivated, output S is equal to *1* without discontinuity between firing of transition (1) and firing of transition (3). Then $S = 0$ without discontinuity between firing of transition (3) and firing of transition (1). Output S thus only changes value at transitions (1) and (3), i.e. it only changes when a changes from *0* to *1*. The change from value *0* to value *1* of variable a, which is an **event**, will be given as $\uparrow a$ (an event has no duration). The same system can thus be represented by the grafcet of Figure 5.10c in which the receptivities are events $\uparrow a$. If Step 14 (which corresponds to the grouping together of Steps 1 and 4 of Figure 5.10b) is active, transition (1) of Figure 5.10c is **receptive to event** $\uparrow a$. That is to say that transition (1) will be fired **when** this event takes place. We shall then have the situation in which Step 23 (corresponding to the grouping together of Steps 2 and 3 of Figure 5.10b) is active, and transition (3) is enabled. This transition is also receptive to event $\uparrow a$. Firing of transition (3) will occur when event $\uparrow a$ takes place a second time (similar to the behaviour of a synchronized Petri net, Section 2.1). A vivid way to explain this behaviour is as follows: if the firing of a transition corresponds to a time which is infinitely short but not zero, the event $\uparrow a$

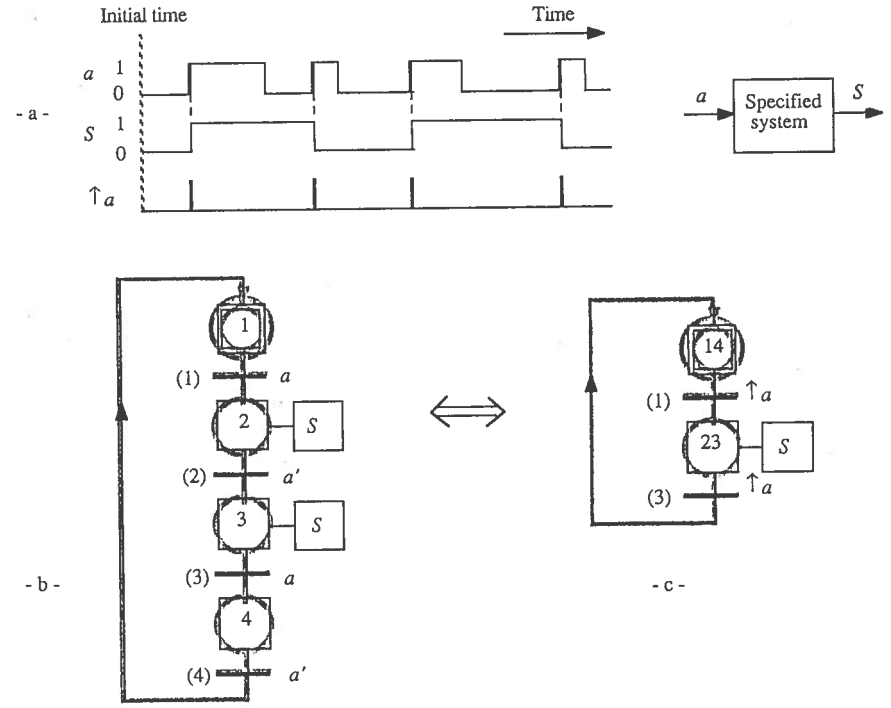

Figure 5.10 Divider by two.

responsible for activating Step 23 is passed when this step becomes active.

This example leads us to make two important remarks.

The first remark is that a condition and an event are quite different in character. A condition is a Boolean variable whereas an event is not, but is rather a *change* in state of a Boolean variable ($\uparrow a$ represents the change from *0* to *1*, i.e. the rising edge of a, and $\downarrow a$ represents the change from *1* to *0*, i.e. the falling edge). A transition receptive to a condition is fired **if** this condition is true. A transition receptive to an event is fired **when** this event occurs.

The second remark is that Figures 5.10b and c represent exactly the same *input/output behaviour*, i.e. the same *sequential machine*. The grafcets of Figures 5.10b and c are said to be equivalent. Two **grafcets** are **equivalent** if *for all input sequences* possible according to the specifications, *they produce the same output sequence* (i.e. the same actions). Although Figure 5.10c is more synthetic than Figure 5.10b, since it contains two steps instead of four, implementation of the corresponding logic controller (by a hardwired or software system) will not necessarily be more simple from the more condensed description.

[EXERCISES 5.4 AND 5.5]

5.3.3 Example 3: Moving a wagon

This example is illustrated in Figure 5.11. At the initial state, the wagon is on the left, i.e. at A. When button m is pressed, the wagon sets off to the right and then returns to the left when it has reached point B, stopping at A. When the wagon arrives at A, it does not set off again immediately, even if the button is pressed. It will only set off when the 'm *is just pressed*' event occurs. This specification sheet gives the functional description of Figure 5.11a. This description makes no assumption concerning the sensors and actuators which will be used (except for the push button). We shall consider two hypotheses which result in different logic descriptions, i.e. in different grafcets. In the two cases considered, Boolean

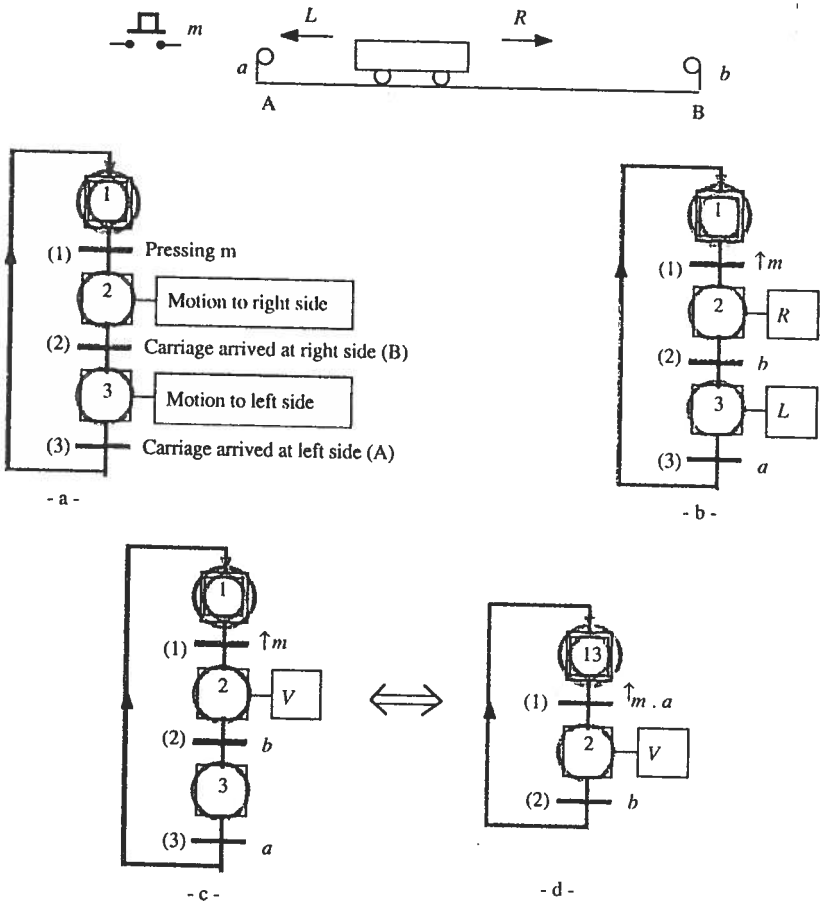

Figure 5.11 Moving a wagon. (a) Functional description. (b) Logic description, first hypothesis. (c and d) Logic description, second hypothesis.

variables a and b are associated with the presence of the wagon at A and B, respectively, and when the push button is pressed, $m = 1$ is obtained.

First hypothesis
Wagon movement is ensured by a two-direction motor, controlled by Boolean variables R and L. When $R = 1$, the wagon moves to the right. When $L = 1$, it moves to the left. When $R = L = 0$, it is stopped. There should (naturally) never be $R = L = 1$ (at any time). The grafcet associated with this hypothesis is represented in Figure 5.11b.

Second hypothesis
Wagon movement is ensured by a jack with spring return, controlled by a single Boolean variable, V. When $V = 1$, the pressure on the jack piston causes the wagon to move to the right. When $V = 0$, this pressure is released and the piston (carrying the wagon with it) returns to its initial position. This functioning is represented in Figure 5.11c. The output variable V is associated with Step 2, corresponding to the right-hand movement, whereas no action is associated with Step 3 since the wagon automatically moves to the left when $V = 0$. It is observed in Figure 5.11c that no action is associated with Steps 3 and 1 which follow each other. That is to say that firing of transition (3), whose receptivity is a, produces a change in situation (i.e. of internal logic controller state) but no change in output in this logic controller. Can this transition be done away with since the next event modifying the output will be $\uparrow m$? The reply is yes, but precautions must be taken. If Steps 1 and 3 are grouped together in Step 13 in Figure 5.11d, the resulting behaviour will be satisfactory if button m is not pressed before the wagon comes to rest. To avoid abnormal functioning when m is pressed while the wagon is returning to the left, condition a needs merely to be added to the receptivity of transition (1). We then have $R_1 = \uparrow m.a$ which is the product of an event, $\uparrow m$, and of a condition a. From the time when Step 13 becomes active, transition (1) will be fired **when** m is pressed and only **if** $a = 1$.

5.3.4 Example 4: Tank filling

The device in question is represented in Figure 5.12. Both tanks are used in a similar way. Tank 1 is empty when the level is less than b_1, i.e. $b_1 = 0$, and is full when the level is greater than h_1, i.e. $h_1 = 1$. At the initial state, both tanks are empty. When button m is pressed, both tanks are filled by opening valves V_1 and V_2. When a tank is full, e.g. tank 1, filling stops (by closing valve V_1) and its contents start to be used (by opening valve W_1). When tank 1 is empty, valve W_1 is closed. Filling may only start up again when both tanks are empty. The filling process is triggered by pressing button m. We shall present a number of grafcets, each one meeting these specifications. The actions relating to the valves are level actions, i.e. if $V_1 = 1$, the corresponding valve is open, and if $V_1 = 0$ it is closed.

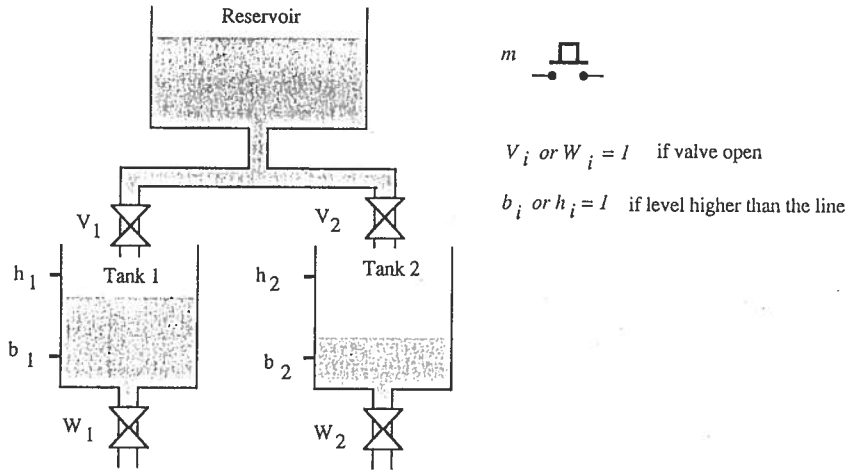

Figure 5.12 Tank filling.

First grafcet (Figure 5.13a)
In the initial situation only transition (1) is enabled. It is receptive to event $\uparrow m$. Firing of this transition simultaneously causes inactivation of Step 1 and activation of Steps 2 and 3. From this moment onwards, each of the tanks functions independently from the other up to transition (6). There is said to be **concurrency** (or *parallelism*). In Step 2, valve V_1 is open, i.e. tank 1 is filled. When it is full ($h_1 = 1$), transition (2) is fired, i.e. Step 2 is inactivated and Step 3 is activated. In this situation, valve V_1 is closed and valve W_1 is opened (since the only action associated with this step is W_1, in other words $W_1 = 1$ and $V_1 = 0$). When tank 1 has been emptied ($b_1 = 0$, i.e. $b'_1 = 1$), transition (3) is fired, thereby inactivating Step 3 and activating Step 4. There is no action in Step 4. It is rather a *stand-by step* at the end of the autonomous evolution of tank 1. If tank 2 is not yet empty, Step 4 remains active until tank 2 has been emptied, expressed by the activation of Step 7. Transition (6) is then enabled. Its receptivity is equal to *1*. An always true receptivity is represented by $=1$, this explicit indication enabling the designer to ensure that he has not forgotten to mention a receptivity in his description). Transition (6) is then fired and there is return to the initial situation. If Step 7 had already been active when Step 4 became active, transition (6) would have been fired as soon as Step 4 was activated. Transition (6) is a **synchronization** transition. Tanks 1 and 2 have evolved in an autonomous manner over a certain time, but their filling is conditioned by the fact that both are empty. Thus one of them must wait until the other is empty. In other words, one of the two stand-by Steps, 4 or 7, will be active for a finite time, while the other will only be active for a short time. If tank 1 is empty before tank 2 when receptivity b'_2 becomes true, this causes the two transitions (5) and (6) to be fired successively.

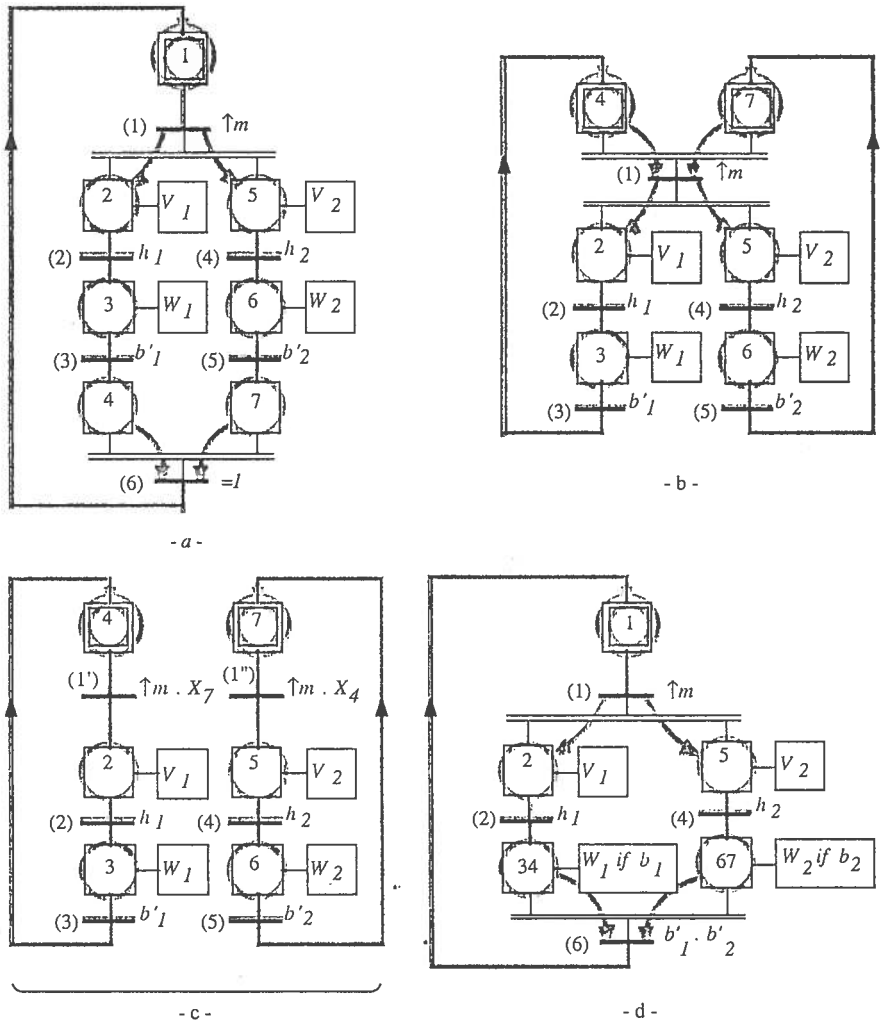

Figure 5.13 Four equivalent grafcets.

Second grafcet (Figure 5.13b)
Each active step is a *component of the state*. It conveys a piece of *information*, i.e. it has a *meaning*. For example, the meaning of active Step 4 in Figure 5.13a is that *tank 1 is empty*. For Step 7, it means that *tank 2 is empty*. For Step 1, it means that *tanks 1 and 2 are empty*. It is immediately seen that there is a redundancy in the description. Indeed, as soon as Steps 4 and 7 are active, transition (6) is fired and Step 1 is activated. However, this Step 1 is not necessary. Transition (1) will be

fired **if** *tank 1 is empty* and **if** *tank 2 is empty* and **when** m *is just pressed*. This may very well take the form shown in Figure 5.13b. There are now two initial Steps, 4 and 7. The joint activity of these two steps is equivalent to the activity of Step 1 in Figure 5.13a. Step 4 has a **single** meaning (*tank 1 is empty*). Step 1 has a **double** meaning (*tank 1 is empty* **and** *tank 2 is empty*).

The grafcet of Figure 5.13a is made in a fairly instinctive manner. We begin with an initial step and then see how this evolves. Figure 5.13b shows that if we start off with two initial steps, each having a **single** meaning (i.e. not **multiple**), the grafcet is simpler (six instead of seven steps). This is a general remark: the grafcet enables the concurrency to be described and it is advisable to associate a single meaning with each step. A multiple meaning or multiple actions associated with the same step come close to the state diagram which is more complicated in the general case where there is concurrency. This will be dealt with in Section 5.8.

Recommendation 5.2 In order to obtain a simple and easily understood grafcet, associate a **single meaning** (i.e. not multiple) and, if it is possible without making the grafcet size too large, with a single level action. □

Third grafcet (Figure 5.13c)
It is clearly visible in Figure 5.13b that there are two almost independent subsystems. Only transition (1) ensures synchronization between the part relating to tank 1 and that relating to tank 2. This system may be represented by using **a grafcet** made up of **two disconnected parts** such as shown in Figure 5.13c. The two transitions (1′ and (1″) of Figure 5.13c correspond to transition (1) of Figure 5.13b. Transition (1′) must be fired **if** Step 4 is active and **if** Step 7 is active, **when** event $\uparrow m$ occurs. In order to ensure this functioning, condition X_7 is added to the receptivity of transition (1′) (variable X_7 is a Boolean quantity which equals *1* when Step 7 is active, in accordance with Definition 5.1). In a symmetrical manner, receptivity $\uparrow m.X_4$ is associated with transition (1″).

Neither transition (1′) nor (1″) can be fired as long as Steps 4 and 7 are not active. When these latter are active, transition (1′) is enabled and condition X_7 satisfied. It becomes fireable when $\uparrow m$ occurs. The same is true for transition (1″). Both transitions are thus simultaneously fireable and are simultaneously fired (Rule 2). Therefore when $\uparrow m$ occurs, there is both inactivation of Steps 4 and 7 and activation of Steps 2 and 5. Functioning is thus identical to that of Figure 5.13b.

The Grafcet allows the variables X_i to be used in the receptivities. This can enable (as in the case of Figure 5.13c) disconnected parts to be obtained, which may be interesting if the part relating to tank 1 (left-hand part of the grafcet of Figure 5.13c) has to be performed on one computer, and the part relating to tank 2 on another. It is clear that the designer must pay great attention to implementation in order to avoid operating hazards, but the functioning described by the grafcet is not ambiguous.

Recommendation 5.3 Avoid using the internal state (i.e. the variables X_i) in the receptivities, in the same connected component of a grafcet. In fact this results in a relation which is **not visualized** by the graphical representation, between the situation and the receptivities, and may give rise to errors. □

Fourth grafcet (Figure 5.13d)
The grafcet of Figure 5.13d may be obtained from Figure 5.13a by grouping together Steps 3 and 4 in a Step 34, and by grouping together Steps 6 and 7 in a Step 67. In Step 34, there must be an action $W_l = 1$ as long as the low level has not been reached. This may be expressed by the **conditional action** W_l **if** b_l. Thus, as soon as Step 34 becomes active, we have $W_l = 1$ (since $b_l = 1$ at that time). The variable W_l will remain at value *1* as long as $b_l = 1$, then W_l will have the value *0*. The reasoning is the same for Step 67. Both tanks must be empty in order to return to Step 1. To ensure this, the receptivity $b_l'b_2'$ is associated with transition (6). If this precaution were not taken, transition (6) would be fireable as soon as it were enabled, i.e. as soon as transitions (2) and (4) were fired. This behaviour would be incorrect since this may correspond, for example, to situation {3, 6} in Figure 5.13a.

Comments on the grafcets
These four grafcets are correct and correspond satisfactorily to the description of the same specifications. However, authors *have a marked preference for Figure 5.13b* in which *all is simple* and clear. There are no steps with a multiple meaning (as in Figure 5.13a). There are no receptivities depending on the internal state (as in Figure 5.13c). There are no conditional actions (as in Figure 5.13d). Note that although Figure 5.13d can be obtained from any of the other three, the reverse is not true. In other words, the conditional actions, in this example, mask the sequential action (we have action $W_l = 1$, then $W_l = 0$).

5.3.5 Example 5: Increase of a counter

The example shown in Figure 5.14 describes a device which counts events $\uparrow a$ of odd rows, starting from initialization. The corresponding grafcet is represented in Figure 5.14a. The **impulse action** $(C \leftarrow 0)^*$ is associated with the initial step, which means placing the value 0 in counter C. An impulse action is considered to be *infinitely short* and is *carried out* once each time the step with which it is associated changes from the **inactive to the active state** (we have chosen to write the *level actions in italics* since they are Boolean variables and the impulse actions in roman characters, but this is not necessary). Thus, in the initial situation which is {1}, the resetting of C is performed. Since transition (1) is enabled and its receptivity is true, it is fired in order to immediately reach situation {2}. Transition (2) is enabled but not fireable before the occurrence of $\uparrow a$. It may thus be seen that at initialization, there was firstly the **unstable situation** {1} for an infinitely short time, followed by situation {2} which is **stable**. This situation will continue as long as event $\uparrow a$ does

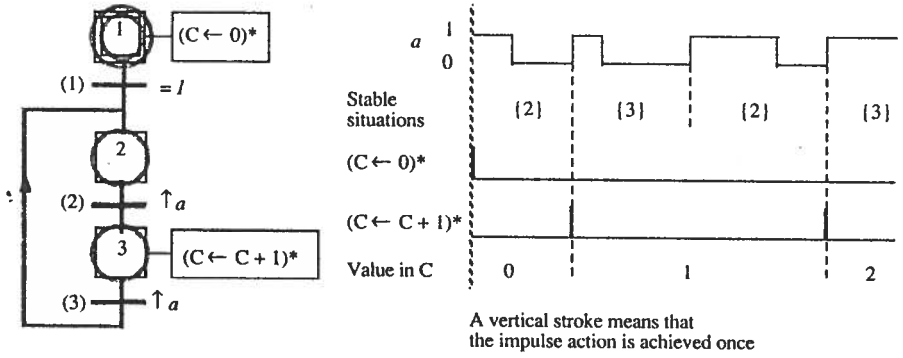

Figure 5.14 Increase of a counter.

not occur. The impulse action associated with Step 1 is carried out even if the activity of this step is transitory. This is a mathematical model, a sequential machine, which describes the input/output relations of a system. This does not mean that an implementation of this machine will function with infinitely short times, but rather that the resulting system must have the behaviour described in Figure 5.14b and which is deduced from Figure 5.14a, i.e. that after initialization the counter will be at 0, after the first rising edge of *a* it will be at 1, after the third rising edge of *a* it will be at 2, etc.

The impulse actions (C ← 0)*, and (C ← C + 1)* which mean incrementing the counter by a unit, are represented by an asterisk which indicates the impulse character of this action. Although this asterisk was originally recommended, it is not always used since it is not always useful. In our example, the increase of a counter corresponds to a **change in state** of the counter and is thus an action which, by definition of the logic and discrete quantities handled, has no duration. The notation C ← C + 1 is without ambiguity.

A comparison may be made between conditions and events on the one hand, and level and impulse actions on the other. The **conditions** and **level actions** correspond to **Boolean variables** (inputs and outputs, respectively), whereas the **events** and **impulse actions** correspond to **changes in state** of Boolean variables (inputs and outputs, respectively).

Remark 5.2 An impulse action, as described above, must not be confused with a **timed action**, which is a level action with a predefined duration, e.g. 2msec (we shall look at examples of this). □

[EXERCISES 5.6 AND 5.7]

5.4 ACTIONS AND RECEPTIVITIES

Time may be used to define a receptivity or to limit the effect of an action. Consequently, we shall show how the *time variable* may be defined.

5.4.1 Taking time into account

In the original Grafcet, a variable T_i^{Δ} is introduced. The same variable is written as *t/i/*Δ in the French standard and this is the notation that we shall use here. In the international standard, other possibilities have been added.

Definition 5.2 The variable *t/i/*Δ is a Boolean variable which equals 1 if and only if a time at least equal to Δ has elapsed since the **last time** that Step i moved from the inactive to the active state. □

This is illustrated in Figure 5.15. Step 5 is active, i.e. $X_5 = 1$. When variable *a* assumes the value *1*, Step 6 becomes active, i.e. X_6 assumes the value *1*. The Boolean variable *t/6/10sec* assumes the value *1*, 10 seconds later. It will keep this value until the variable X_6 next changes from state *0* to state *1*. Note that the moment when Step 6 becomes inactive again has no influence. This is illustrated in Figure 5.15 where *t/6/20sec* assumes the value *1*, 20 seconds after Step 6 changes from the inactive to the active state, although this step had become inactive again.

Details on the notation t/i/Δ
See Figure 5.16.

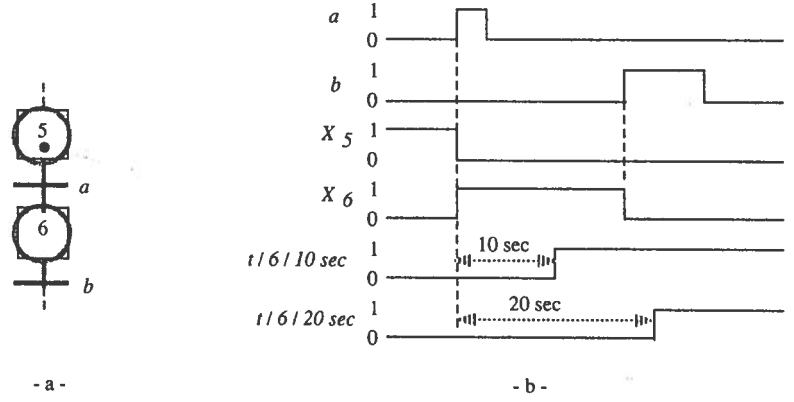

Figure 5.15 Illustration of the notation *t/i/*Δ.

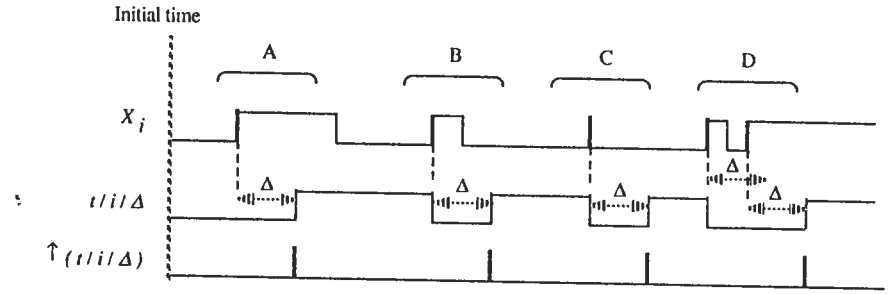

Figure 5.16 Details of the notation $t/i/\Delta$.

a) Even when Step i is only active transiently (unstable situation), there is a change-over from the inactive to the active state and thus this suffices to trigger a time delay (case C of Figure 5.16 which is a special case of case B where Step i has remained active for a time less than Δ). In other words, the triggering of a time delay may be likened to an impulse action.
b) When two events $\uparrow X_i$ occur in an interval of time less than Δ, it is the last change-over to the active state that is considered (case D of Figure 5.16).
c) The various cases stated comply with Definition 5.2 which may be specified as follows. Let t_1, t_2, t_3 be three successive times where Step i changes from the inactive to the active state:

   If $t_1 + \Delta < t_2$ and $t_2 + \Delta < t_3$
   then $(t/i/\Delta) = 1$ in intervals $[t_1 + \Delta, t_2)$ and $[t_2 + \Delta, t_3)$
   If $t_1 + \Delta \geq t_2$ and $t_2 + \Delta < t_3$
   then $(t/i/\Delta) = 1$ in the interval $[t_2 + \Delta, t_3)$

d) Since $(t/i/\Delta)$ is a Boolean variable, the event $\uparrow(t/i/\Delta)$ can be defined and is illustrated in Figure 5.16

[EXERCISE 5.8]

5.4.2 Actions and outputs

There are two major categories of actions, namely **level** and **impulse** actions.

5.4.2.1 Level action

A level action is modelled by a Boolean variable. It may be **unconditional** or **conditional**. Some examples are presented in Figure 5.17.

The action *valve V open* associated with Step 2 in Figure 5.17a is unconditional.

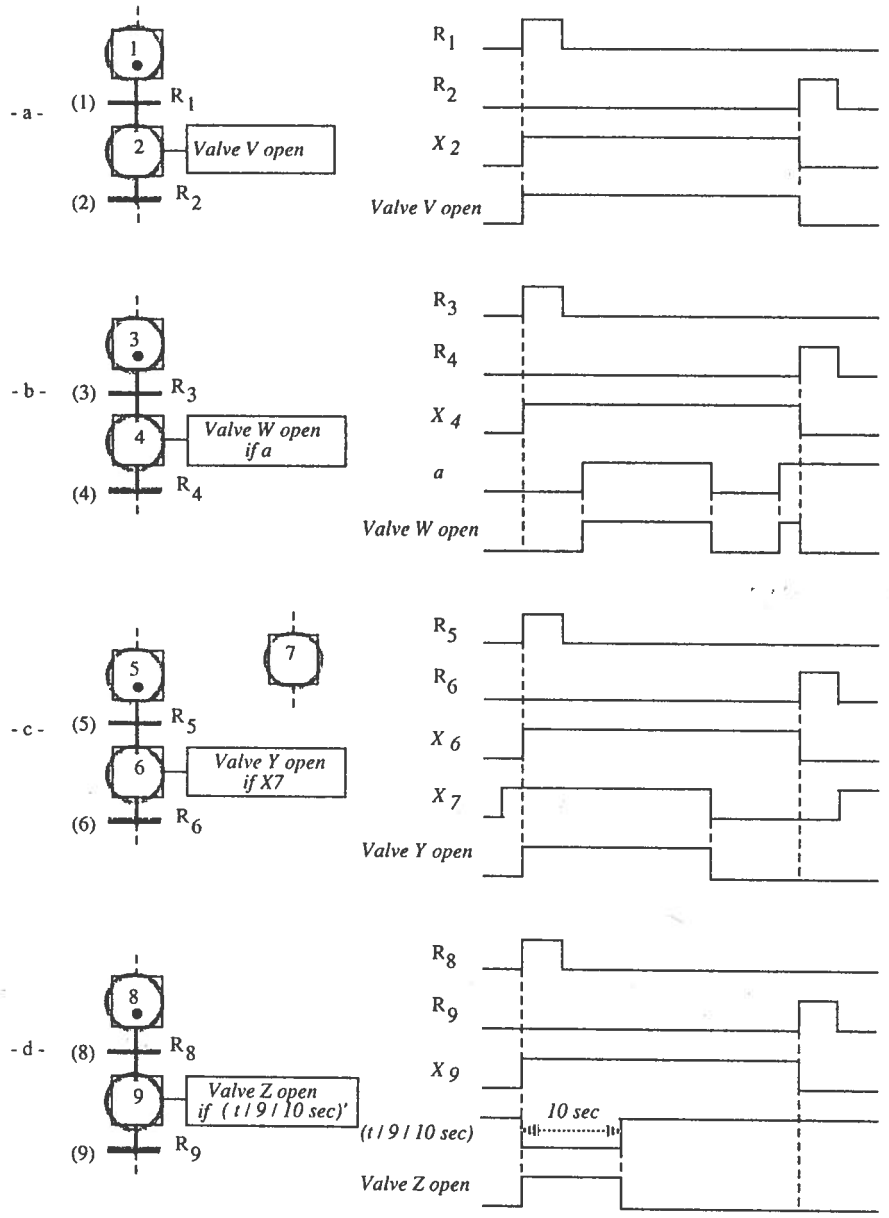

Figure 5.17 Level actions.

This means that valve V is opened as from when Step 2 becomes active up to the moment when this Step becomes inactive, without any other condition.

The actions are conditional in Figures 5.17b, c and d. In Figure 5.17b, the action *valve W open* is conditioned by the Boolean variable a. This means that the action is carried out (i.e. that the corresponding Boolean value is *1*) for all the time that Step 4 is active **and** $a = 1$. We thus have the Boolean product *valve W open* $= X_3.a$. In Figure 5.17c, the condition is the Boolean variable X_7, which corresponds to the active state of another grafcet step, Step 7. In Figure 5.17d, the condition is relative to time. Valve Z is in the open state if $(t/9/10\text{sec})' = 1$, i.e. if $(t/9/10\text{sec}) = 0$. This means to say that valve Z is open for 10 seconds at most as from when Step 9 becomes active (possibly less if this Step remains active less than 10 seconds). If we wish this opening period to last exactly 10 seconds, it may easily be described as shall be seen further on. This is a **limited duration action** which must not be confused with an impulse action which has no duration (Remark 5.2).

The condition associated with an action may be any Boolean function of the input, internal and time delay variables, for example:

Fill the tank if $a'.(t/8/1\,min) + X_3X_4'$

[EXERCISE 5.9]

Examples of level actions

Filling tank B.
Valve V open.
Robot arm moving to the right.
Opening of clamps.

Remark 5.3 A **level action** has *by definition* a **finite duration** (or possibly infinitely long). In other words a level action with an infinitely small duration cannot exist, since this would have no meaning. It follows that a level action is only defined for **stable situations**. This is illustrated in Figure 5.18. When receptivity R_8 becomes true, receptivity R_9 is already so. Transitions (8) and (9) are fired successively. Step 8 is transiently active (i.e. for an infinitely short time). Thus the *valve V open* action **remains at** *0*. Bear in mind that a grafcet describes a sequential machine in the mathematical sense, and not an implementation of this sequential machine. The designer must ensure that the implementation really corresponds to the sequential machine described by the grafcet, i.e. that the hardwired or software logic controller does not produce hazards in a case such as the one we have just shown. □

[EXERCISES 5.10 AND 5.11]

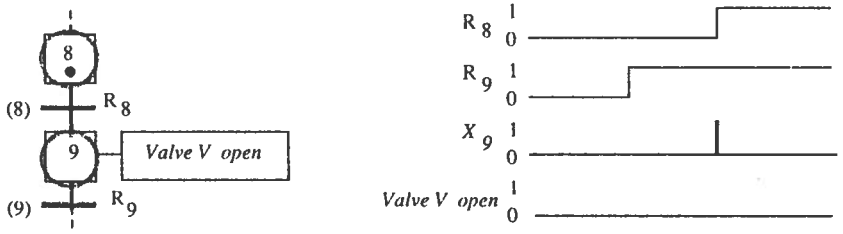

Figure 5.18 A level action cannot be of infinitely short duration.

5.4.2.2 *Impulse action*

An impulse action is responsible for changing the value of a discrete variable. This latter may or may not be a Boolean variable (e.g. the value of a counter). An impulse action associated with a Step is carried out as soon as this Step changes from the **inactive to the active state**, regardless of the duration during which this Step remains active (even infinitely short). An *impulse action may be said to be an order* (do this ...), whereas a *level action indicates a state*. Two examples are presented in Figure 5.19.

In Figure 5.19a, the actions 'Open valve V' and 'Close valve V' are impulse actions (orders). They are infinitely short actions which modify the state of the Boolean variable *valve V open*. This Boolean variable is not an action produced by the grafcet, but is deduced from it. These impulse variables may be used if there is a device for memorizing the state of valve V. It will be observed that Step 4 only changes transiently to the active state. The grafcet describes the fact that valve V must be closed as soon as Step 4 becomes active.

Figure 5.19b shows a **calculation** example. Variable P has a numerical value which is 5 in the situation shown in the figure. As soon as Step 7 becomes active, we have an impulse action ($P \leftarrow P - 3$) which consists in modifying the value of P, which thus becomes 2. The model assumes that the calculation action is carried out in an infinitely short time. This means that if the receptivity R_9 depends on the value of P, it is the value after calculation, i.e. $P = 2$, which must be considered when calculating this receptivity.

Examples of impulse actions

Calculate the speed of the mobile M.
Open the clamps.
Switch off pilot lamp L.
Start up motor M_2.

Remark 5.4 The *asterisk* recommended at the origin of Grafcet to indicate the impulse character of an action does not always serve a purpose. We could have

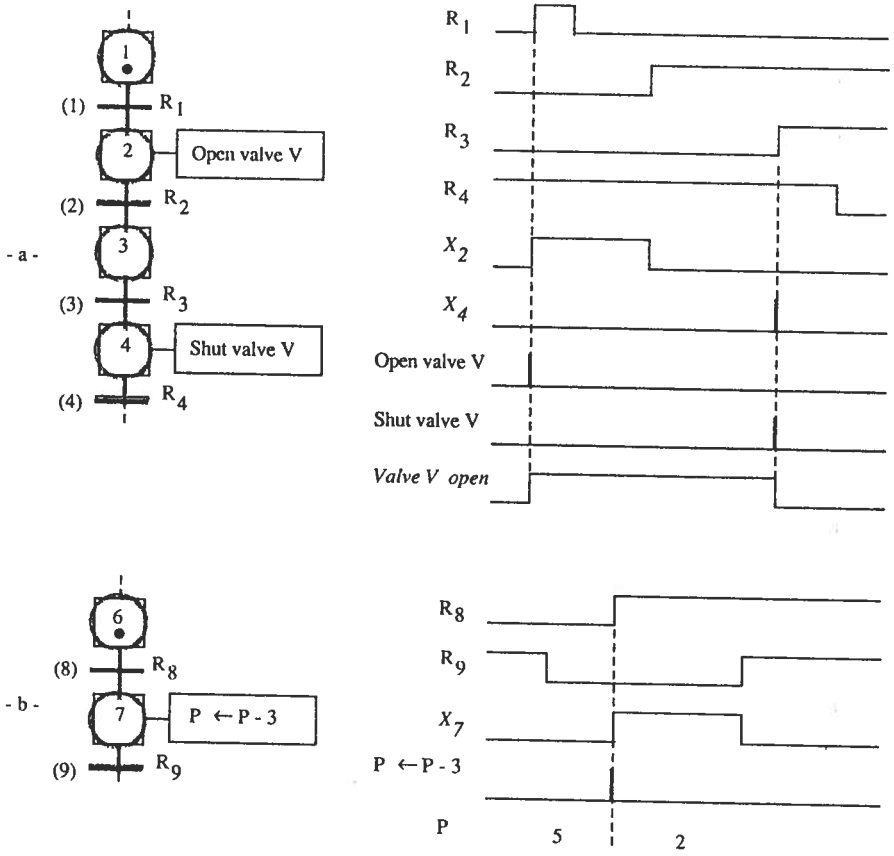

Figure 5.19 Examples of impulse actions.

written, for example (Open valve V)* and (P ← P − 3)*, but there is no ambiguity concerning the impulse character of these actions. We shall only use this notation when the symbol associated with the action does not permit the nature of the action to be distinguished (see Figure 5.20). □

5.4.2.3 *Actions and outputs*

Only the notion of *action* is initially defined in the Grafcet. However, certain actions are not outputs and certain outputs are not actions if the following definition is admitted. An **output** corresponds to a signal which acts on the environment of the system described (either on the process controlled, or on a

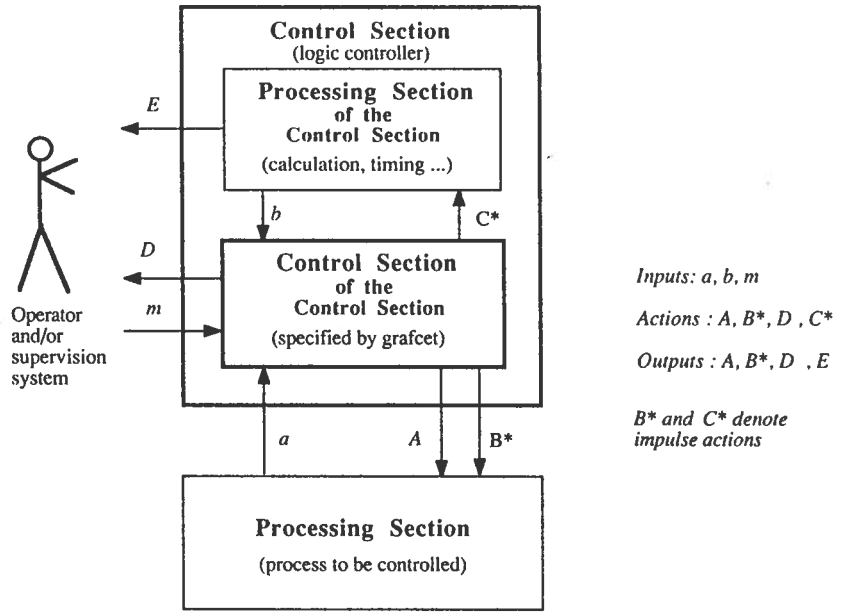

Figure 5.20 The logic controller and its environment.

supervision system or a human operator, for example by means of a visualization). This is shown in diagram form in Figure 5.20. The impulse action C* is not an output but controls an operation on the processing section of the control section which is the logic controller: counter incrementation, calculation order, etc. (The triggering of a time delay start may be likened to an impulse action such as C*, but does not need to be explicit. Indeed, if the variable $t/i/\Delta$ is used, this implies that the time delay must be performed when Step i changes from the inactive to the active state. Variable E is a logic controller output, but is not an action in the Grafcet sense (it may be, for example, a counter value or any Boolean or numerical quantity whose evolution is controlled by impulse actions such as C*).)

Remark 5.5 There is no defined agreement for *conditional impulse actions*. A number of definitions are possible and these can always be disregarded when describing a logic controller. They may, however, be used, provided that they are clearly defined. Figure 5.21 presents three possible interpretations for action A* *if* B. The first corresponds to action A* *when* $\uparrow X_3$ *if* $B = 1$, the second to *action* A* *when* $X_3 = 1$ *as soon as* $B = 1$, and the third to *action* A* *when* $\uparrow (X_3 . B)$. The notation *action* A* *if* B is therefore insufficient. □

[EXERCISE 5.12]

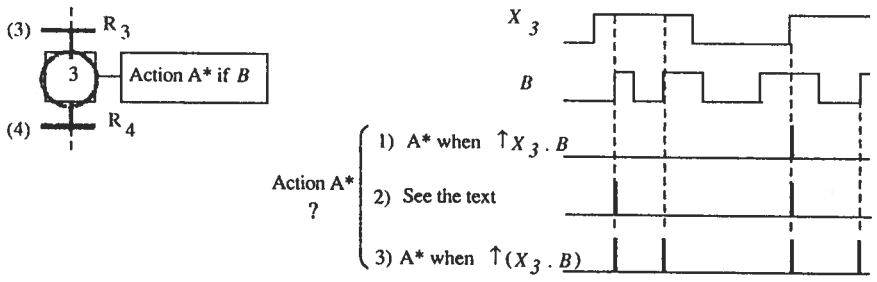

Figure 5.21 Three possible interpretations of *action* A* *if B*.

5.4.3 Receptivities

Let us first define what is meant by external and internal variables.

An **external variable** is a Boolean variable which may either come from the process to be controlled (e.g. a in Figure 5.20), or the outside world (e.g. m in Figure 5.20 which may come from an operator or another system), or a variable $t/i/\Delta$ relative to time.

An **internal variable** is a Boolean variable relating either to the situation of the grafcet (e.g. X_5), or the state of the processing section of the control section (e.g. $b = 1$ if counter C contains a value greater than 10).

Remark 5.6 The Boolean variables (or predicates) coming from the control section are internal variables, except for those which relate to time delays. This is because the end of a time delay occurs in an asynchronous manner, like an external variable. The ends of timings could possibly be considered to be not part of the control section. This is purely formal. □

A **receptivity** may be either a *logic condition* or an *external event*, or *an event and a condition*.

5.4.3.1 *Condition*

A condition written as C_i is a Boolean function of the *external and internal variables*, for example:

$C_1 = a' + b.X_4$
$C_2 = (t/4/10sec) + b.c$

5.4.3.2 *Event*

An external event (or simply an event) written as E_i, is a rising or falling edge of an *external variable* (or of an external variable function), for example:

$E_3 = \uparrow a$
$E_4 = \downarrow (a + b)$
$E_5 = \uparrow (t/8/10min)$

5.4.3.3 *Event and condition*

For example, the event may be $\uparrow a$ and the condition $(b + X_3)$. We thus have:

$R_6 = \uparrow a.(b + X_3)$

This third case is general, whereas the first two are particular cases. Indeed, a receptivity R_i may always be written as:

$R_i = E_i.C_i$

using:

a) the **'always true' condition**, written as $C_i = 1$, which is associated with receptivities R_i which only depend on an external event E_i. For example $R_2 = E_2$ means that $C_2 = 1$, thus

$R_2 = E_2 \Leftrightarrow R_2 = E_2.1$

b) the **'always occurrent' event**, written as e, which is associated with receptivities R_i which only depend on a condition C_i. For example $R_3 = C_3$ means that $E_3 = e$, thus

$R_3 = C_3 \Leftrightarrow R_3 = e.C_3$

Notation The symbol $\uparrow$ takes priority over the symbols (.) and (+). That is to say that $\uparrow a.b = (\uparrow a).b$, and $\uparrow a + b = (\uparrow a) + b$. □

5.4.3.4 *Examples of receptivities*

$R_1 = \uparrow a.b.X_2 \Rightarrow E_1 = \uparrow a$ and $C_1 = bX2$
$R_2 = \uparrow (t/4/5sec)(a + b) \Rightarrow E_2 = \uparrow (t/4/5sec)$ and $C_2 = a + b$
$R_3 = \downarrow a \Rightarrow E_3 = \downarrow a$ and $C_3 = 1$
$R_4 = b \Rightarrow E_4 = e$ and $C_4 = b$

There may also be receptivities expressed in various forms.

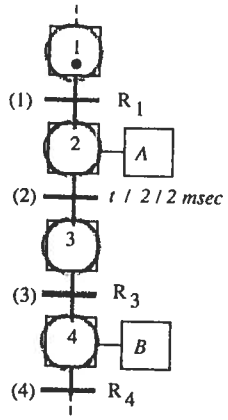

Figure 5.22 Timed action.

R_5 = Temperature greater than 100°C (condition)
R_6 = (Value of counter C_i = reference value) (condition)
R_7 = Mobile is at end of travel to left (condition)
R_8 = Mobile arrives at end of travel to left (event)
R_9 = End of file printing (event)
R_{10} = The start button is pressed for the third time (event)

Figure 5.22 shows an example of a timed action. The level action A lasts exactly 2msec. This is imposed by the receptivity *t/2/2msec*, associated with transition (2).

[EXERCISE 5.13]

Recommendation 5.4 Do not use internal events such as $\uparrow X_i$ in the receptivities. Their interpretation is delicate and may be ambiguous. This has been commented on by M. Moalla and R. David. □

5.4.4 Events and firings

5.4.4.1 Events

In order to avoid all ambiguity, we shall introduce some precise notations of an event which has been presented in a fairly intuitive fashion.

Definition 5.3

a) Let a be a Boolean variable such that $a = 1$ in the time intervals $[t_1, t_2)$ then $[t_3, t_4)$, etc., such that $t_1 < t_2 < t_3 < t_4$. The event $\uparrow a$ occurs at times $t_1, t_3 \ldots$ and the event $\uparrow a'$ occurs at times $t_2, t_4 \ldots$.

b) $\uparrow a.b$ is an *event* which occurs at the same times as $\uparrow a$, each time that $b = 1$ at the corresponding time.

c) $\uparrow a.\uparrow b$ is an *event* which occurs at the times when $\uparrow a$ and $\uparrow b$ occur simultaneously (which is only possible if a and b are not independent, as we shall specify). □

Let E_1 and E_2 be two events. We write $E_1 = E_2$, if E_1 and E_2 are equivalent, i.e. if they always occur at the same times. We write $E_1 = 0$, if E_1 is an event which never occurs. Remember that e denotes the always occurent event.

Hypothesis 5.1 Two *independent events never* occur *simultaneously*. That is to say that $\uparrow a.\uparrow b = 0$, if a and b are two independent variables. □

Hypothesis 5.1 is based on the fact that an event has no duration. Then the probability that two independent events occur at the same time is zero. □

Property 5.1

a) $\uparrow a = \downarrow a'$

b) $\uparrow a.a = \uparrow a, \quad \uparrow a.a' = 0, \quad \downarrow a.a' = \downarrow a, \quad \downarrow a.a = 0$

c) $\uparrow a.\uparrow a = \uparrow a, \quad \uparrow a.\uparrow a' = 0, \quad \uparrow a.e = \uparrow a$

d) If a and b are two independent variables, then:

$$\uparrow(a.b) = \uparrow a.b + \uparrow b.a, \quad \uparrow(a + b) = \uparrow a.b' + \uparrow b.a'$$

e) If a, b and c are three independent variables, then:

$$\uparrow(ab).\uparrow(ac) = \uparrow a.bc$$ □

Proof

a), b) and c): evident according to Definition 5.3.

d) $\uparrow(ab) = \uparrow a.b + \uparrow b.a$: the function (ab) can only move from the value *0* to the value *1* from the state $a = 0$ and $b = 1$ or from the state $a = 1$ and $b = 0$, but not from the state $a = b = 0$, according to Hypothesis 5.1. The property $\uparrow(a + b) = \uparrow a.b' + \uparrow b.a'$ is dual.

e) According to Property 5.1d, we can write:

$$\uparrow(ab).\uparrow(ac) = (\uparrow a.b + \uparrow b.a)(\uparrow a.c + \uparrow c.a) = \uparrow a.b\,c,$$

since

$$\uparrow a.\uparrow c = 0, \ \uparrow b.\uparrow a = 0 \text{ and } \uparrow b.\uparrow c = 0.$$

Property 5.1e shows that the two events $\uparrow(ab)$ and $\uparrow(ac)$, which are not independent since ab and ac depend on the same variable a, can occur simultaneously when a changes from *0* to *1*, if b and c have the value *1* at this time. This is only one example and naturally many others could also be given.

Note that the *event which causes a transition* to fire may *depend on the context.*

Let $R_i = ab$ be the receptivity associated with a transition (i). If $ab = 1$ when (i) becomes enabled, firing takes place on occurrence of event e. If $ab = 0$ when (i) becomes enabled, firing takes place on occurrence of $\uparrow a$ (if $b = 1$ at this time) or on occurrence of $\uparrow b$ (if $a = 1$ at this time).

We now possess all the elements to understanding how the evolution of a grafcet is interpreted in the general case.

5.4.4.2 *Firing of a transition*

A transition (i) whose receptivity is $R_i = E_i . C_i$ is fireable if it is enabled and if the receptivity is 'true' (Section 5.3.1). This receptivity is said **to be true** if condition C_i is equal to *1* and **when** event E_i occurs. It is thus event E_i which triggers firing. A transition (i) is said **to be fireable on occurrence** of E_i if it is enabled and $C_i = 1$.

From a formal point of view, according to Definitions 5.3b and c, *the product of an event and a condition is an event, and the product of two events is an event* (which may never occur). This enables it to be known clearly whether two receptivities may be simultaneously true (which is important for the application of Rule 2, if there are any receptivities R_1 and R_2 in Figure 5.8b).

Let $R_1 = E_1 . C1$ and $R_2 = E_2 . C_2$. Both receptivities are simultaneously true if there is occurrence of the event which is the product $R = R_1 . R_2$, i.e. if $R = (E_1 . C_1)(E_2 . C_2) = (E_1 . E_2)(C_1 . C_2)$ occurs. It is observed that the two receptivities R_1 and R_2 are not simultaneously true if the conditions are incompatible ($C_1 . C_2 = 0$) or if E_1 and E_2 are independent external events ($E_1 . E_2 = 0$). We may also write $R = (E_1 . C_2)(E_2 . C_1)$, which gives still more cases where $R = 0$, e.g. $E_1 = \uparrow a$ and $C_2 = a'$.

5.4.4.3 *Iterated firing*

This notion is illustrated in Figure 5.23. Let us consider the grafcet of Figure 5.23b which indicates the situation at the time t. At this time, $a = 0$ and $b = c = 1$. Situation {3} is stable and consequently cannot change without an external event. The next event will be $\uparrow a$ at the time $t_1 > t$. At this point, transition (3) fireable on occurrence of $\uparrow a$ is fired. Situation {4} is reached. Transition (4) is then fireable since $b = 1$ (formally, it is fireable on occurrence of e which is the always occurrent event) and thus it is fired. Situation {5} is reached. Since transition (5) is fireable on occurrence of $\uparrow c$, we must wait for this event to occur and thus situation {5} is stable. We have moved from the stable situation {3} to the stable situation {5} via the unstable situation {4}. See Figures 5.23c and d.

An iterated firing of transition (3) followed by transition (4) has thus occurred. These *successive firings* are triggered solely by event $\uparrow a$. There is said to be **iterated firing on occurrence** of $\uparrow a$ (firing on occurrence of $\uparrow a$, possibly followed by one or more firings on occurrence of e).

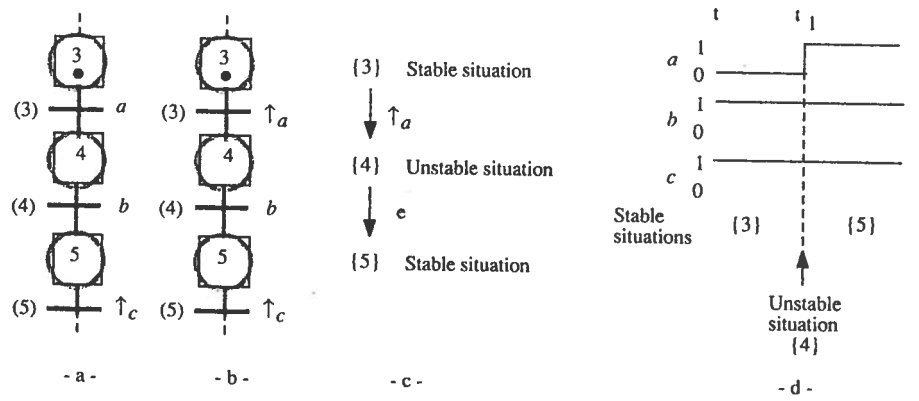

Figure 5.23 Illustration of an iterated firing.

The behaviour corresponding to Figure 5.23a is fairly similar since $a = 0$ at the time t. The receptivities a (Figure 5.23a) and $\uparrow a$ (Figure 5.23b) become true at the same time t_1.

5.5 ALGORITHM FOR GRAFCET INTERPRETATION

A grafcet describes a *logic controller* (which is a *sequential machine*): it specifies *relations between the inputs and outputs*. Using any input timing diagram, the grafcet lets us know which is the timing diagram of the corresponding outputs. The interpretation must thus be without ambiguity. Two persons faced with the **same grafcet** and the **same input timing diagram** must come up with the **same output timing diagram**. This is the aim of the interpretation algorithm given below. All the elements found there have already been presented on simple examples.

5.5.1 Hypotheses

The interpretation algorithm is based on two hypotheses. The first is Hypothesis 5.1 which considers that two uncorrelated **external events cannot be simultaneous** (without this hypothesis the Grafcet model would not be deterministic). The second hypothesis is as follows.

Hypothesis 5.2 A grafcet has the time to reach a **stable situation** between two distinct external event occurrences (in fact the change-over from one stable

situation to another has a zero duration, even if there are several unstable situations between them). □

Remark 5.7 These hypotheses are not new: they correspond to functioning in the **fundamental mode**, which is *classical in the theory of asynchronous sequential systems* (roughly speaking, this means that the processing part, i.e. the process to be controlled, represented in Figure 5.20, is considered to be slower than the logic controller). □

5.5.2 Interpretation algorithm

Algorithm 5.1 Interpretation of the Grafcet

Step 1. Initialization: activation of the initial steps and execution of the associated impulse actions. Go to *Step 5*.

Step 2. When a new external event occurs, determine the set T_1 of the transitions fireable on occurrence of this event. If T_1 is not empty, go to *Step 3*. Otherwise, modify if necessary the state of the conditional actions associated with the active steps (in fact, certain actions may depend on conditions whose values may have changed). Wait for a new external event at *Step 2*.

Step 3. Fire *all* the fireable transitions. If the situation remains unchanged after this simultaneous firing, go to *Step 6*.

Step 4. Carry out all the *impulse actions* associated with the steps having become active at *Step 3* (including initialization of the time delays).

Step 5. Determine the set T_2 of the *transitions fireable on occurrence of event* e. If T_2 is not empty, go to *Step 3*.

Step 6. A stable situation is reached.

Step 6.1. Determine the set A_0 of the *level actions* which must be inactivated (actions associated with the steps which were active at *Step 2* and which are inactive now, and conditional actions associated with the steps having remained active for which the conditions are no longer verified).

Step 6.2. Determine the set A_1 of the *level actions* which must be activated (actions associated with the steps which were inactive at *Step 2* and which are now active, possibly under certain conditions, and conditional actions associated with the steps having remained active for which the conditions are verified whereas they were not so at *Step 2*).

Step 6.3. Set to *0* all the actions which belong to A_0 and do not belong to A_1. Set to *1* all the actions which belong to A_1. Go to *Step 2*. □

5.5.3 Example

Let us consider Figure 5.24. The grafcet is represented in Figure 5.24a. At

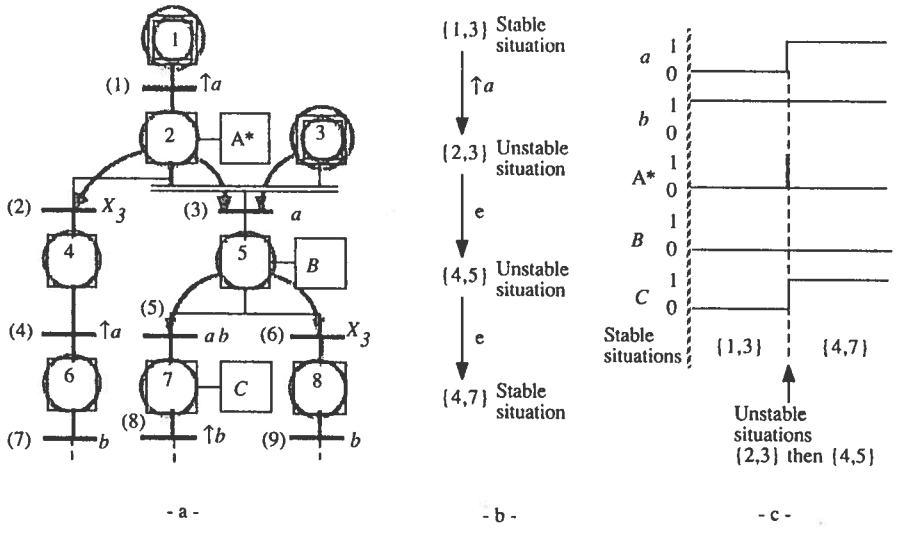

Figure 5.24 Application of Algorithm 5.1.

initialization, we have $a = 0$ and $b = 1$. The first external event occurring is the rising edge of a. The iterated firing on occurrence of this event is represented in Figure 5.24b, and the timing diagram in Figure 5.24c.

Example (Figure 5.24)

Step 1. Initialization: we have the initial situation {1, 3}. No impulse action associated with these steps.

Step 5. Only transition (1) is enabled. It is not fireable on occurrence of e. Thus T_2 is empty.

Step 6. Stable situation {1, 3}. A_0 and A_1 are empty.

Step 2. When event $\uparrow a$ occurs, we have $T_1 = \{(1)\}$.

Step 3. Firing of the transition (1). Situation {2, 3} is reached.

Step 4. The impulse action A* is carried out.

Step 5. The enabled transitions are (2) and (3). We have $X_3 = 1$ (since the current situation is {2, 3} and $a = 1$. Thus $T_2 = \{(2), (3)\}$.

Step 3. Simultaneous firing of (2) and (3). Situation {4, 5} is reached.

Step 4. No impulse action associated with Steps 4 and 5.

Step 5. The enabled transitions are (4), (5) and (6). Transition (5) is fireable on occurrence of e since $ab = 1$. Transition (4) is not fireable on occurrence of e, but on occurrence of $\uparrow a$. The condition associated with transition (6) is not verified since the current situation is {4, 5}. Thus $T_2 = \{(5)\}$.

Step 3. Firing of (5). Situation {4, 7} is reached.

Step 4. No impulse action associated with Step 7.

Step 5. T_2 is found to be empty.
Step 6. Stable situation {4, 7}. A_0 is empty and $A_1 = \{C\}$. Thus the level action C assumes the value *1*. Go to *Step 2* to wait for the next external event. □

The timing diagram summarizes what has happened during this iterated firing (whose duration is infinitely short). We have moved from the stable situation {1, 3} to the stable situation {4, 7}. The impulse action A* has been carried out since Step 2 has moved to the active state (transiently) between the two stable situations. The level action B has remained at *0*, although Step 5 was transiently active. The level action C changes from state *0* to state *1* because Step 7 is active in the stable situation after iterated firing.

Remark 5.8 Section 5.7.2 will provide an example in which the situation may be unchanged by *Step 3* (illustration in Exercise 5.26). □

5.5.4 Comments on the interpretation algorithm

a) The loop *Step 3* → *Step 4* → *Step 5* → *Step 3* enables the grafcet to evolve until a new *stable situation* is reached.
b) An *impulse action* is carried out even if the situation is not stable (at *Step 4*), and a *level action* is not modified before reaching a stable situation (at *Step 6*).
c) A level action which is at *1* in a stable situation and which is still at *1* in the following stable situation remains at *1 without interruption*. Even if this action is no longer associated with the same step, *Step 6.3* ensures continuity.
d) An interpretation is a *theoretical understanding of the behaviour*. This interpretation is based on external events (like the algorithms relative to non-autonomous PNs in Chapter 2). Roughly speaking we can say that, when an external event occurs, the 'time is stopped' and the algorithm takes step. As soon as *Step 2* is reached again, the 'time restarts' up to the next external event.
e) This interpretation algorithm may also be *used for the software implementation* of the Grafcet. Certain practical constraints must then be taken into consideration in order to obtain a behaviour consistent with the interpretation (as in the example given below, where we shall show the need to have reached a stable situation before seeing if any inputs have changed since the last event).

5.5.4.1 Example showing the need to have reached a stable situation before considering new events

This example is presented in Figure 5.25. At initialization, a moving body is found at the position indicated in Figure 5.25a and it sets off both upwards (action H) and to the right (action R). Two cases may occur. If it reaches the right (contact b) before reaching the top (contact a), it stops rising and returns to the left (action L)

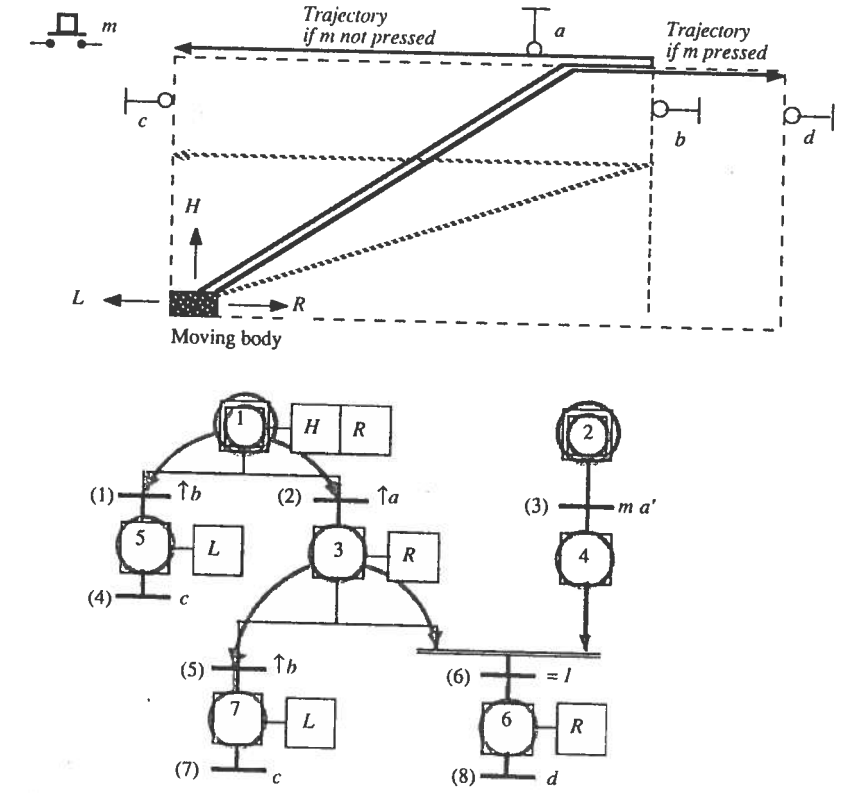

Figure 5.25 Need for research on stability.

up to contact c and stops. If it reaches the top before reaching contact b, it stops rising and two cases may occur: either push button m has been pressed before contact a was reached and the trajectory will continue to the right up to contact d where the moving body will stop, or button m has not been pressed and the moving body will continue up to contact b and then return to contact c (on the left) before stopping. A grafcet meeting this specification is represented in Figure 5.25b.

Assume that m is pressed before reaching contact a and that this contact is reached before contact b, i.e. that we have in order ↑m, ↑a and ↑b. The initial situation is {1, 2}. After pressing m, the situation is {1, 4}. It is then the event ↑a which will cause an evolution. The interpretation algorithm shows that there is a change from the stable situation {1, 4} to the unstable situation {3, 4}, then to the stable situation {6} in which action $R = 1$ until contact d is reached.

Hypothesis 5.3: Implementation with no search for stability
This implementation would correspond to the following algorithm.

Step 1. Reading of the system inputs.
Step 2. Evolution of the situation (a single firing, or possibly simultaneous firings, but no iteration).
Step 3. Carrying out of the actions. Go to *Step 1*. □

Let us now assume that this implementation is made and that we have in order ,↑ *m*, ↑ *a*, ↑ *b* with ↑ *b* following in a very short time the event ↑ *a*. After ↑ *m*, the situation {1, 4} is reached. The following sequence could take place as from when event ↑ *a* occurs.

Step 1. Variable *a* has just changed to *1* and *b* = *0*.
Step 2. Firing of (2). The situation {3, 4} is reached.
Step 3. Action *R* = *1*.
Step 1. Variable *b* has just changed to *1*.
Step 2. Simultaneous firing of transitions (5) and (6). Situation {7, 6} is reached.
Step 3. Actions *L* = *1* and *R* = *1*.

In this case, not only the grafcet would not have a *deterministic behaviour* (since depending on whether the duration between ↑ *a* and ↑ *b* is very small or slightly less small, either the situation {7, 6} or {6} would be reached), but above all it would give rise to *incoherent actions* (possibly dangerous) since in situation {7, 6} actions *L* and *R*, simultaneously occur, i.e. movement to the left and movement to the right! This is absurd, since two contradictory actions are given (the case where it is impossible to distinguish the order in which ↑ *a* and ↑ *b* occur is commented on in Section 5.5.4.3).

It could be admitted that there is *no search for stability*, if all external event occurrences led from any one stable situation to another without an intermediary unstable situation, and grafcets could be implemented so that they exhibited this property. However, it would then be necessary to *sacrifice the concurrency possibilities* provided by the grafcet. Indeed, iterated firing often occurs when there is a junction which is preceded by stand-by steps. As soon as the last of these steps is marked, an unstable situation is reached (and thus an iterated firing). Thus, in the grafcet in Figure S.5.18 (solution to Exercise 5.18), that we shall call G_1, there are two successive junctions with receptivities equal to *1*. This may result in an iteration of three firings. The solution proposed for this exercise is naturally worked out from the specifications. If we wished to avoid the iteration of firings, a grafcet G_2 would have to be constructed, certain steps of which would have the following property: a step of the grafcet G_2 would be associated with a situation corresponding to several active steps of grafcet G_1. The grafcet G_2 would be more like a state diagram and would have more states.

Implementation
In the *general case*, the *search for stability* (Algorithm 5.1) is indispensable for a correct implementation of the Grafcet. A correct behaviour may be obtained without search for stability in certain special cases, notably if the *use* of the Grafcet *is restricted* by prohibiting the possibility of unstable situations.

Interpretation
The algorithm without research of stability *cannot be an interpretation* (i.e. a theoretical understanding of the behaviour) of the Grafcet. As a matter of fact, let δ be the total duration of the *Steps 1*, *2* and *3* of the algorithm. If δ is assumed to have a finite value, then some event can occur during this time, and the behaviour may be different if this event would not occur (interpretation is not deterministic). If δ is assumed to be zero or to have a infinitely small value, then it may be considered that no event can occur during δ. It follows, unfortunately, that the number of cycles (*Steps 1*, *2* and *3*) between two successive external events is infinite (in other words *Steps 1*, *2* and *3* are infinitely performed without reading a new external event).

5.5.4.2 Unstable cycle

The interpretation algorithm is based on the assumption that a finite number of iterations results in a stable situation. This is nearly always the case. But what if there were an unstable cycle, anticipated or not by the designer? Certain authors, taking this assumption as their base, recommend the use of an algorithm without search for stability (despite the considerable disadvantages that we have shown by means of an example). *Two solutions* to this problem may be considered.

The *first solution* is a **modification of Algorithm 5.1** which consists in memorizing all the unstable situations from the initial situation (at *Step 2*). If an unstable situation already encountered is reached, with the same internal state (e.g. same counter value), this is an unstable cycle. All the steps are known which may be active during the cycle and the conclusions are drawn concerning the possible evolutions depending on the external events. All this is rather complicated.

The *second solution* consists of **modifying the grafcet** so that there are no unstable cycles. This is *always possible if the grafcet* having an unstable cycle *has a deterministic behaviour*. In general, an unstable cycle is due to an error that the designer must quickly rectify. However, this unstable cycle may also sometimes have been created on purpose. Let us consider the example presented in Figure 5.26.

Example Figure 5.26a presents a scanning system which is sensitive to the two variables *a* and *b*. When *a* = *b* = *0*, the system evolves according to the cycle of unstable situations {1} → {2} → {1} etc. When variable *a* changes to value *1* (the behaviour is symmetrical for variable *b*), the system changes into the stable situation {3} in which there is the level action *A*. When *a* changes back to *0*, the system returns to the unstable cycle (possibly only to leave it again at once if *b* has moved to *1* between time). The grafcet of Figure 5.26a may be replaced by the

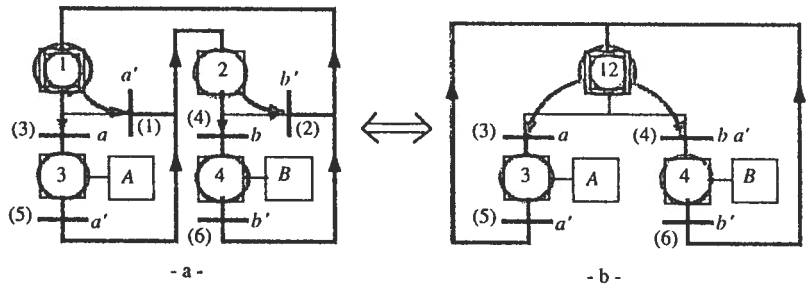

Figure 5.26 Equivalence of grafcets with and without unstable cycle.

grafcet of Figure 5.26b, which describes exactly the same input/output behaviour and which has no unstable cycle. Cycle {1} – {2} of Figure 5.26a forms a *stationary* situation which is replaced by Step 12. The receptivity $R_4 = b\,a'$ (instead of $R_4 = b$) may be necessary if $b = 1$ at initialization. □

Recommendation 5.5 Do **not** construct grafcets which may have **unstable cycles**. This is always possible if the input/output behaviour which we wish to describe is deterministic. □

5.5.4.3 Apparent simultaneity of events

Two independent events cannot take place simultaneously. However, when a logic controller is implemented, the two events may sometimes be so close together that *there is no technological means to distinguish* which took place first. For example, when Step 1 is active in Figure 5.25b, the evolution varies according to the order in which events $\uparrow a$ and $\uparrow b$ occur. If the logic controller implemented 'sees' these two events taking place simultaneously, it has to choose, possibly arbitrarily, to fire either transition (1) or transition (2), but not both of them. In fact, the grafcet clearly indicates that only one of the transitions must be fired, in accordance with the hypothesis of non-simultaneity of events. Simultaneous firing would be **inconsistent** with the specification given by the grafcet (and the situation would occur in which there was movement to the left and to the right at the same time, impossible since these actions are contradictory). Priority may be given to transition (1) over transition (2) for example, by replacing receptivity $\uparrow a$ by the **redundant receptivity** $\uparrow a.b'$. Consequently, in the event of *apparent simultaneity* of $\uparrow a$ and $\uparrow b$, only transition (1) will be fired.

[EXERCISES 5.14 TO 5.19]

5.5.5 Comparison with interpreted Petri nets

Let us first look at the **points common to the interpreted PNs and the Grafcet.**

Each of the models possesses two types of nodes, namely the *steps* and the *transitions* for the Grafcet, and the *places* and *transitions* for the interpreted PN. The original graphical representation of the Grafcet is similar to the one of the interpreted PNs. In both models, the evolutions of the markings are *synchronized* on the occurrences of *external events*.

Let us now look at the **differences between the interpreted PN and the Grafcet.**

Difference 1. The *marking* of a grafcet step is Boolean whereas the marking of an interpreted PN place is numerical.[1]

Difference 2. In a grafcet, all the simultaneously fireable transitions are *simultaneously fired* (see Section 5.2.3). On the other hand, if several transitions are fireable in an interpreted PN, a sequence known as CFS (complete firing sequence, Section 2.1.2) is fired. If this CFS is not maximal, all the fireable transitions are not fired.

Difference 3. The *conditions* used in a grafcet can depend on the state of the marking, which is not the case in an interpreted PN.

The dissimilarities relating to the emission of outputs to the environment, or conditions coming from this environment, are not really differences, but rather different possible interpretations (which we have, moreover, done away with in the definition of a control-interpreted PN).

Differences 1 and 2 are fundamental, but difference 3 is not very important since the same result can be reached using the interpretation. This is illstrated in Figure 5.27. Figure 5.27a represents a grafcet with a receptivity X_6 for transition (1). This means that this transition will be fired when Steps 1 (enabling) and 6 (condition) are active. A similar result may be obtained with a safe interpreted PN, without modifying the graph (Figure 5.27b), by associating a condition Y to the firing of T_1. The variable Y is set at the value 1 when a token is deposited in P_6. It is reset to 0 when the token is removed from P_6, which corresponds to the firing either of T_7 or T_8, to the deposition of a token either in P_7 or P_8.

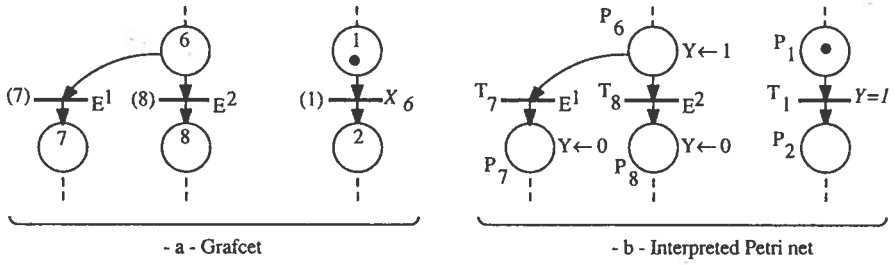

Figure 5.27 Condition depending on the marking.

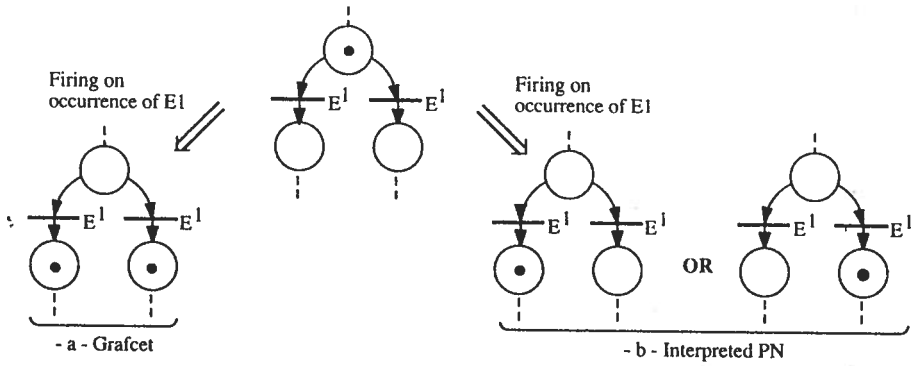

Figure 5.28 Simultaneous firings.

Impact of the differences 1 and 2
There is a difference between Grafcet and an interpreted PN in three cases which are illustrated by the simple examples shown in Figures 5.28 to 5.30. These cases are named in accordance with the behaviour of the Grafcet.

Case 1. Simultaneous firing (Figure 5.28)
Two firings are simultaneously fireable but enabled by a single token. In a grafcet, both transitions are simultaneously fired, whereas either one or the other is fired in an interpreted PN.

Case 2. Reactivation (Figure 5.29)
A grafcet step is active, and the firing 'reactivates' it. This step remains active (the marking is Boolean). In an interpreted PN, the same situation results in the deposition of a second token in the place which already contained one.

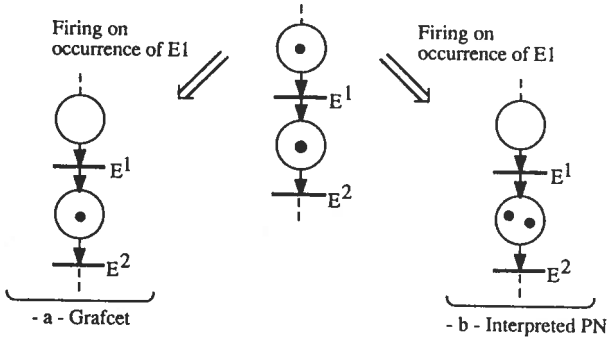

Figure 5.29 Reactivation.

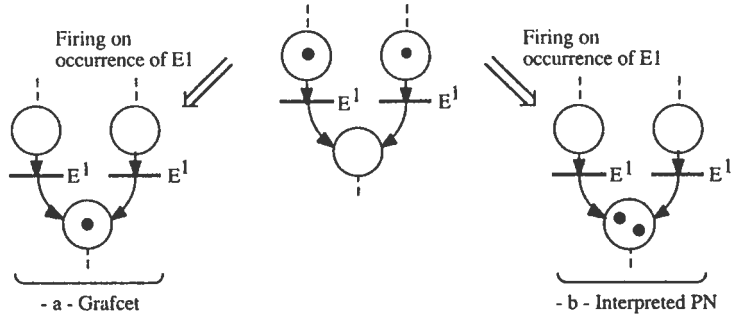

Figure 5.30 Simultaneous activation.

Case 3. Simultaneous activation (Figure 5.30)
When two transitions having a common output step/place are simultaneously fired (in a grafcet) or in the same CFS (in an interpreted PN), the corresponding place is simply active in a grafcet, but contains two tokens in an interpreted PN.

The three cases which have just been presented and which represent all the differences in marking evolutions of a grafcet compared with an interpreted PN are important to the extent that these behaviours mean that the grafcets, in the general case, do not have any marking invariants (and we remember that these invariants correspond to the only properties of autonomous PNs which are preserved in a synchronized PN, therefore in an interpreted PN). Figure 5.28 corresponds to a token which 'divides into two', whereas Figures 5.29 and 5.30 correspond to tokens which 'merge together' for the grafcet. Thus, a *grafcet is equivalent to a Petri net if none of these three cases can occur*. This is the reason for the definition of a *sound grafcet*.

Definition 5.4 A **sound grafcet** is characterized by the following three properties:

1. At no moment can an active step be such that one of its input transitions is fireable, and none of its output transitions is simultaneously fireable (we say that the grafcet is not reactivable).
2. For each *actual conflict* $\langle P_1, \{T_1, T_2, \ldots\} \ldots\rangle$ which may exist, there are arcs $T_1 \rightarrow P_1$, $T_2 \rightarrow P_1 \ldots$.
3. No step can have two simultaneously fireable input transitions (except Case 2 above). □

Remark 5.9 A *sound grafcet*[2] is not a new model which would not respect the rules of the Grafcet such as they have been presented. It is rather a grafcet which possesses certain properties. And if it has these properties, then none of the three cases which could distinguish it from an interpreted PN can occur. □

A grafcet such as an interpreted PN represents a sequential machine (in the mathematical sense). *Two sequential machines are said to be equivalent* if for every input sequence, they produce the same output sequence.

Property 5.2 A *sound grafcet* is *equivalent* to *a control interpreted PN.* □

The meaning of this property is as follows: namely a net of places (or steps) and transitions possessing an *interpretation of the Grafcet type* (level and impulse actions associated with the places/steps, events and conditions associated with the transitions), for example the one in Figure 5.31 If this net is a control interpreted PN or sound grafcet and *it is interpreted like an interpreted PN or a grafcet, the same input/output behaviour is deduced from it.*

What are the *practical restrictions* of a control interpreted PN and a sound grafcet compared to the general cases? The only important restriction that a control interpreted PN presents compared with an interpreted PN is that it must be *safe* (the determinism that is imposed by the definition is sought to describe a logic controller, and we have seen that the timing of places is not indispensable to describe a logic controller). In theory, describing a logic controller by a sound grafcet involves no restriction. It is generally sufficient to follow *Recommendation 5.1* (Section 5.2.3) in the construction of a grafcet for it to present this quality, thus Petri net properties.

Property 5.2 is undoubtedly interesting from a theoretical point of view. However, how can we in practice verify that a grafcet is a sound grafcet or that an interpreted PN satisfies the conditions required to be a control interpreted PN?

Algorithm 5.2 Verification that an interpreted PN is a control interpreted PN and that a grafcet is sound

Step 1. We can easily observe that it is not P-timed.

Step 2. The hypothesis is taken that the net is a PN and we prove:

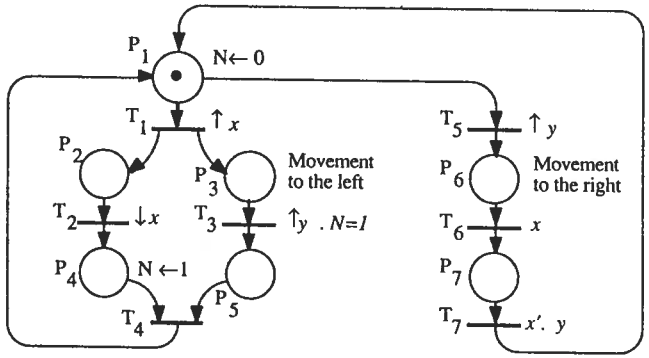

Figure 5.31 Example: control interpreted PN and sound grafcet.

Step 2.1. That it is safe

Step 2.2. That for every pair (P_i, P_j) of places with which incompatible operations or actions are associated, we have a marking invariant $M(P_i) + M(P_j) + \ldots = 1$.

Step 3. We verify that for all structural conflicts $\langle P_1, \{T_1, T_2\}\rangle$, the receptivities R_1 and R_2 are exclusive. □

If the net satisfies these conditions, then it is a control interpreted PN and a sound grafcet. These conditions are sufficient without always being necessary. They are sufficient because conditions 1, 2 and 3 of Definition 2.8 are verified. Condition 1 is verified at *Step 2.1*. Condition 2 is verified at *Step 1*. Condition 3 is verified thanks to *Steps 2.2* and *3* (in accordance with Property 2.9).

The concept of compatible and incompatible operations has been specified in Section 2.3.3. This is easily generalized to level actions. For example, the actions *movement to the left* and *movement to the right* are incompatible for the same object, whereas *movement to the left* and *movement upwards* are compatible. The conditions in which receptivities are exclusive have been specified in Section 5.4.4.1 for the Grafcet. This is similar for an interpreted PN since an event and a condition are associated with each transition.

Let us apply this verification to the net (interpreted PN or grafcet?) of Figure 5.31:

Step 1. The net is not timed.

Step 2. It is assumed to be a PN. Two marking invariants are found:

$$I_1 = M(P_1) + M(P_2) + M(P_4) + M(P_6) + M(P_7) = 1$$
$$I_2 = M(P_1) + M(P_3) + M(P_5) + M(P_6) + M(P_7) = 1$$

Step 2.1. Each place is contained in at least one of these marking invariants, thus the PN is safe.

Step 2.2. Two incompatible actions ($N \rightarrow 0$ and $N \rightarrow 1$) are associated with places P_1 and P_4. These two places form part of the invariant I_1.

Two incompatible actions (movements to the left and to the right) are associated with places P_3 and P_6. These places form part of the invariant I_2.

Step 3. A single structural conflict $\langle P_1, \{T_1, T_5\}\rangle$. The receptivities associated with the transitions T_1 and T_5 cannot be simultaneously true (they are two external events).

Conclusion

Figure 5.31 corresponds both to a control interpreted PN and to a sound grafcet. This means that whatever the evolution of inputs x and y may be from the initial marking indicated, the evolution of the marking and of the impulse ($N \rightarrow 0$ and

N → 1) and level actions (movement to the left and right) are exactly the same, whether this net is interpreted as a grafcet (Algorithm 5.1, Section 5.5.2) or as an interpreted PN (Algorithm 2.5, Section 2.3.2).

[EXERCISE 5.20]

5.6 MACROSTEPS AND PSEUDOMACROSTEPS

The aim of the **macrostep** and **pseudomacrostep** concepts is to facilitate the description of complex systems. The macrostep makes it possible to lighten the graphical representation of a grafcet by detailing certain parts separately. The pseudomacrostep is useful when the designer roughly outlines a grafcet, before the latter is complete. These concepts have been introduced by M. Moalla and R. David. The macrostep concept has been taken up by AFCET, which has modified it slightly.

5.6.1 Macrostep

The concept of macrostep is illustrated in Figure 5.32. A macrostep is represented by a *circle surrounded by a rhombus*, in the original graphical representation, and by a *square divided into three parts* by two horizontal lines in the standardized graphical representation. A macrostep given as 5/M30 is represented in Figure 5.32a. It represents a grafcet part which is detailed elsewhere and which is known as a **macrostep expansion**. Figure 5.32b is the expansion corresponding to M30. If this macrostep expansion is used to replace macrostep 5/M30, the grafcet of Figure 5.32c is obtained.

The complete set of Figures 5.32a and b (grafcet with a macrostep, plus expansion of the macrostep) is strictly equivalent to Figure 5.32c in which the macrostep has been replaced by its expansion.

Definition 5.5 A **macrostep** and its **expansion** satisfy the following rules:

1. A macrostep expansion has only one *input step* (written as I) and one *output step* (written as O).
2. All firings of a *transition upstream* of the macrostep activate the input step of its expansion.
3. The output step of the macrostep expansion participates in the enabling of the *downstream transitions*, in accordance with the structure of the grafcet containing this macrostep.
4. *No directed links* either join or leave the macrostep expansion. □

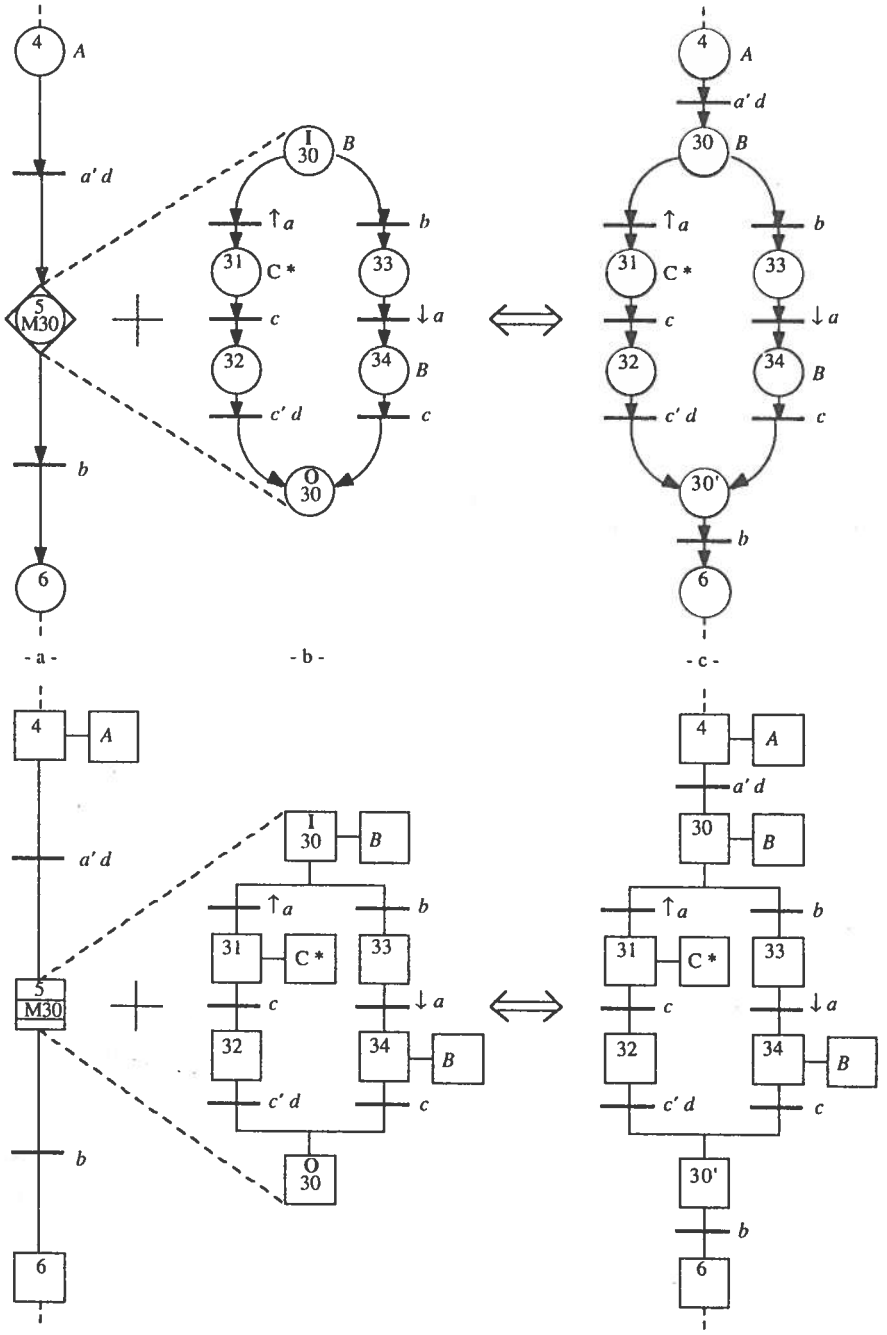

Figure 5.32 Macrostep, original and standardized graphical representations. (a) Grafcet with macrostep. (b) Macrostep expansion. (c) Equivalent grafcet without macrostep.

Let us now return to Figure 5.32 to give further details on certain points. The grafcet of Figure 5.32a contains a macrostep which is written as 5/M30. Number 5 is the *macrostep number*, i.e. it indicates its position in the grafcet. Symbol M30 relates to the *macrostep expansion*. Number 30 is found in the expansion to mark the input step, I30, and the output step, O30, of expansion M30. When the grafcet part corresponding to expansion M30 has been used to replace macrostep 5 (i.e. in Figure 5.32c), the symbols M, I and O have disappeared. We now have an *ordinary* grafcet in which every step has a number. During replacement, Steps I30 and O30 have become 30 and 30′ (any other numbering would have been possible, provided that no two grafcet steps have the same number).

5.6.1.1 *Obtaining a grafcet without a macrostep*

Starting out from a grafcet with macrosteps and corresponding expansions, a grafcet without a macrostep can be obtained in a number of ways. One of these is the only one which *functions correctly in every case*. Another way, with variants, can only be used if it is impossible that a transition upstream of the macrostep in question be fired while the macrostep is active (i.e. that one of its steps is active). It is said that the macrostep *cannot be 'reactivated'*. If a grafcet with macrosteps, in which each macrostep is considered to be an ordinary step, is a *sound grafcet* (this notion has been studied in Section 5.5.5), it then has the property that none of its macrosteps can be reactivated.

General method

The **expansion is used to replace** the macrostep as has been shown in Figure 5.32 Figure 5.33a represents another grafcet using the same macrostep expansion M30. The replacement results in Figure 5.33b in which it may be observed that all the directed links joining macrostep 8 reach the input step written as 30, and that all the directed links leaving macrostep 8 leave the output step written as 30′.

Special method

(only used for a macrostep which cannot be reactivated)

First variant. As we have already seen on several occasions, the same specification can be represented by several grafcets, in particular by a grafcet made up of **several disconnected parts** (see Figure 5.13c). Using the rules relating to the macrosteps, grafcet 2 of Figure 5.33c may also be obtained, which is made up of two disconnected parts. The *first part* is identical to the grafcet with a macrostep with the following two modifications: macrostep 8 becomes an ordinary step which has the same number, and the receptivities of the steps enabled by the macrostep are modified by adding a condition which is the active state of the final step of the macrostep (e.g. receptivity $R_4 = b\,d'$ in Figure 5.33a becomes $b\,d'\,X_{30'}$ in Figure 5.33c). The *second part* corresponds to the expansion of the macrostep and is modified as follows. The input step is preceded by a transition (30) whose

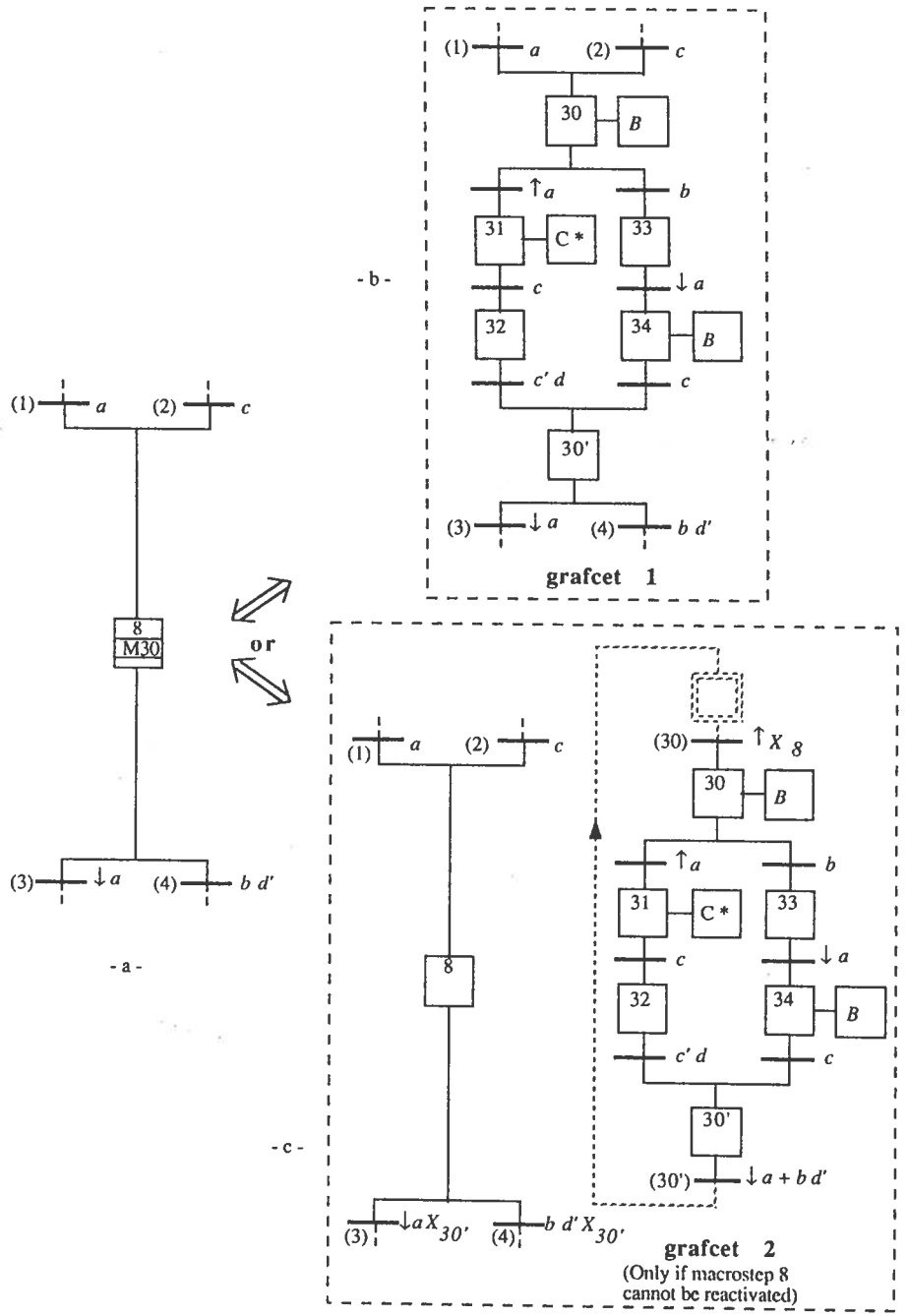

Figure 5.33 Obtaining a grafcet without a macrostep.

receptivity is $\uparrow X_8$. Since this transiton has no upstream step (the dotted line part is not considered for the time being), it is always enabled (it is a *source* transition). Consequently, as soon as Step 8 becomes active, this transition will be fired and Step 30 will become active. The output step is followed by a transition (30′) whose receptivity is true if and only if the conditions for firing one of the transitions downstream of macrostep 8 are satisfied. It is therefore the sum of the receptivities, if there are no other steps upstream of these transitions. In our example this receptivity is $\downarrow a + bd'$ which is the sum $R_3 + R_4$. If these transitions (3) and (4) had other upstream steps, these would have to be taken into account when calculating receptivity $R_{30'}$. Transition (30′) has no downstream step. Its firing inactivates Step 30′ and does not activate any other step. It is a *sink* transition.

Second variant. Grafcet 2 can be modified by adding an **initial step** between transition (30′) and transition (30). This is shown by a dotted line in Figure 5.33c. The behaviour described is strictly the same as before, since once transition (30) has been fired, another event $\uparrow X_8$ cannot occur before transition (30′) is fired. Indeed, firing of transition (30′) coincides with firing either of transition (3) or transition (4), i.e. it coincides with event $\downarrow X_8$. However, if this initial step is added, either event $\uparrow X_8$ or condition X_8 can be used for receptivity R_{30} (condition X_8 would result in incorrect functioning in the event that there is a *source* and a *sink* transition since Step 30 would be active for all the time that Step 8 is active).

5.6.1.2 *Expansion common to several macrosteps*

Figure 5.34 shows a grafcet with two macrosteps, 4 and 6, which have the same expansion M50. Figure 5.35 shows two grafcets without a macrostep which are equivalent. Grafcet 1 **uses expansion M50 twice.** The numbering must be changed so as *not to have two steps with the same number*. Steps 150, 151 and 150′ have been made to correspond to macrostep 6 (any other form of numbering would be

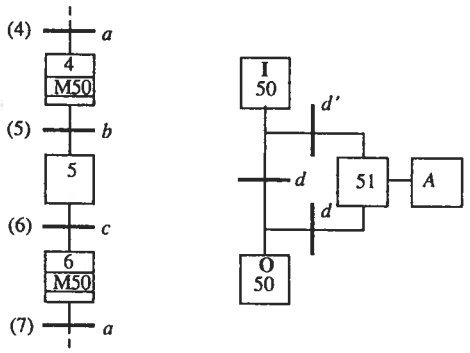

Figure 5.34 Expansion common to several macrosteps.

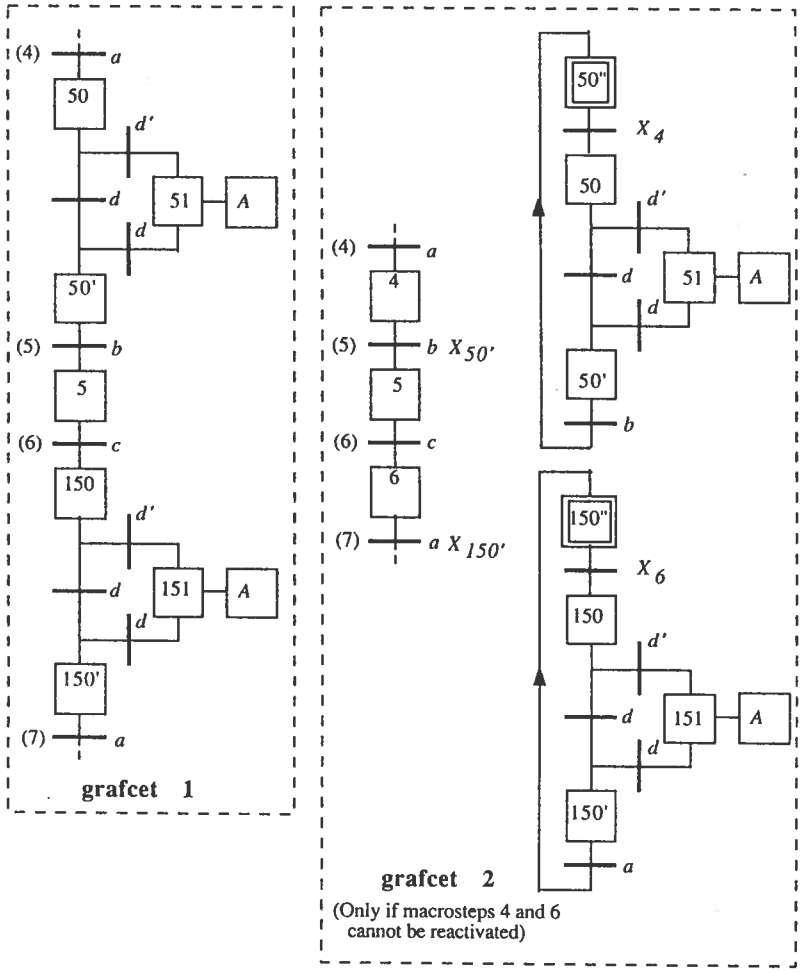

Figure 5.35 Two possible grafcets, without a macrostep, for the example of Figure 5.34.

acceptable). Grafcet 2 (which is only a correct solution if neither of the two macrosteps can be reactivated) is made up of *three parts* which are not connected: one which is associated with the grafcet having macrosteps, and two which are associated with the two macrosteps 4 and 6. Care has also been taken not to have two steps with the same number in grafcet 2.

[EXERCISES 5.21 TO 5.23]

5.6.2 Pseudomacrosteps

A pseudomacrostep is represented by a *rhombus* in the original graphical representation. We propose it be represented by *a square whose vertical sides are not whole*, as shown in Figure 5.36, in order to obtain a symbol consistent with the standardized graphical representation.

Figure 5.36 Pseudomacrostep.

The pseudomacrosteps are intended to assist the designer in a summary description, even if all is not yet clear. The macrostep, satisfying exact criteria, is not always suitable in this rough outline phase of the grafcet. The presence of a **pseudomacrostep** means that the **grafcet** is **incomplete**. The pseudomacrostep represents a part of the grafcet which is poorly defined for the time being. It permits a grafcet *skeleton* to be described which must then be developed in order to obtain a complete grafcet.

Definition 5.6 A **pseudomacrostep** is such that:

1. It represents a *set of steps*.
2. All the *directed links do not necessarily reach the same step* (thus the input step is not necessarily unique).
3. All the directed links leaving a step *do not necessarily leave the same step* (so the output step is not necessarily unique).
4. All the directed links reaching or leaving the pseudomacrostep, are *not necessarily represented*. □

Remark 5.10

a) A pseudomacrostep thus *cannot* generally be *defined by a separate subgrafcet* (unlike a macrostep).
b) The diagram containing a pseudomacrostep is *not a grafcet* in the strict sense of the term. It can be said to be a **pseudografcet**. Precisions will need to be made before a real grafcet is obtained (which may or may not have macrosteps, but will not have pseudomacrosteps). □

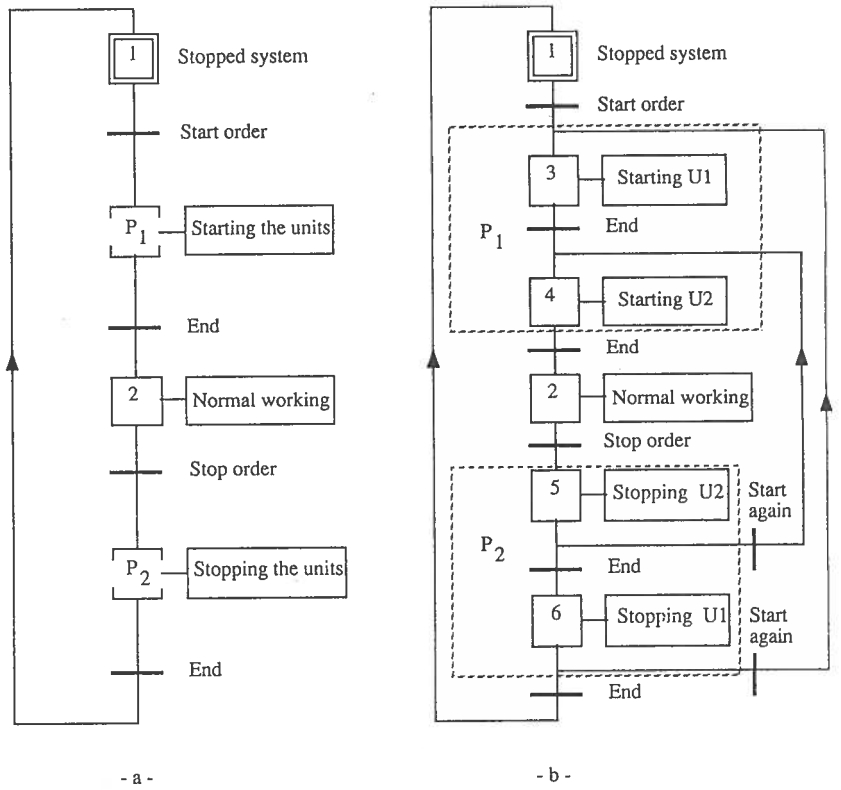

Figure 5.37 (a) Representation with pseudomacrosteps. (b) Grafcet specified without pseudomacrosteps.

Example Figure 5.37 gives a description example using pseudomacrosteps in a rough outline phase. We describe the sequencing relating to the start-up and stopping phases of a set of two units U_1 and U_2 with a possible restart during the stopping phase. Normally, a start-up order causes the units to start and, at the end of this phase, a state of normal working is reached. When the stopping order is given, there is return to the initial state after the stopping phase. These data already enable the pseudografcet of Figure 5.37a to be described before going into greater detail. Links are missing since the possibility has been indicated of a restart during the stopping phase.

First unit U_1 is started followed by U_2, with stopping in the reverse order. If a restart is decided on while U_2 is in the stopping phase, the start-up of U_1 is not necessary. We end up with Figure 5.37b. The following may be noted:

a) All the *directed links do not exist* in Figure 5.37a. They will appear by specifying the operation.
b) The pseudomacrostep P_1 *has several input steps* and the pseudomacrostep P_2 *has several output steps* (they thus cannot be modelled by macrosteps).
c) In this example, it has been assumed that the start-up and stopping phases of each unit correspond to a single step. They may, however, be more complicated. If each of these phases has a single input and a single output step, then it may be modelled by a macrostep. □

[EXERCISE 5.24]

5.7 MACROACTIONS

When describing complex systems, the *size of the grafcets* may increase so that they become difficult to work out and thus to understand, correct, update, etc. Taking the safety devices into account, in particular, is an important reason for increasing complexity if we wish to treat them like other parameters, whereas they have a different role, which is both less frequent (we hope!) and has a greater priority. The concept of **hierarchy** naturally springs to mind. It is easy to imagine that a logic controller (described by a grafcet) *has a global influence* on another logic controller (described by another grafcet). By abuse of language, *a grafcet is said to have a global influence* on another, this being known as a **macroaction**.

A *macroaction* may be *a level* or an *impulse action* (just like an ordinary action). It is produced by a grafcet G_1 and has an effect on the behaviour of a grafcet G_2. A macroaction is thus *homogeneous with an action* from the point of view of G_1.

The concept of macroaction has been introduced in the thesis of H. Deneux, who has defined the macroactions *inactivate*, *initialize*, *mask*. If a grafcet G_1 can inactivate and initialize another grafcet G_2, G_1 is said to *govern* G_2. The group GREPA has specified the notions of *forcing* and *freezing*. The notion of forcing includes and generalizes those of inactivation and initialization. We shall present a synthetic and rather extrapolated view of these various works.

5.7.1 Specifications of example 6

The system to be described has seven Boolean inputs, as shown in Figure 5.38, and produces five actions including one impulse action.

1. As from the initial state, as soon as event $\uparrow a$ occurs, the level action A is set to state *1*, together with action B if $b = 0$ or C if $b = 1$. As soon as variable b assumes value *1*, action B is stopped and action C started. Then, when $ab' = 1$, actions A and C are stopped and the impulse action D* is performed. Finally, on the rising edge of c, there is return to the initial state.

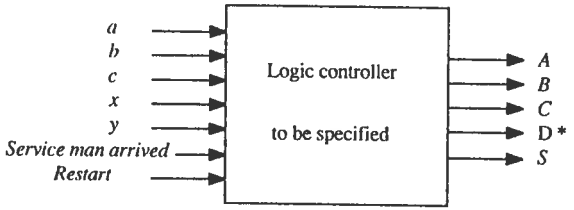

Figure 5.38 Inputs and outputs of example 6.

2. The operating of this system may very well be disturbed by the appearance of *two faults* assumed not to be able to occur simultaneously. The Boolean variables x and y indicate the presence of the first and second faults, respectively. When one of the faults appears, all the actions previously defined are reset to zero ($A = B = C = 0$ and action D* prohibited), and an acoustic alarm is set to *1* (Boolean variable S). When the fault has cleared, operating starts up again, but not necessarily at the initial state. In the case of the *first fault*, there is restart at the initial state. In the case of the *second*, there is restart in the first state after the initial state (i.e. the state reached as soon as event $\uparrow a$ occurs). In both cases, the acoustic alarm stops when the fault has cleared.
3. When the service man arrives, it will be possible to stop the acoustic alarm before the fault clears, by pressing a button which produces the Boolean variable known as *service arrived*.
4. If the first fault occurs twice in less than a minute, the fact that this fault clears will not be enough for operating to resume as indicated in number 2 above). A manual intervention will also be required to authorize restart (change-over from *0* to *1* of the Boolean variable *restart*).

The reader feeling brave enough may undertake to describe this system by a grafcet using the means already presented. This is rather complicated. We shall now present some *macroactions* which enable us to do this more easily.

5.7.2 Macroactions

We shall present several types of macroactions. In the same way as a macrostep, a macroaction should allow an easier description of a sequential machine and of the specifications of a logic controller, but *must under no circumstances be in contradiction with the Grafcet rules*. We have concentrated on showing that a grafcet without macrosteps (and even several) can always be made to correspond to a grafcet with macrosteps. In the same way, we shall show that *a set of grafcets without macroaction*, and even a single grafcet, *can always be made to correspond to a set of grafcets with macroactions*. This can even be carried out algorithmically, but is only of interest from a theoretical point of view. Thus **the macroactions** just like

the macrosteps **do not increase the** specification power (i.e. do not enable systems to be described which it would be impossible to describe without that), but they *increase the specification facility* (it is different for a pseudomacrostep which is characteristic of a grafcet which is not complete).

Just like an action, a macroaction can be either level or impulse. In order to facilitate characterization, we shall use a verb in the infinitive in order to indicate an impulse macroaction, for example:

Force: impulse macroaction
Forcing: level macroaction

5.7.2.1 *Force*

This macroaction is illustrated in Figure 5.39. Figure 5.39a represents a set of two grafcets, G_1 (partial representation) and G_2. Grafcet G_1 produces the **impulse macroaction** *force* G_2 {12} which has the following meaning: *place grafcet* G2 *in situation* {12} *irrespective of the current situation of this grafcet*. The grafcet set $\{G_1, G_2\}$ is equivalent to the grafcet set $\{G_1', G_2'\}$ which does not have recourse to macroactions (Figure 5.39b). Grafcet G_1' emits an impulse action F* which is an input acting on grafcet G_2'. It is observed that in grafcet G_2', Step 12 is activated by the firing of a source transition whose receptivity is F*, and the other steps are inactivated by sink transitions which are receptive to F* (the arrows indicated on the horizontal parts of directed links are redundant, but they facilitate understanding). Since the macroaction *force* and action F* are impulse, grafcets G_2 and G_2' can immediately evolve. Let us assume that $a = b = 1$ when F* is emitted in Figure 5.39b. It can be observed that the only stable situation of G_2' is situation {13}. When F* occurs, the iterated firing {13} → {12} → {13} will take place (this can be verified by applying Algorithm 5.1). The only essential difference between Figures 5.39a and b is that in Figure 5.39a, there are three fewer transitions and three fewer directed links! The set of the two grafcets $\{G_1', G_2'\}$ can now be replaced by a single grafcet G_{12} presented in Figure 5.39c. The impulse action F* is replaced by event $\uparrow X_4$ and this functions exactly as Figure 5.39b.

Remark 5.11 Figure 5.39c does not respect Recommendation 5.4 (Section 5.4.3) since it uses the internal event $\uparrow X_4$. However, we can point out that this is a special case for two reasons: $\uparrow X_4$ is only used for source or sink transitions, and Step 4 is in a part of the grafcet which is not connected with the part of the grafcet where event $\uparrow X_4$ is used. Moreover, the grafcet of Figure 5.39c can be replaced by another grafcet which does not use internal events. This transformation is proposed in an exercise. □

[EXERCISE 5.25]

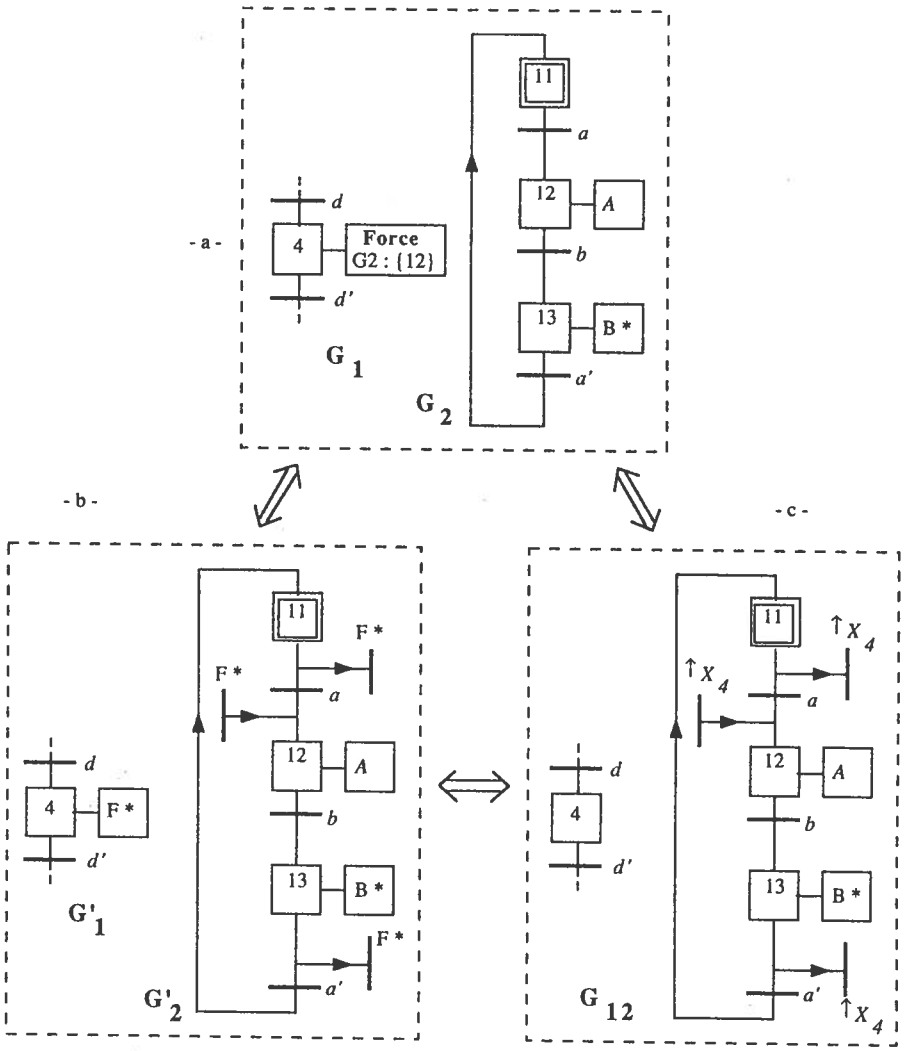

Figure 5.39 Illustration of the macroaction *force*.

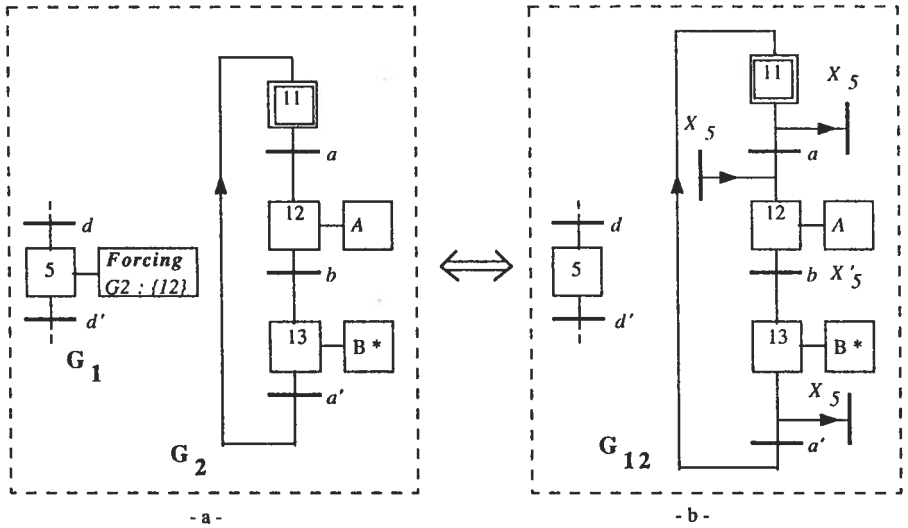

Figure 5.40 Illustration of the macroaction *forcing*.

5.7.2.2 *Forcing*

This macroaction is illustrated in Figure 5.40. The macroaction *forcing* is a **level macroaction**, i.e. it is maintained as long as Step 5 in Figure 5.40 remains active. The set of the two grafcets G_1 and G_2 of Figure 5.40a is equivalent to grafcet G_{12} of Figure 5.40b. It is observed in this latter that Step 12 is forced to the active state by a source transition, and that Steps 11 and 13 are forced to the inactive state by sink transitions, these three transitions having for receptivity the internal variable X_5. Firing of the transition downstream from Step 12 is prohibited by the variable X_5 in the receptivity. It follows that when $X_5 = 1$, Step 12 is active, while Steps 11 and 13 are inactive. Grafcet G_{12} is an example in which a simultaneous firing at Step 3 of Algorithm 5.1 (Section 5.5.2) may not modify the situation. The reader may verify this by completing Exercise 5.26.

[EXERCISES 5.26 AND 5.27]

5.7.2.3 *Freezing*

This macroaction is illustrated in Figure 5.41. In Figure 5.41a, when Step 6 of grafcet G_1 is active, the grafcet G_2 is 'frozen', i.e. it stops evolving. The grafcet set $\{G_1, G_2\}$ has a behaviour equivalent to grafcet G_{12} of Figure 5.41b. This figure

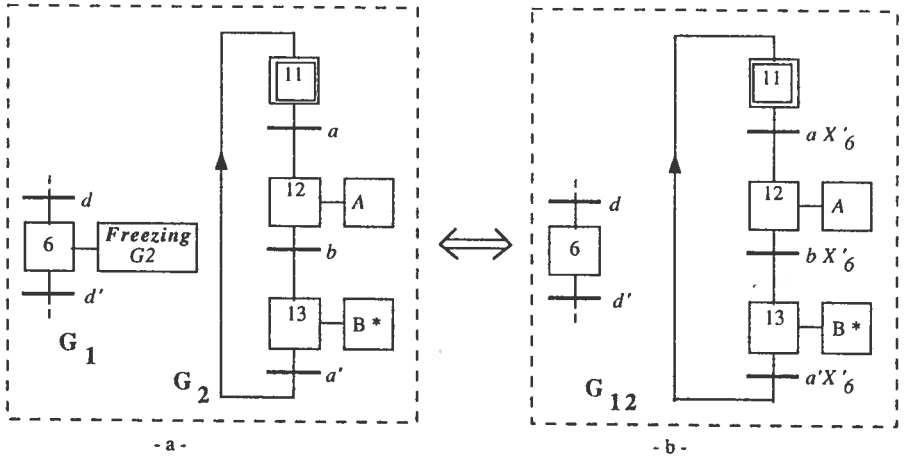

Figure 5.41 Illustration of the macroaction *freezing*.

shows that all the transitions have a receptivity which contains X'_6 as a factor. That is to say that when $X_6 = 1$, all the receptivities on the right-hand part of grafcet G_{12} are zero. Therefore this part does not evolve. The macroaction freezing is a **level macroaction** and lasts as long as the step (or steps) with which it is associated is active.

An impulse macroaction ***freeze*** may also be considered which prevents the frozen grafcet from evolving as long as the reverse order is not given. It is thus necessary to have an impulse macroaction which ends the freezing, and which can be expressed by ***release***. In other words, the set of orders *freeze* G at time t_1 and *release* G at time $t_2 > t_1$ would be equivalent to the macroaction *freezing* G between the times t_1 and t_2.

5.7.2.4 *Masking*

This **level macroaction** is illustrated in Figure 5.42. Grafcet G_1 of Figure 5.42a produces the macroaction *masking* G_2 $\{A\}$ which has the following meaning: as long as Step 7 is active, grafcet G_2 does not produce the level action *A*. Figure 5.42b gives a grafcet G_{12} equivalent to the grafcet set G_1 and G_2 of Figure 5.42a. If the macroaction *masking* G_2 $\{B^*\}$ had been associated with Step 7, this would mean that action B* could only occur when Step 13 changes from the inactive to the active state if Step 7 is not active at this time.

Impulse macroactions, ***mask*** and ***unmask***, could also be used, which would correspond to a start and end of masking, respectively.

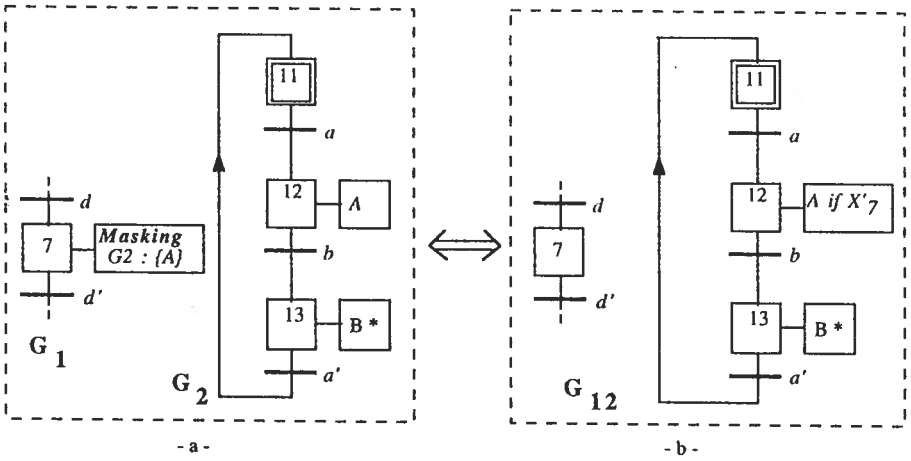

Figure 5.42 Illustration of the macroaction *masking*.

Remark 5.12

a) In Algorithm 5.1, the *impulse macroactions* are performed at the same time as the impulse actions (i.e. even if the activity of the corresponding step is transient) and the *level macroactions* are performed like level actions (i.e. when a stable situation is reached).

b) Among the macroactions defined, the only *impulse macroaction* which is truly indispensable is *force*, which initializes a grafcet in a situation without influencing its future behaviour. The impulse macroactions *freeze*, *release*, *mask* and *unmask* are not indispensable, since the same results can be obtained with the level macroactions *freezing* and *masking*. □

5.7.3 Reply to the specifications of example 6 using macroactions

The set of four grafcets $\{G_1, G_2, G_3, G_4\}$ of Figure 5.43 meets the specifications of Section 5.7.1. Grafcet G_1 corresponds to functioning without faults.

Grafcet G_2 modifies the behaviour of G_1. When a fault appears, Step 22 or 24 becomes active (Step 22 if $x = 1$ or Step 24 if $y = 1$), thereby inactivating the entire grafcet G_1 (since G_1 is forced into a situation where there are no active steps), and giving rise to action S (alarm). When the fault clears, the grafcet is reinitialized either in situation {11} or in situation ({12, 13}). Note that we would have obtained a behaviour also satisfying the specifications by replacing the macroaction *force* G_1 {} by the macroaction *masking* G_1 {A, B, C, D*}.

Grafcet G_3 enables the alarm to be stopped, which is an action of G_2. This grafcet returns to its initial situation as soon as the fault has cleared.

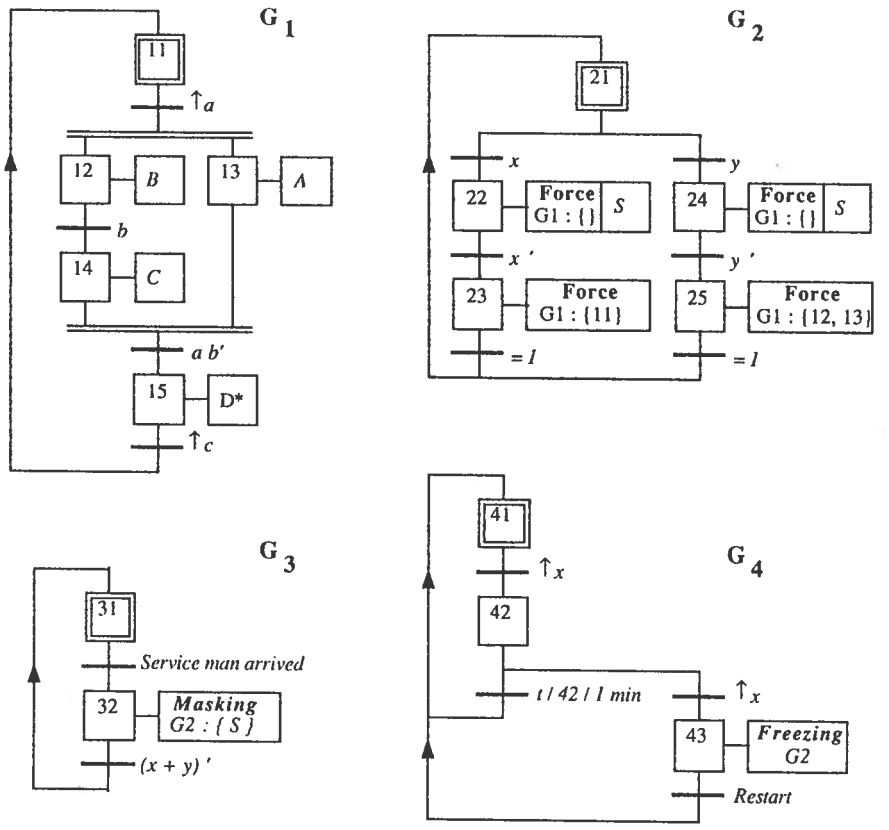

Figure 5.43 Example 6: using the macroactions.

Grafcet G_4 could prevent grafcet G_2 from evolving if the event $\uparrow x$ occurred twice with less than a minute's interval, by the macroaction *freezing*.

A set of four grafcets is thus obtained in which they interact with one another: G_2 acts on G_1, while G_3 and G_4 act on G_2. A hierarchy is thereby established between grafcets.

5.7.4 Hierarchy between grafcets

When a grafcet G_1 emits a macroaction which relates to a grafcet G_2 (e.g. *freezing* G_2), G_1 is said to ***act globally on*** G_2. This may be symbolized as shown in Figure 5.44a: an arrow from G_1 to G_2 indicates that there is one (or more) macroaction(s) acting globally in the direction of this arrow.

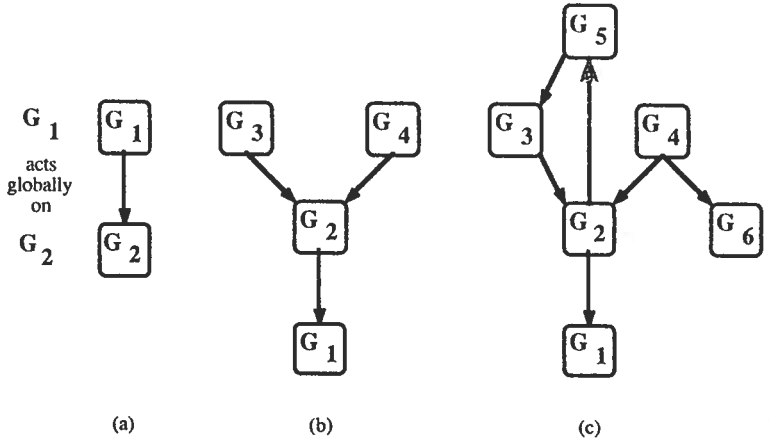

Figure 5.44 Hierarchy between grafcets. (a) G_1 superior in hierarchy to G_2. (b) Graph corresponding to Figure 5.43. (c) General case (the grey arrow should be avoided).

The hierarchies between the grafcets of Figure 5.43 are represented in Figure 5.44b. It can be seen that *several grafcets* (G_3 and G_4) *can act globally on the same grafcet* (G_2), but this must be used with care in order to avoid macroactions with contradictory effects. Figure 5.44c (the part with black lines) indicates a more general case. The same grafcet (G_4) is seen to act globally on several others (G_2 and G_6).

The grey arrow in Figure 5.44c corresponds to a case *which must be avoided*. If there is an arrow from G_2 to G_5, there is the cycle: G_5 acts globally on G_3, G_3 acts globally on G_2 and G_2 acts globally on G_5. This cycle may result in an aberrant behaviour such as an instability.

Recommendation 5.6 The graph indicating the *hierarchical relations* between grafcets (such as Figure 5.44) must **not** include **cycles**. Furthermore, the case in which two grafcets act globally on the same third grafcet must be handled with care. □

[EXERCISES 5.28 AND 5.29]

5.8 DESCRIPTION POWER OF GRAFCET

In this section we shall see that the Grafcet is a **powerful** description tool. The classical models describing sequential systems correspond to *asynchronous* machines on the one hand and to *synchronous* machines on the other. In each case, there are the so-called *Moore* machines, for which the outputs depend on the

internal state only, and the *Mealy* machines, for which the outputs depend on the *total state*, i.e. both on the internal and the input state. We shall see that a grafcet describing the same functioning *can immediately be associated* with each of these models (by this we mean that *a step* is associated with *a state* and *a transition* of the grafcet with *a transition* of the state diagram or flow table). On the other hand, however, grafcets exist which cannot be translated immediately by means of one of these models.

5.8.1 Description of sequential machines

5.8.1.1 Asynchronous machines

Figure 5.45 represents an asynchronous sequential machine with two inputs (a and b) and an output (S). The internal state and/or output can change each time one of the inputs changes.

Figure 5.45a represents a *Moore* machine, having three states known as 1, 2 and 3. The two representations, i.e. by flow table and state diagram, are strictly

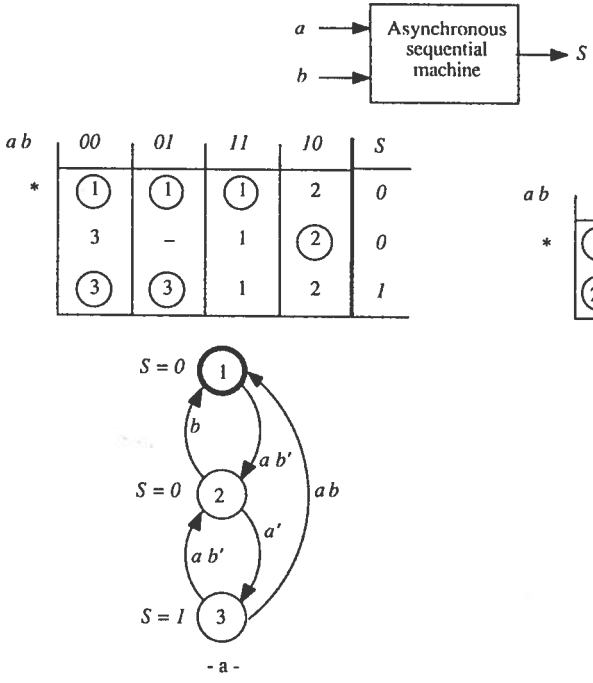

Figure 5.45 Asynchronous sequential machines. (a) *Moore* machine. (b) *Mealy* machine.

equivalent. If the input state is $ab = 11$, the internal state is state 1. On the flow table, the corresponding stable state is represented by a 1 surrounded by a circle in column *11*. If input b changes to *0*, we procede to state 2: on the flow table, we move to column *10* where a 2 appears on the same line which is not surrounded (this represents an unstable state) which indicates the stable state which will be reached. This stable state, 2, corresponds to the second line.

The *internal state* 1 on the state diagram corresponds to the *internal state* (made up of three *total* stable *states*) which have the same number on the flow table. The transition from state 1 to state 2 is represented by an arc which carries the transition condition, i.e. ab'.

The **initial state** is represented by an asterisk on the flow table, and by a thicker line on the state diagram. In this machine, the output $S = 0$ is associated with the internal states 1 and 2, and the output $S = 1$ with the internal state 3. It will be noted that this machine represents a functioning in the *fundamental mode* (Remark 5.7 in Section 5.5.1). Indeed, it is not possible to move directly from column *10* to *01* (which would correspond to the simultaneity of events $\downarrow a$ and $\uparrow b$), hence the unspecified transition in column *01*.

Figure 5.45b represents an equivalent *Mealy* machine. States 2 and 3 have been merged together in a state 23. In this case, the output depends not only on the internal state but also on the input state. On the flow table, this is shown by the representation of the output state after the internal state (for state 23, $S = 1$ for the first two columns and $S = 0$ for the fourth). On the state diagram, this is shown by the fact that the output depends on the inputs ($S = a'$) for state 23).

5.8.1.2 *Synchronous machines*

Figure 5.46 represents a synchronous sequential machine with two inputs, a and b, and an output, S. Input h is a special input, a clock. The internal state and/or output state can only change on a rising edge of clock h. Since this applies to all the transitions, this condition is implicit and is not represented on the flow table and state diagram.

Figure 5.46a represents a *Moore* machine. There are three internal states, 1, 2 and 3. On the flow table, the internal state is represented on the left, and the transitions correspond to the different cases. For example, if we are in state 1 and $ab = 10$, at the next rising edge of h, we shall move to state 2. This is represented by a branch marked ab' on the state diagram. We have the output $S = 0$ when the system is in states 1 or 2 and $S = 1$ when we are in state 3.

Figure 5.46b represents the same sequential machine in the form of a *Mealy* machine. States 1 and 3 are merged together in a state 13. Now the outputs are associated, to the total not to the internal states.

The notion of a stable state no longer has any meaning in a synchronous machine. Each time that edge $\uparrow h$ occurs, both the internal state and the output state may well be changed. In this case, each total state is represented by an arc on

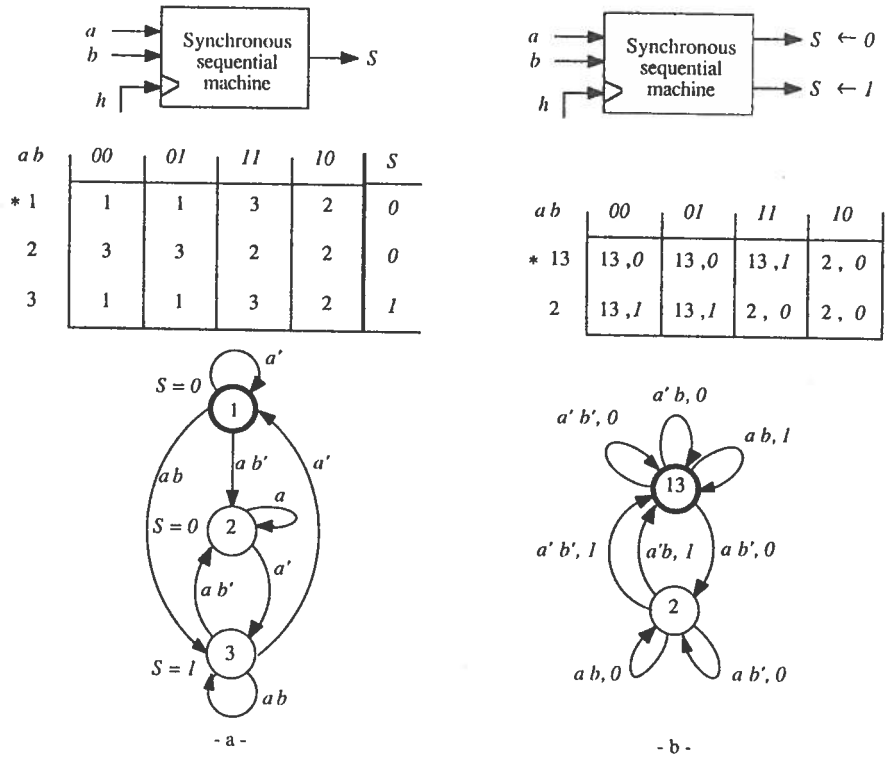

Figure 5.46 Synchronous sequential machines. (a) *Moore* machine. (b) *Mealy* machine.

the state diagram. This arc bears the corresponding input state and the corresponding output state (since we are dealing with a transition, the output state mentioned is the state which will be maintained until the next $\uparrow h$ edge).

Note that a synchronous machine can always be transformed into an asynchronous machine (by considering the clock to be an ordinary input), but that the reverse is not true.

5.8.2 Moving from a flow table or state diagram to a grafcet

5.8.2.1 *Asynchronous machines*

The transformation is immediate and is illustrated in Figure 5.47

Moore machine (Figures 5.45a and 5.47a). A step of the grafcet is associated with each state of the Moore machine. A transition on the grafcet is associated with each

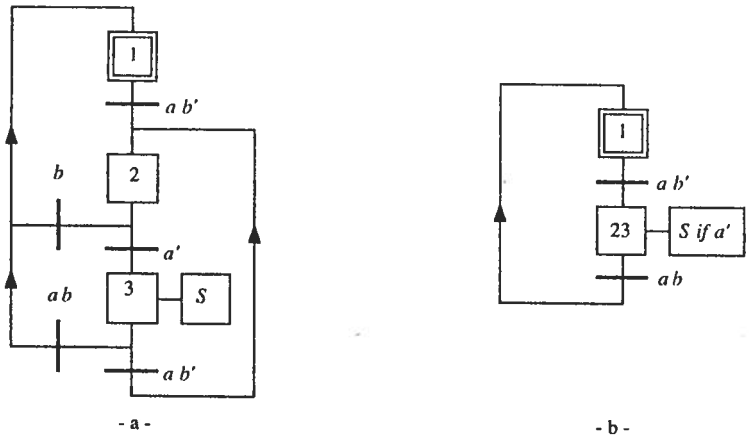

Figure 5.47 Grafcets corresponding to Figure 5.45.

transition of the Moore machine (with a single upstream and a single downstream step). The receptivities correspond to the firing conditions marked on the state diagram. Output S is equal to 1 in state 3. Action S is associated with Step 3 on the grafcet.

Mealy machine (Figures 5.45b and 5.47b). The transformation is carried out in the same way, the only difference being that there are now *conditional actions*. The conditional action S *if* a' of Step 23, corresponds to the output $S = a'$ associated with state 23.

5.8.2.2 *Synchronous machines*

Moore machine (Figures 5.46a and 5.48a). The transformation is carried out immediately as for the asynchronous case. The only difference is that each receptivity *explicitly* mentions that it is on occurrence of *event* $\uparrow h$ that the transition takes place. For example, the transition from state 1 to state 2 is marked ab' on the state diagram of Figure 5.46a. The receptivity $\uparrow h.ab'$ is associated with the corresponding transition of the grafcet of Figure 5.48a.

Mealy machine (Figures 5.46b and 5.48b). In this case, the outputs of the flow table or diagram correspond to *impulse actions* on the grafcet. Certain of these are conditional and *care must thus be taken to explain them thoroughly*.

5.8.3 Moving from the grafcet to the flow table or state diagram?

Although a *grafcet step* can always be made to correspond to *a state*, the *reverse* does *not apply*. This is clear when a grafcet has a number of active steps, but is less

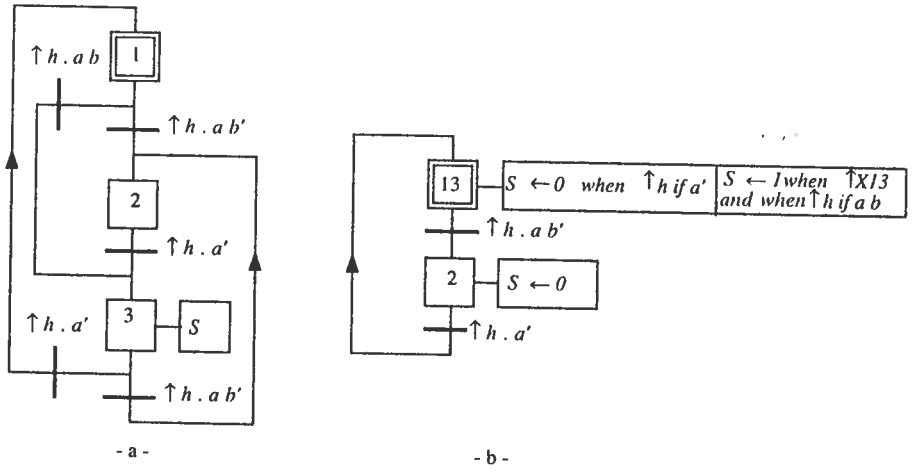

Figure 5.48 Grafcets corresponding to Figure 5.46.

so when a grafcet has only one active step. We shall however provide examples of this. We shall distinguish two cases, namely a grafcet *without concurrency* and a grafcet *with concurrency*. The comparison between grafcet and flow table is rather qualitative in the first case and quantitative in the second.

5.8.3.1 *Grafcet without concurrency*

We shall consider at this point grafcets in which there is *one and only one step active* at any one time (like a sequential machine is in one and only one state at a time), and all transitions have a single directed link which joins up with it and a single directed link leaving it. Given these restricting conditions, it would be easy to imagine that the grafcet could always be transcribed as a flow table or state diagram. However, this is absolutely not the case due to the *interpretation of the grafcet*, and we shall now give three examples of this.

The first example is given in Figure 5.49. The grafcet in Figure 5.49a has three steps and two transitions. If the system is in the initial state and $b = 1$, and if a changes from 0 to 1, situation {3} is reached, passing through the unstable situation {2}. A state diagram requires an additional transition, as shown in Figure 5.49b. The two transitions from 1 to 2 and 1 to 3 also appear on the flow table of Figure 5.49c.

The second and third example relate to the same specifications (see Figure 5.50). The system in question is asynchronous. It has two inputs, a and b, and an output S. Input a is a periodic signal and b a signal which varies in a random manner with the following restriction: over a period where $a = 1$, b does not change value more than

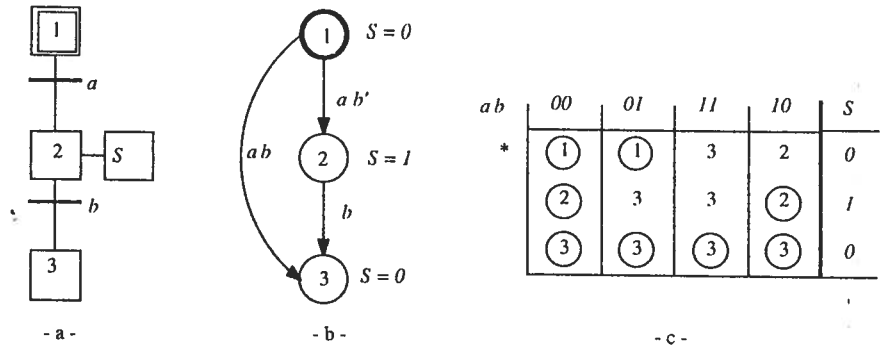

Figure 5.49 The state diagram requires more transitions than the grafcet.

once. Output S equals *1* for all the time of the first impulse a which appears over a period where $b = 1$. An evolution example is shown in diagram form in Figure 5.50a. The *primitive* flow table (i.e. with a single total stable state per line) is represented in Figure 5.50b. The number of lines in this table can be reduced by merging. The minimum number of lines is four. There are two four-line solutions which are represented in Figures 5.50c and d. A grafcet can naturally be associated with each of these three tables (Figures 5.50b, c and d) by immediate transcription, as we have seen in Section 5.8.2. For example, the grafcet of Figure 5.50c corresponds to the table of Figure 5.50d. Step *1* corresponds to the line having the total states, 1, 6 and 8, which is written as 168. Step *2* corresponds to state 2, and so on. This system may also be described by other grafcets. We shall now look at the second and third examples.

The second example is represented in Figure 5.50f which meets the specifications. This grafcet has four steps, but does not correspond to either of the two possible four-internal flow tables. Internal states 168, 2 and 3 can be made to correspond to Steps *1*, *2* and *3*, respectively, but an internal state cannot be made to correspond to Step *4*. Note that this grafcet contains no external events in its receptivities, only Boolean conditions.

The third example is represented in Figure 5.50g. The reader could check that this grafcet also corresponds to the specifications. This grafcet contains receptivities with external events. It has three steps, none of which can be likened to a state in the sense of a flow table/state diagram.

5.8.3.2 *Grafcet with concurrency*

In this case it is quite clear that no flow tables can be obtained immediately. A *state* corresponds to a *situation* and thus to a *set of steps* in the general case. The only fact we wish to point out here is the concession obtained due to the representation of

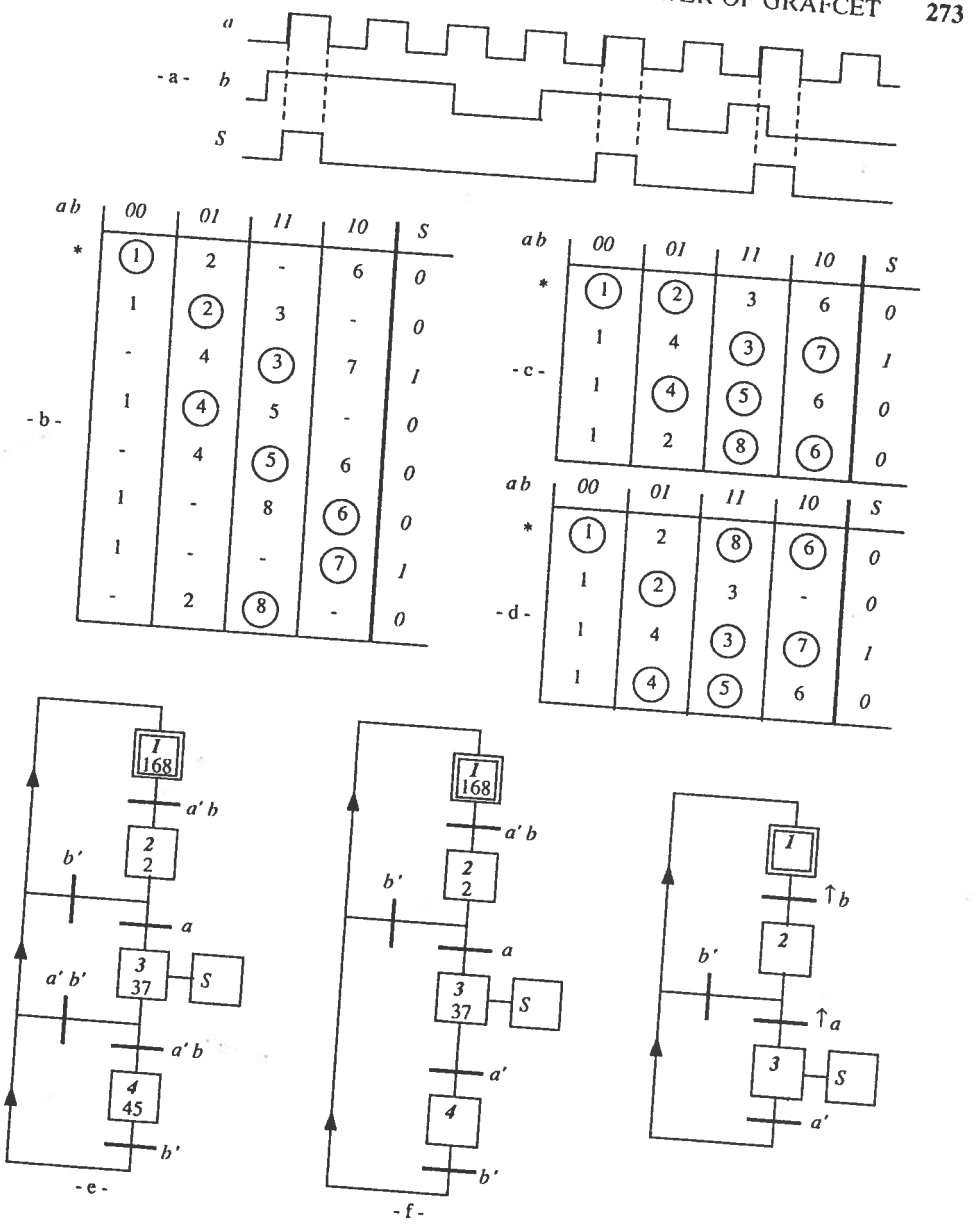

Figure 5.50 Examples of steps not corresponding to states.

the concurrency (which is inherited from the Petri nets).

Let us consider Example 4 (tank filling) presented in Section 5.3.4. Without conditional actions, this system requires at least six steps to be represented (three steps for each tank). Each tank can have three states (being emptied, empty, being filled) independently of the other tank. At least nine states (i.e. 3×3) are thus required to represent this sequential machine. Since this system depends on five variables (two per tank, plus m), a flow table would have thirty-two columns. If we were to consider a similar system with four tanks whose filling is synchronized in the same conditions, we would have:

Grafcet: $3 \times 4 = 12$ steps, representing $3^4 = 81$ states.
Flow table: $2^9 = 512$ columns, thus impracticable.

The possibility of representing the concurrency is a considerable improvement compared with the state models.

The word *concurrency* is evocative and indicates the fact that several steps may be active simultaneously. However, in a general case, it is more prudent to speak of *simultaneous evolutions*. Indeed, the simultaneously active steps are not necessarily in concurrency but may follow on from each other (thus appearing 'in series'). The receptivities associated with the transitions between these steps may be such that an active step will never be reactivated. Exercise 5.30 gives an example of this.

[EXERCISE 5.30]

5.8.4 Conclusion on the description power

The Grafcet has the *same description power* as the models of asynchronous and synchronous machines united. Moreover, the number of *steps and transitions* required for the description by Grafcet is *at the most equal to* the number of states and transitions, respectively, required for the description by state diagrams or flow tables.

5.9 CONCLUDING REMARKS

The Grafcet is thus a powerful tool which is not ambiguous if used with the necessary rigor. Grafcets with *macrosteps* and *macroactions* can be said to form *abbreviations* of ordinary grafcets, to the extent with which an equivalent ordinary grafcet can always be made to correspond to them (this word 'abbreviation' has already been used, with the same meaning, to qualify Petri net categories in Chapter 1). This does not apply if pseudomacrosteps are used.

The Grafcet can be used to describe any input/output behaviour, provided that these inputs and outputs are discrete variables, independently of any implementation. However, its main vocation is the description of *logic controllers* which *must be implemented*. This implementation may be hardwired or, more commonly today, software. The development of the Grafcet software implementation is closely linked to that of the programmable logic controllers, which are special purpose data processing systems.

Some references are provided at the end of the book. We shall only give here a few indications relating to this implementation.

5.9.1 Hardwired implementation

A hardwired implementation is carried out from a grafcet for which all its receptivities are conditions. If the given grafcet has any external events in its receptivities, it must first be transformed. For example, the grafcet in Figure 5.10c would be transformed as shown in Figure 5.10b.

The majority of hardwired implementations have as their base a **request–acknowledge** technique. A memory element of the type *RS latch* is associated with each step. The latch B_i, whose inputs are S_i (setting to *1*) and R_i (setting to *0*) is associated with Step i. Its output is Q_i. Let us consider a transition (1), whose receptivity is condition C_1, between Steps 1 and 2, and let us assume that Step 1 is active. An AND gate carried out the function $S_2 = Q_1 . C_1$. As soon as C_i assumes value *1*, $S_2 = 1$ and Q_2 is thus set to *1* (Step 2 becomes active). This output Q_2 is fed back on input R_1 and so sets output Q_1 to *0* (Step 1 becomes inactive). This implementation is easily generalized in the case of a transition having several input and several output steps. However, it is not possible to construct all grafcet structures using this technique. Problems are seen to arise if there is another transition causing change-over from Step 2 to Step 1. This technique could be improved by setting Q_1 to *0* by $Q_2 . C_1$ instead of Q_2 in order to carry out a more 'selective' resetting to *0*, but examples of grafcets can still be found which cannot be constructed in this fashion.

Another implementation technique is based on the **universal cell** CUSA which may be likened to a RS latch from a logic standpoint, but which possesses an *internal delay* carefully calibrated to avoid hazards. A CUSA i has several inputs S_i and several inputs R_i, as well as an output Q_i. Let us consider the above example again. When function $Q_1 . C_1$ assumes value *1*, this value simultaneously sets CUSA 2 (input S_2) to state *1* and CUSA 1 (input R_1) to state *0*. This is *not* therefore a *request–acknowledge* since it is not the output of CUSA 2 which sets CUSA 1 to *0*. The wiring functions in a suitable manner, without hazards, since the internal delay of the CUSA has been designed with this in mind. This technique enables grafcets to be constructed with *any graphic structure*. However, great care should be paid to the risk of hazards if receptivities depend on the internal state.

5.9.2 Software implementation

In the 1970s, the software implementation of combinational and sequential functions underwent considerable development. These functions were programmed either by using assembler-type languages, or by designing microprograms specific to the application, to gain rapidity of execution. This type of implementation presented two disadvantages which have now been reduced. The first is that the size of the program increases with the size of the grafcet, an increase which may well be considerable. The second is the risk of hazards (which may result in a wrong interpretation) because a grafcet describes concurrency, whereas a program has a sequential operation. Improvements have been made to solve these problems and facilitate the user's task.

In order to avoid increase in program size, the grafcet *data* (graphic structure, receptivities, actions) can be separated from their *interpretation*. A *Grafcet editor* is thus conceived which can either be textual or graphic. In a *textual* editor, the data are introduced in the form of lists. For example, for each transition all the steps upstream and downstream are given, etc. (A *pivot step* may possibly be defined for each transition T_j. This is a step upstream of T_j, and we shall only check whether the firing conditions of T_j are verified if this step is active, hence saving time.) In a *graphic* editor, the data are introduced in the form of drawings. The user draws the grafcet directly on the screen. The majority of programmable logic controller manufacturers have opted for the latter solution. Note that the use of a grafcet editor enables certain description errors to be avoided, since the program imposes/verifies consistency with the Grafcet construction rules (alternance of steps and transitions).

In order to avoid programming hazards, a *Grafcet interpretor* based on the *interpretation algorithm* (Algorithm 5.1) has merely to be implemented. If care is taken *to ensure that the behaviour will be consistent in the event of apparent simultaneity of events* (by introducing if necessary a redundancy into the receptivities, as was shown in Section 5.5.4.3), we can then be sure to have obtained an implementation which thoroughly respects the specifications. This interpretation is transparent to a user, who merely needs to have a 'good' description.

We recommended that grafcets be described in which each step has a single meaning, to help understanding. Implementation will also be simpler if each step has a single meaning, if conditional actions are avoided and if the receptivities are conditions without variable rising or falling edges.

NOTES

1 Nets with Boolean markings have also been defined by C. A. Petri: *condition/event nets* (or *C/E-nets*).

2 The concept of a *sound grafcet* was introduced by H. Deneux and R. David under the name of *restricted grafcet*. We have changed the name which could lead to thinking that there is a restriction in the definition of the model, whereas the restriction is only in its use.