Cours
Systèmes à événements discrets

Chapitre : GRAFCET=GRAphe Fonctionnel de Commande Etape Transition

MAZA Samia, Maître de conférences ENSEM –Centre de Recherche en Automatique de Nancy
2021

Historique

Un des groupes de travail de l'AFCET « Associations Française pour la Cybernétique Economique et Technique » : le **groupe *Systèmes Logiques***, décida en 1975 de créer une commission ***Normalisation de la représentation du cahier des charges d'un automatisme logique***.

Le besoin de décrire des automatismes logiques de plus en plus complexes, dont la mise en œuvre allait être facilitée par l'avènement des microprocesseurs, se faisait sentir.

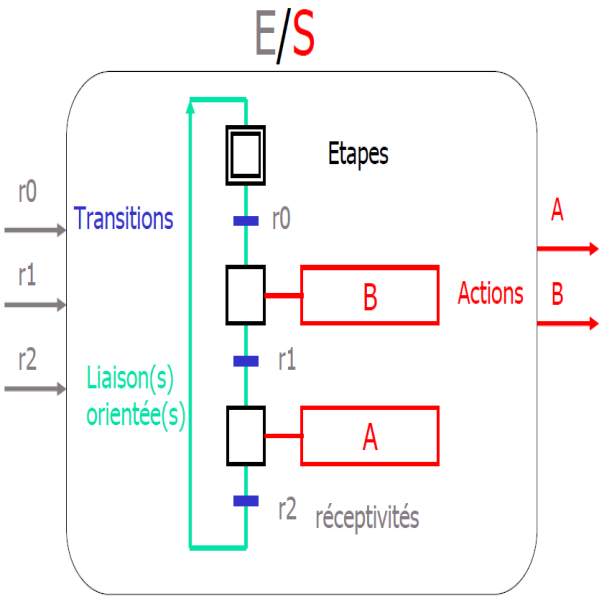
Après un état des lieux des outils utilisés pour décrire les automatismes logiques, 3 grandes classes se sont dégagées : (1) les organigrammes, (2) la représentation de systèmes logiques à évolutions simultanées (dont les RdPs) et (3) les graphes d'état.

⇨ Un outil de représentation fut retenu (1977): c'est un outil de spécification qui décrit uniquement la fonction à réaliser, indépendamment de toute technologie et de toute réalisation.
Il est inspiré des RdPs, avec quelques petites différences, avec une interprétation unique des entrées-sorties (ce qui n'était pas le cas des RdPs) [Alla & David, 92].

⇨ On l'appela : le **Grafcet**. Aujourd'hui, il est largement enseigné et utilisé en industrie (après avoir été une norme française en 1982, il est devenu une norme internationale en 1987).

1. INTRODUCTION.

1.1 Les automates programmables industriels

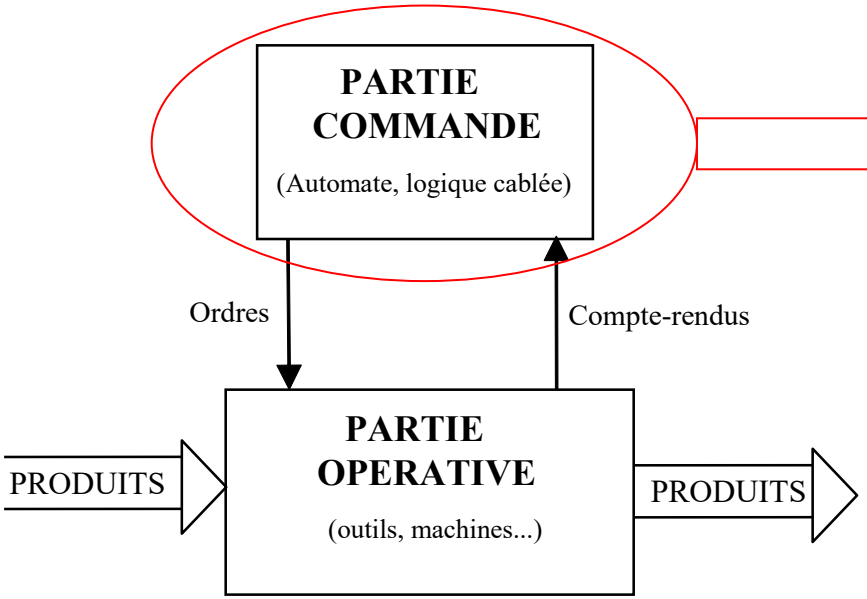


Le **GRAFCET** est un outil graphique destiné à spécifier les **comportements attendus** de **systèmes séquentiels** ou de **parties séquentielles de systèmes**, en spécifiant les actions à effectuer et leur enchaînement conditionné par des événements ou par des conditions (prédicats).

Des **outils industriels d'implantation**, les **API** (Automates Programmables Industriels) ou les **SNCC** (Systèmes Numériques de Contrôle Commande), permettent la **réalisation programmée** de comportements spécifiés en Grafcet. L'écriture des programmes se fait alors dans un **langage graphique proche et issu du Grafcet**.

⇨ Le langage SFC (norme CEI 61131-3)* : Langage de programmation graphique des automates programmables industriels (APIs).

*Voir Annexe



Exemple d'un API de la gamme TSX micro (de Schneider Electrics)

Un automate programmable industriel ou API (on dit PLC pour le nom en anglais : *Programmable Logic Controller*) est une machine électronique programmable par un personnel non informaticien et destinée à réaliser, en milieu industriel, la fonction d'une partie commande.

Le terme programmable signifie que le cahier des charges traduisant le comportement de cette partie commande est indiqué à l'API par un programme pouvant facilement être modifié.

Il est constitué de plusieurs modules fonctionnels :

- Le module processeur,
- Les modules Entrées (*Inputs*),
- Les modules Sorties (*Outputs*),
- Le module mémoire (*Memory*),
- Le module de communication (liaison série, Ethernet,…),
- Le module d'alimentation électrique.

Exemple introductif.

Un premier Grafcet simple décrivant l'allumage et l'extinction d'une lampe L au moyen d'un bouton poussoir m.

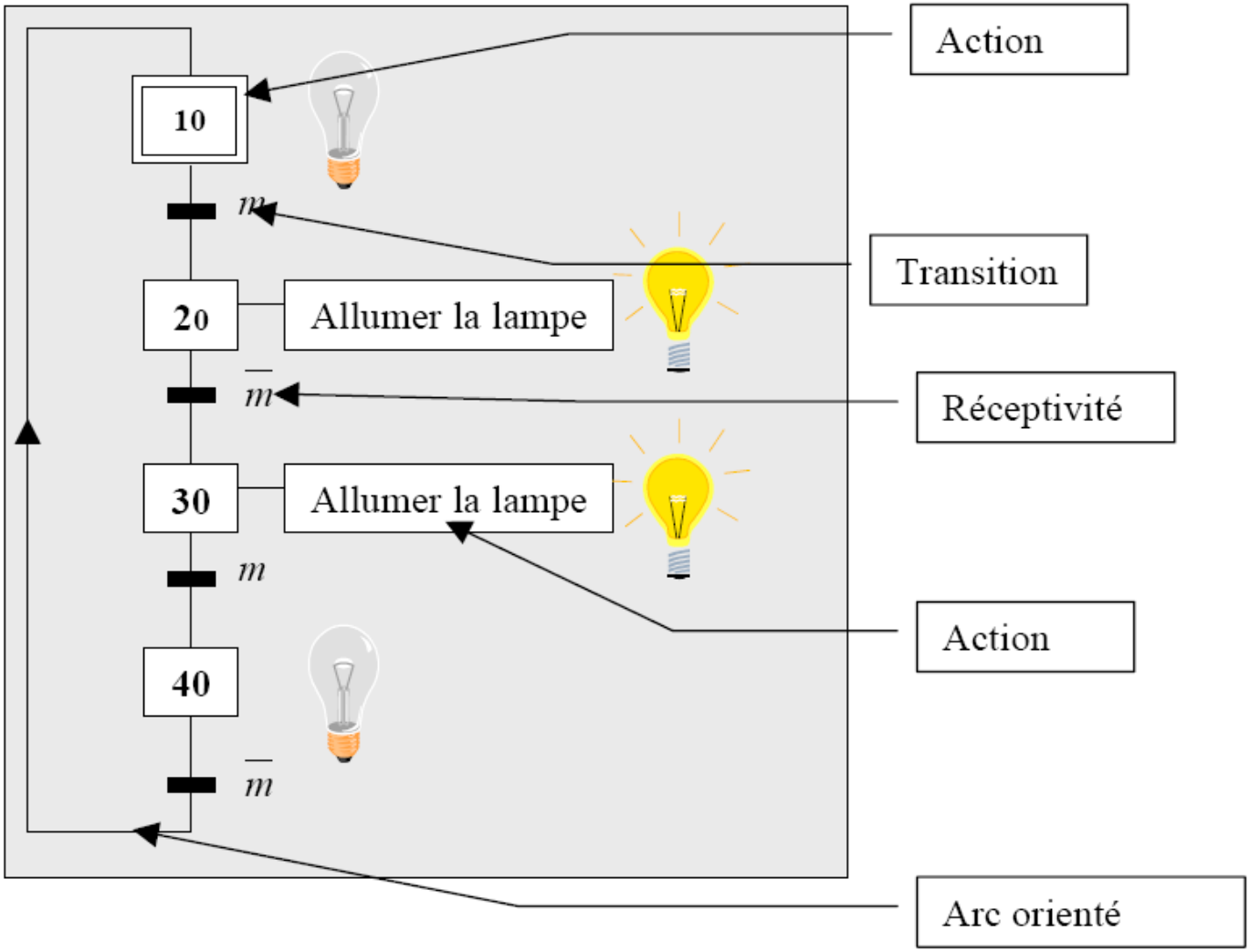
Si la lampe est éteinte et que l'on appuie sur m, la lampe L s'allume.

Si on relâche m elle reste allumée.

Si on appuie de nouveau sur m elle s'éteint.

Elle reste éteinte si on relâche m.

Ceci se traduit par le grafcet ci-contre :

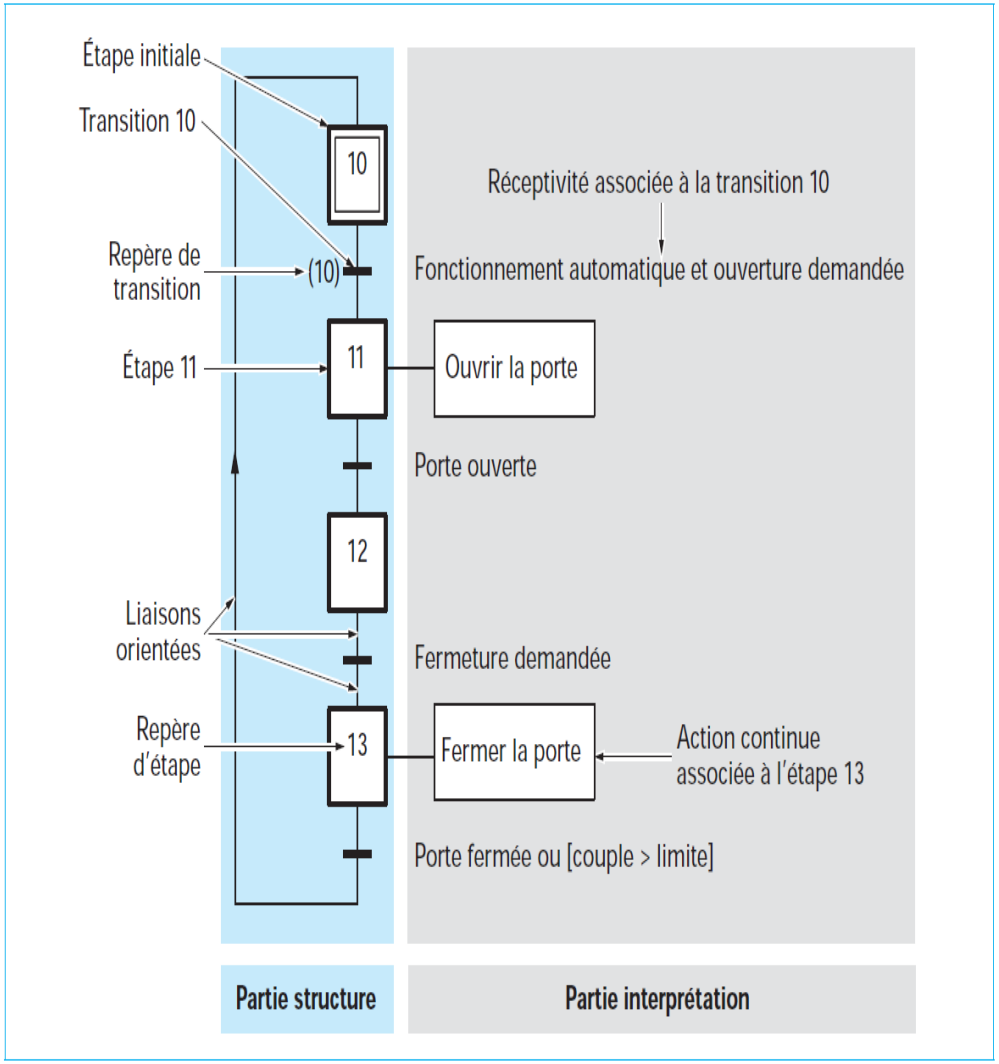


Le langage GRAFCET est défini par un ensemble constitué :

— *d'éléments graphiques de base :* les étapes, les transitions et les liaisons orientées, formant la partie graphique du grafcet et sa **structure** qui décrit l'évolution possible entre les situations du système ;

— *d'une* ***interprétation,*** traduisant les comportements de la partie commande vis-à-vis de ses entrées/sorties et caractérisée par les actions associées aux étapes et les réceptivités associées aux transitions ;

— *de cinq* ***règles d'évolution***, définissant formellement le comportement dynamique.

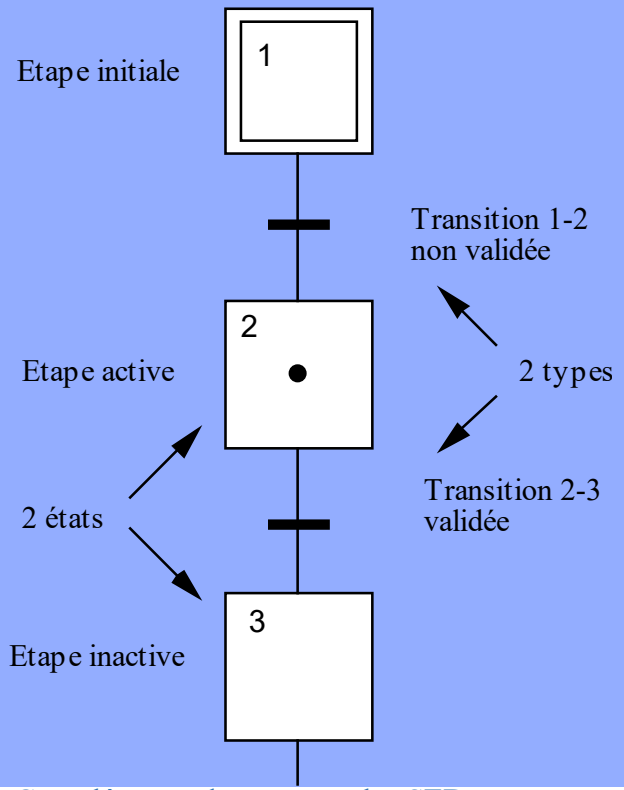


2. DESCRIPTION FORMELLE.

2.1 Les éléments du SFC ou Grafcet

Formellement, le SFC ou Grafcet est un graphe orienté biparti: G = (E, T, A, M_0) avec E un ensemble fini d'**étapes**, A un ensemble d'**actions** associées aux étapes, **T** un ensemble fini de **transitions** et M_0 un **marquage initial**.

Figure 2.4. Un exemple de Grafcet



Définition

Une étape est une situation dans laquelle le comportement de la partie opérative est **invariant** vis à vis des entrées et des sorties. Elle est symbolisée par carré (double pour l'étape initiale), avec un numéro à l'intérieur. Elle peut avoir deux états, active ou non active selon qu'elle contient une marque ou non (figure 2.4). L'**état interne** du graphe à un instant donné est l'ensemble des étapes actives.

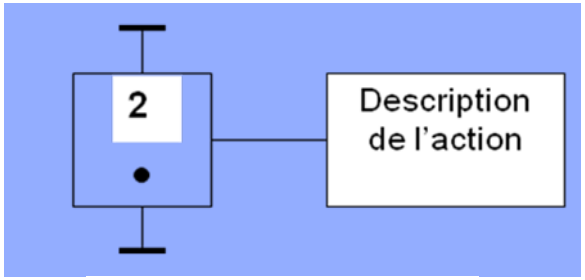
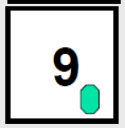


Figure 2.5

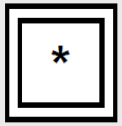
Etapes

X9 variable d'étape de l'étape 9

X9 =0 (False) 

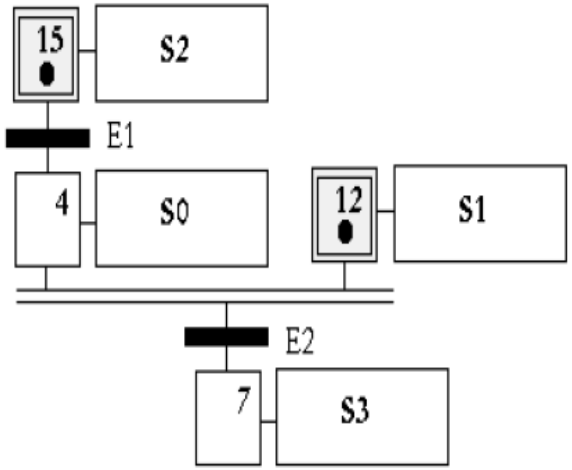
X9 =1 (true) 

Etat actif de l'étape



Symbole de l'étape initiale

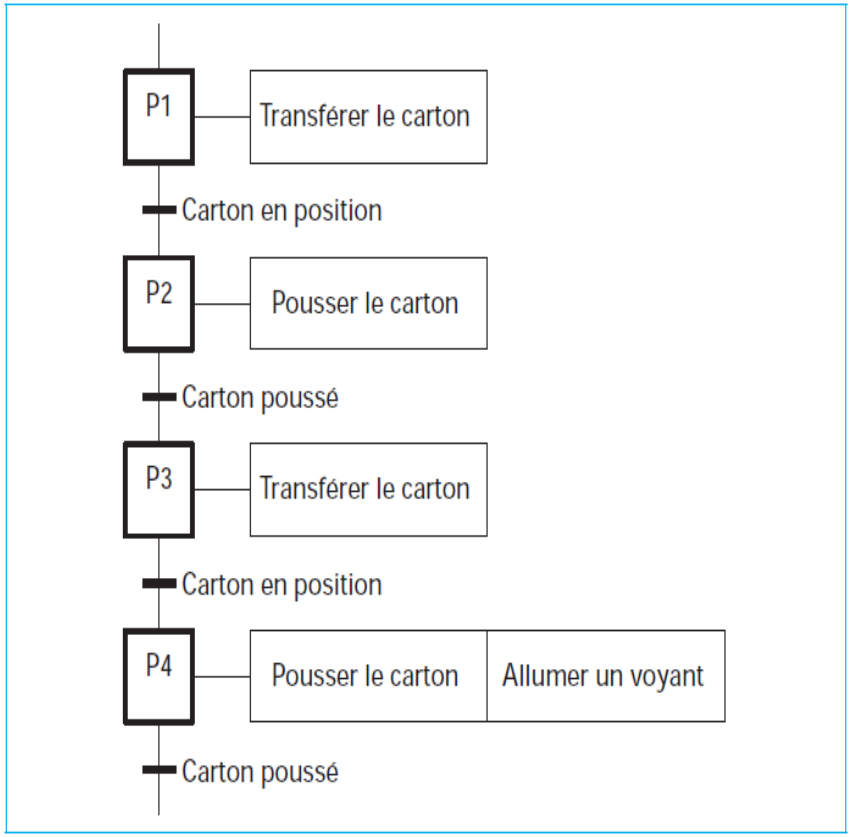
* est un repère alphanumérique



La situation initiale d'un Grafcet caractérise le comportement initial de la partie commande (vis à vis de la PO, de l'opérateur…).
Elle correspond aux étapes actives au début du fonctionnement. Elle traduit généralement un état de repos.

Propriétés

Les **actions** associées à une étape décrivent **l'effet à obtenir du système contrôlé** (on parle souvent de partie opérative par opposition à la partie commande que l'on décrit) ou un ordre à produire pour obtenir cet effet.



➔Elles sont symbolisées par un rectangle relié à l'étape contenant la description des actions.

➔Dans une même étape plusieurs actions sont possibles.

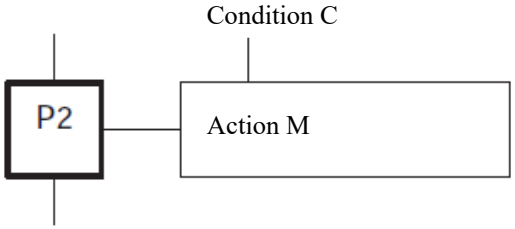
➔Deux types d'actions : **Continue** (ou permanente) et **Impulsionnelle** (ou mémorisée).

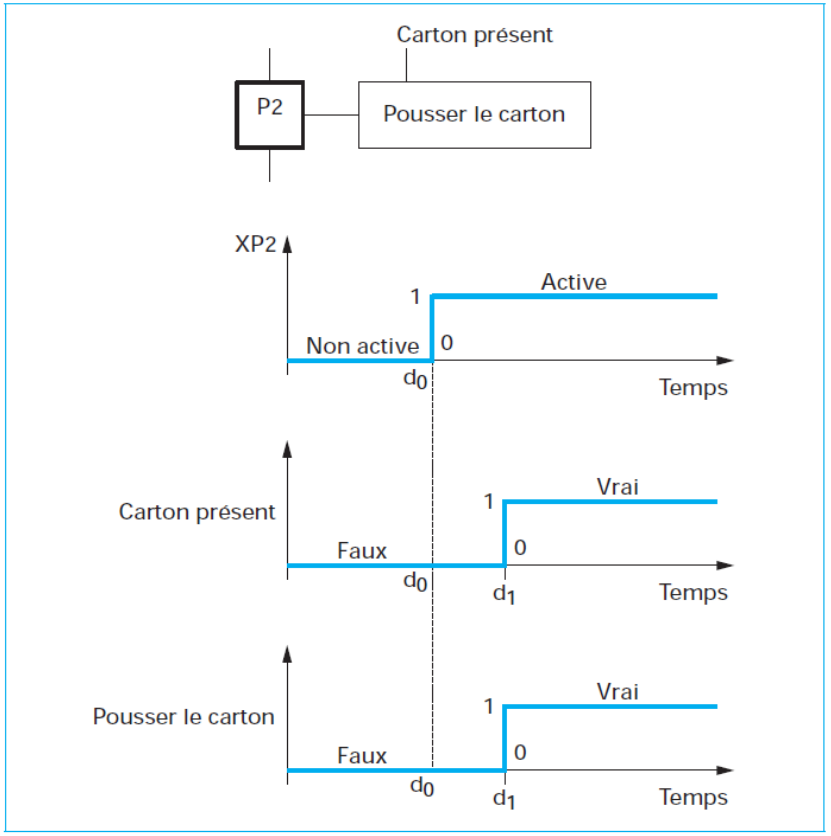
(a) Action Continue

Si l'action est **continue** elle dure tant que l'étape est active.

Une action continue peut être **conditionnelle** : dans ce cas l'action sera réalisée, si l'étape est active et tant que la condition associée sera vraie.

Ici, quand l'étape P2 est active, l'action M est vraie tant que la condition C est vérifiée.

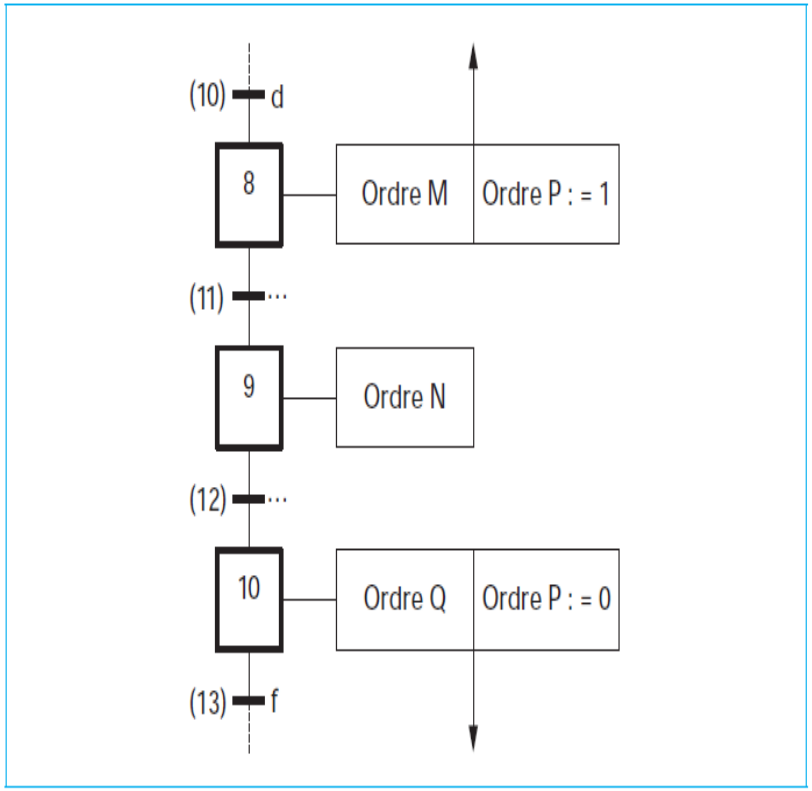




(b) Action Impulsionnelle

L'action peut être **impulsionnelle :** C'est l'association d'une action à des événements internes (activation d'une étape, désactivation d'une étape...) qui permet d'indiquer qu'une variable de sortie **prend et garde** la valeur imposée si l'un de ces événements se produit. La valeur d'une variable de sortie relative à une action mémorisée reste inchangée tant qu'un nouvel **événement** spécifié **ne la modifie pas**.

On peut la noter avec un astérisque : (action)*.



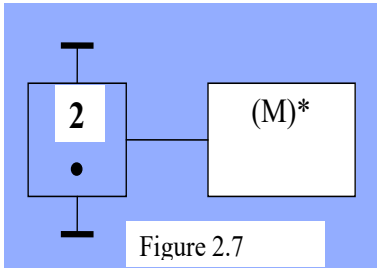


Figure 2.7

Définition

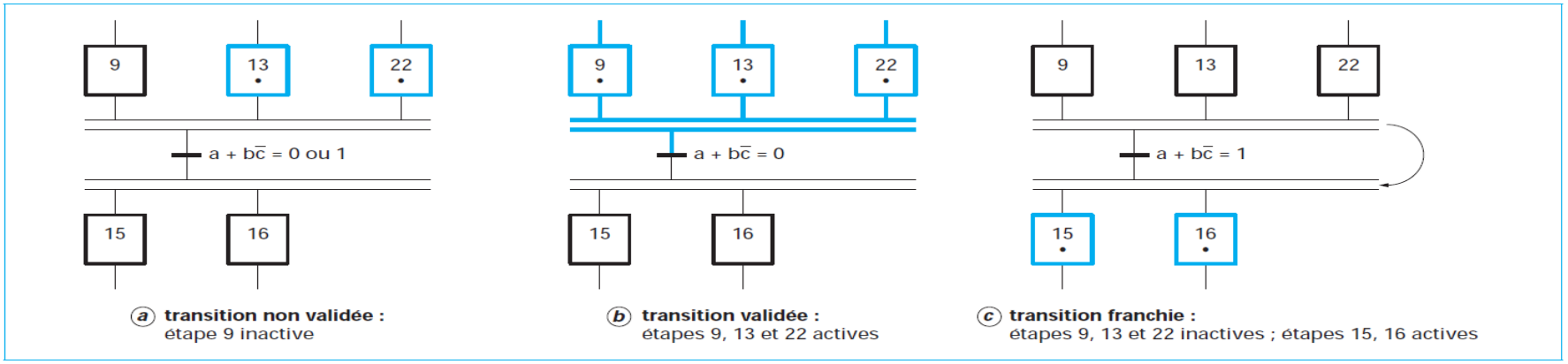
Une **transition** indique la possibilité d'évolution d'activité entre deux ou plusieurs étapes. Cette évolution s'accomplit par le franchissement de la transition d'une étape à une autre. Le symbole est une barre horizontale avec un identificateur à côté.

Une **réceptivité** est un élément du langage GRAFCET. Associée à une transition, la réceptivité exprime le résultat d'une expression booléenne. Une réceptivité est soit vraie, soit fausse.

⇨ La réceptivité d'une transition peut être une fonction logique des variables externes (entrées) et des variables internes (l'état) ou un événement externe (changement d'état, égalité, seuil, date) qui se note par ex. ↑B ou une combinaison des deux. Elle peut aussi **dépendre du temps**.

Propriétés

Si la réceptivité **est vraie** ➔ la transition est **franchissable** sssi elle **est validée par le marquage** (présence de marques dans les places amont = étapes amont actives).



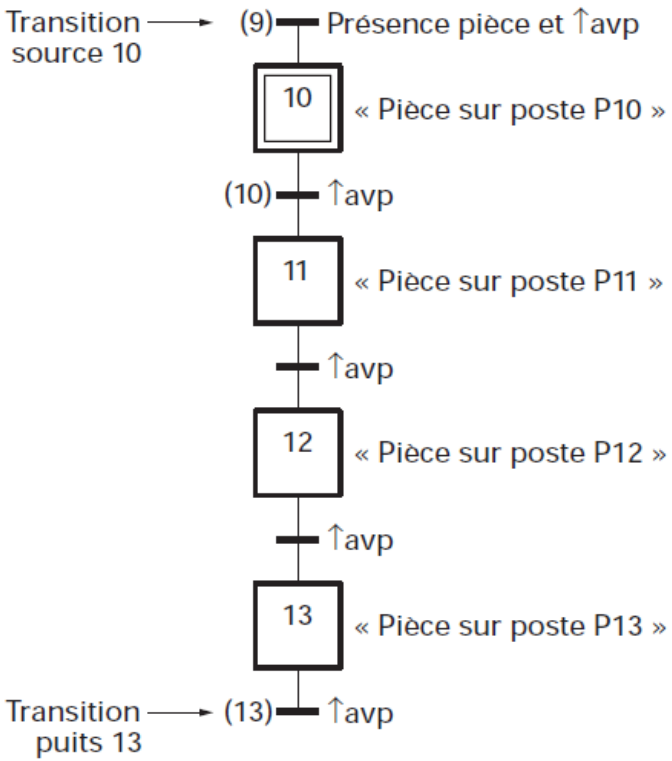
(a) transition non validée : étape 9 inactive

(b) transition validée : étapes 9, 13 et 22 actives

(c) transition franchie : étapes 9, 13 et 22 inactives ; étapes 15, 16 actives

Quelques cas de transitions particulières :

- La *transition toujours réceptive* : sa réceptivité est toujours vraie. Elle est notée **1**.
- La *transition puits* est une transition qui ne possède aucune étape aval. Lorsque la transition puits est validée et que sa réceptivité associée est vraie, le franchissement de cette transition a pour unique conséquence de désactiver les étapes amont.
- La *transition source* est une transition qui ne possède aucune étape amont. Par convention, la transition source est toujours validée et est franchie dès que sa réceptivité est vraie.



Définition

Les **arcs orientés** relient les étapes aux transitions et réciproquement. Il y a obligatoirement alternance étape / transition, transition / étape (deux étapes ou deux transitions ne peuvent être reliées directement). Les arcs sont représentés par des traits verticaux (éventuellement horizontaux dans le cas particulier des divergences). Le sens normal des arcs est du haut vers le bas. On indiquera seulement par une flèche les arcs remontant du bas vers le haut.

Définition

Le marquage initial M_0 représente l'ensemble des étapes actives à l'instant t = 0. Avant l'instant initial, le système de commande peut être hors énergie ou alimenté mais inactif (pas d'étapes actives). Initialiser c'est forcer les étapes initiales à l'état actif. En cas de coupure d'énergie intempestive, au retour de l'énergie, on peut reprendre à l'état initial ou le dernier état avant celui interrompu ➔ Dans ce cas, il y a un certain nombre de risques d'incohérence avec l'état du système sous contrôle auxquels il faut veiller particulièrement.

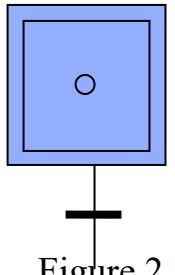


Figure 2.8

2.2 Les règles d'évolution.

Règle n°1 (*Initialisation*): *Il y a au moins une étape initiale; la situation initiale du système modélisé doit être précisée par l'activation de l'ensemble des étapes initiales.* Une transition est **validée** quand toutes les étapes qui la précédent (étapes d'entrée de la transition) sont actives.

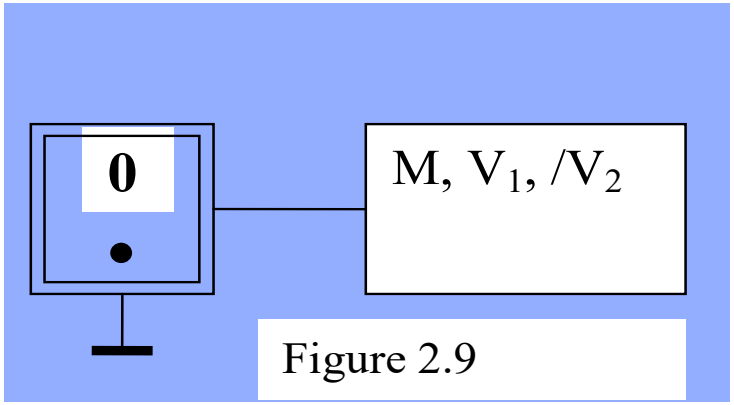


Figure 2.9

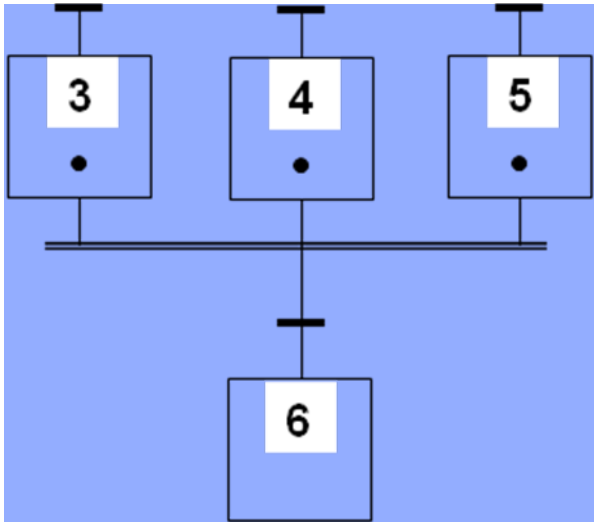


Figure 2.10

Règle n°2 : *Une transition peut être **franchie** si elle est validée et si sa réceptivité est vraie.*

Toute transition franchissable est obligatoirement franchie.

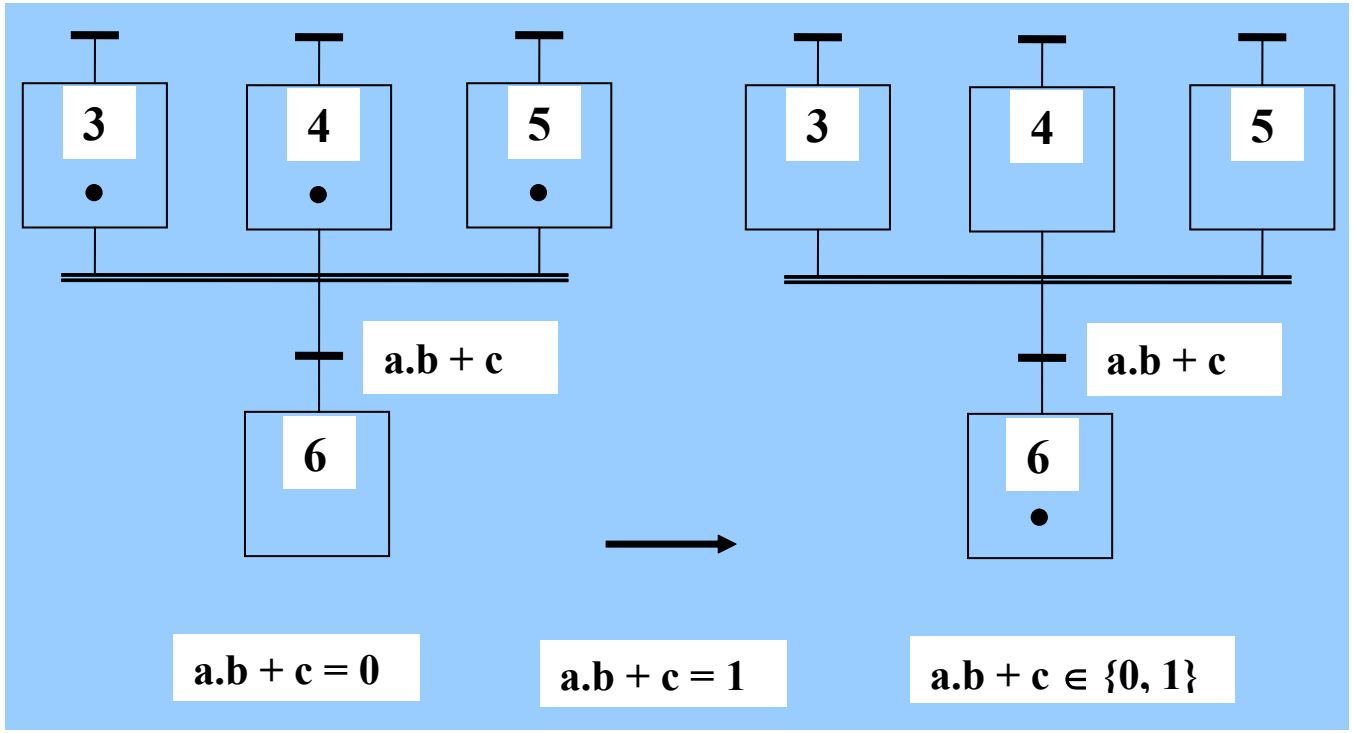


Figure 2.11

Règle n°3: *le franchissement d'une transition entraîne simultanément la désactivation de toutes les étapes d'entrée de la transition et l'activation de toutes ses étapes de sortie.*

La durée de franchissement est considérée comme infiniment petite

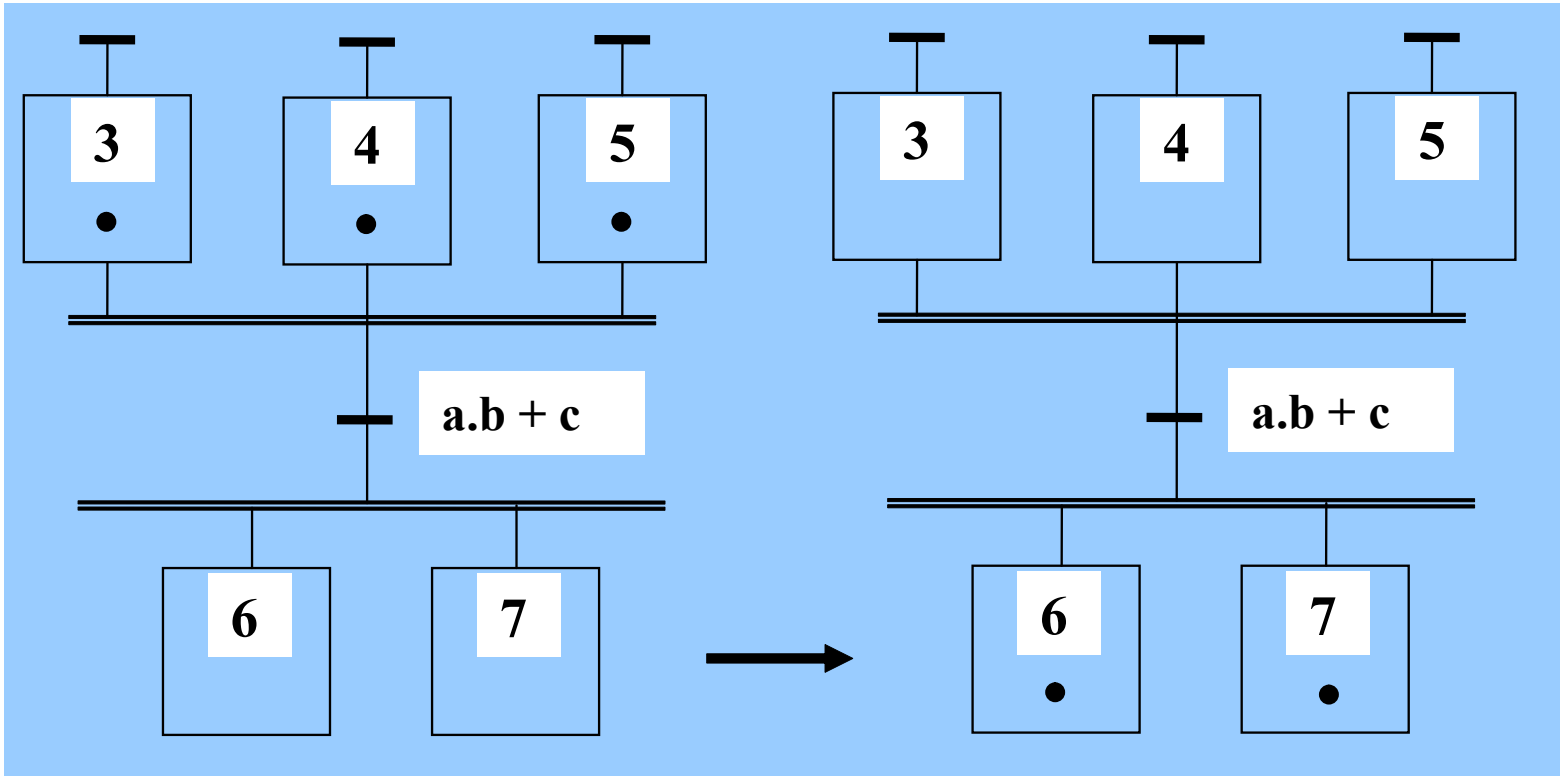


Figure 2.12

Règle n°4 : *plusieurs transitions simultanément franchissables sont simultanément franchies*.

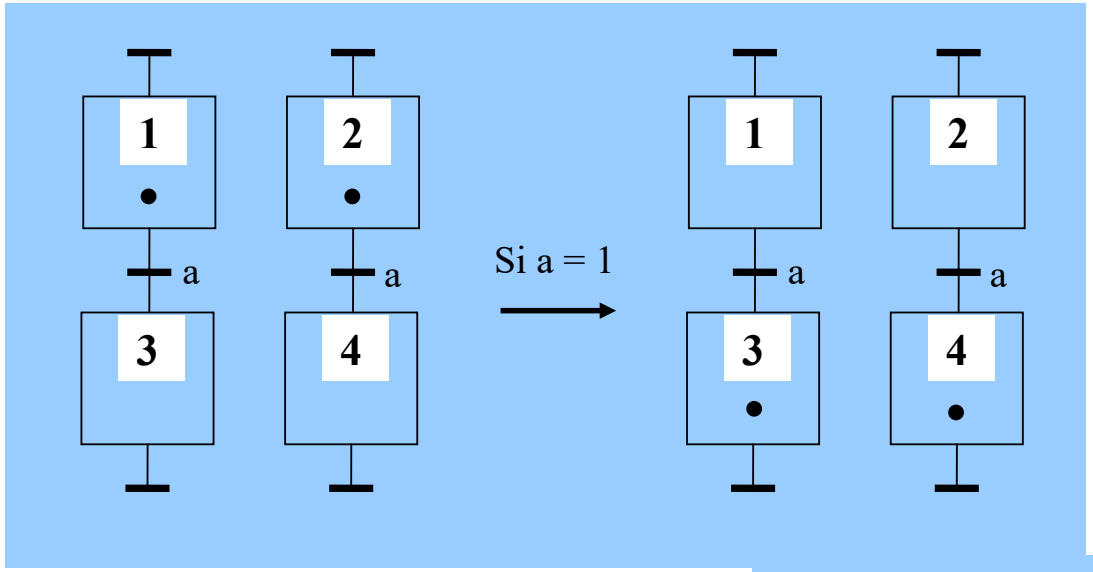
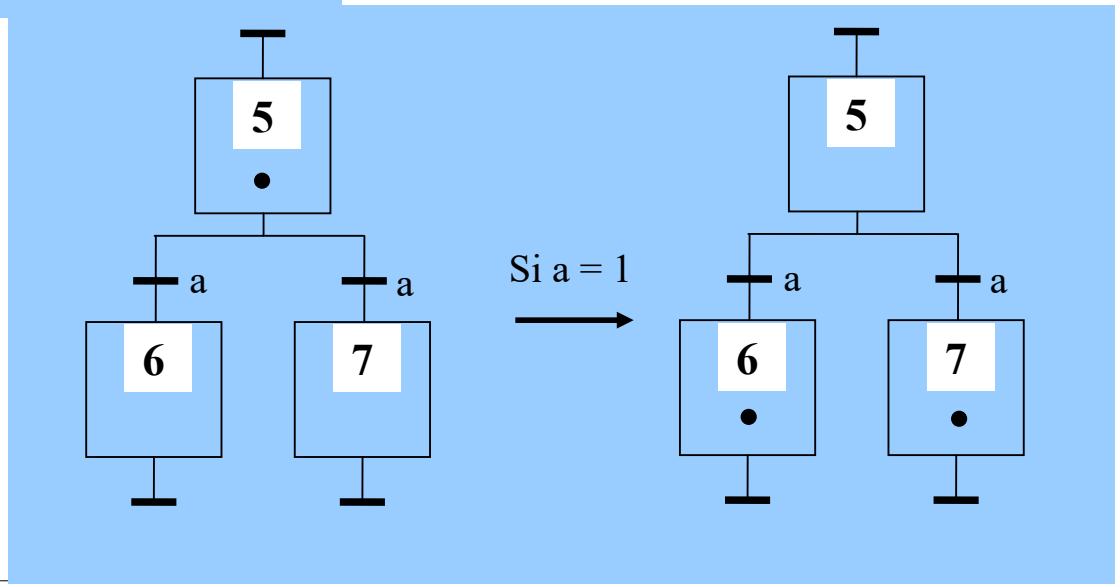


Figure 2.13



Règle n°5 : *Si au cours du franchissement d'une ou plusieurs transitions, une même étape doit être simultanément désactivée et activée, elle reste active.*

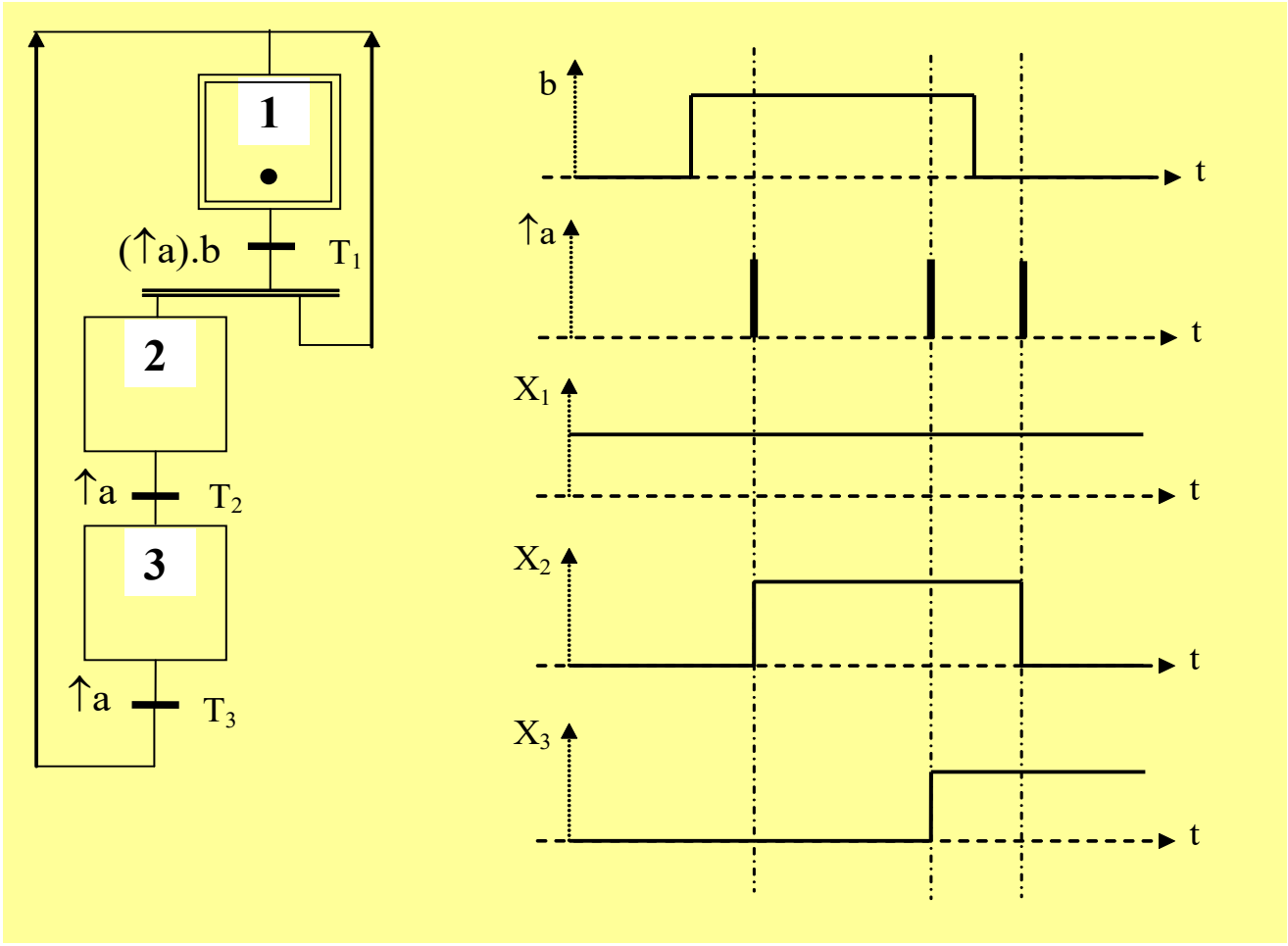


Figure 2.15

Les variables booléennes X_i représentent les états des étapes. Si l'étape i est active, la variable X_i est vraie (niveau haut), sinon, elle est fausse (niveau bas).

2.3 Les structures de base :

Les structures de base sont des configurations de grafcets associées aux concepts de base des systèmes logiques.

Elles permettent d'exprimer notamment des successions d'états, des sélections entre plusieurs séquences, des parallélismes de séquences, des sauts et des reprises de séquences, des partages de ressources, des couplages entre séquences.

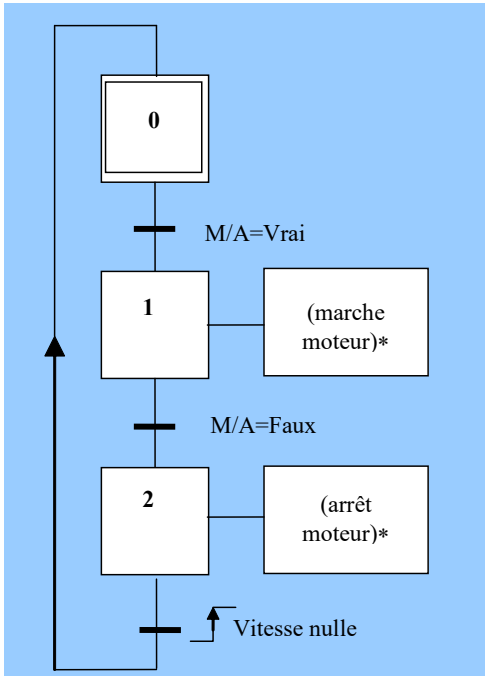
2.3.1 Séquence unique :

La figure ci-contre représente une **séquence unique** dans laquelle les étapes 0, 1 et 2 vont être enchaînées indéfiniment.

La séquence est dite « active » si au moins une de ses étapes est active.

Elle est dite « inactive » lorsqu'aucune de ses étapes n'est active.

Le nombre d'étapes formant une séquence est aussi grand que l'on veut.



2.3.2 Séquences simultanées :

Les séquences simultanées sont définies par les notions de **divergence** et **convergence**.

- ***La divergence en ET*** (distribution/parallélisme) est telle que le franchissement de la transition *T1* active simultanément toutes les places *E1, E2, ... En*. La désactivation des places dépend des réceptivités aval individuelles.

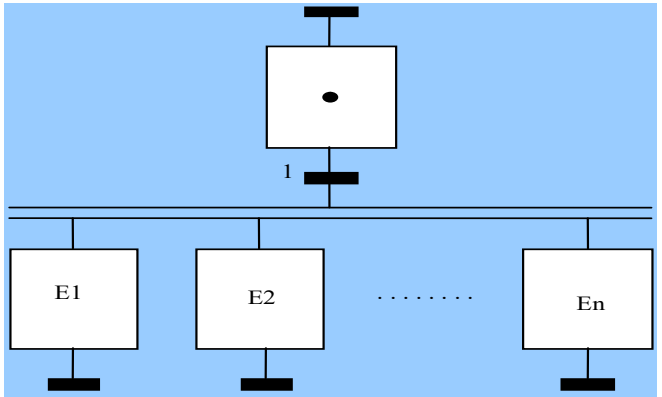
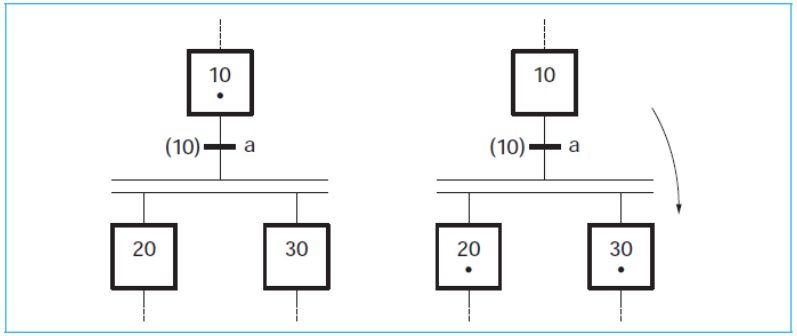


Figure 2.17. Divergence en ET



Exemple de l'activation simultanée de **séquences parallèles**

- ***La convergence en ET*** (jonction/synchronisation) est telle que si toutes les étapes *Di* sont actives et la réceptivité de la transition *T2* vraie, alors le franchissement désactive simultanément toutes les places *Di* et active l'étape *A*.

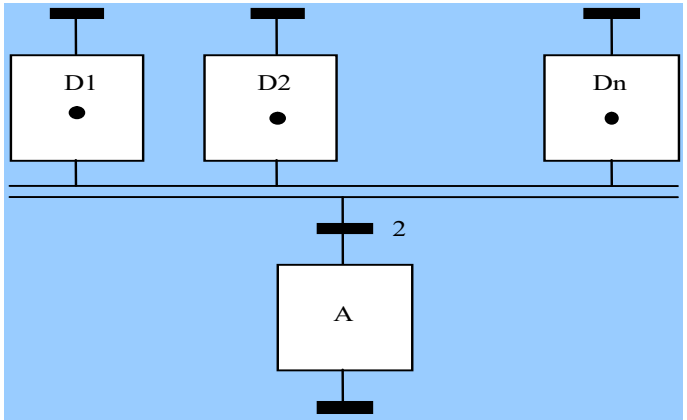
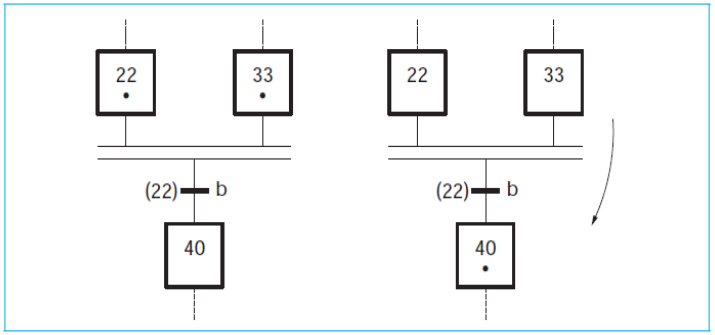
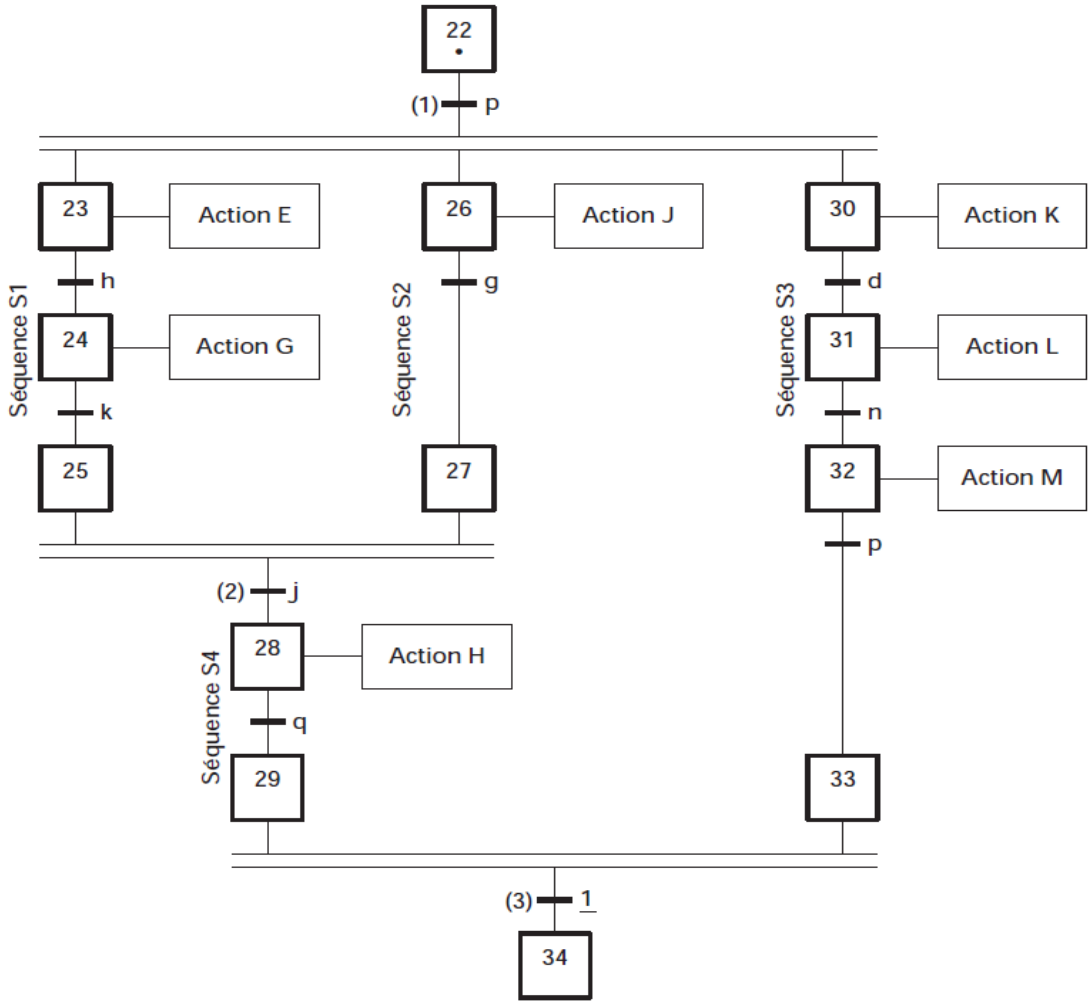


Figure 2.18. Convergence en ET



Exemple de synchronisation de **séquences parallèles**



A partir de l'étape 22 active, le franchissement de la transition (1) par la réciptivité p = 1 provoque simultanément l'activation des trois séquences S1 = 23 – 24 – 25, S2 = 26 – 27, S3 = 30 – 31 – 32 – 33 et la désactivation de l'étape 22.

Ces trois séquences évoluent alors de façon totalement indépendante et lorsque les étapes 25 et 27 d'attente réciproque seront actives et la réceptivité *j* = 1, le franchissement de la transition (2) conduira à désactiver les deux séquences S1 et S2 et à activer une nouvelle séquence S4 = 28 – 29.

De la même façon quand, à leur tour, les étapes 29 et 33 seront actives en même temps, la transition (3) sera immédiatement franchie (réceptivité (3) toujours vraie), provoquant l'activation de l'étape 34 et la désactivation des séquences S4 et S3.

Figure 22 – Séquences simultanées et étapes d'attente réciproque

La divergence en OU (sélection, aiguillage) est telle que (figure 50) une ou plusieurs des transitions d'arrivée peut être franchie. Il y a alors désactivation de *D* et activation de l' (ou des) étape (s) correspondante (s). Les réceptivités doivent être vraies avant que *D* soit active. Le OU peut être inclusif ou exclusif.

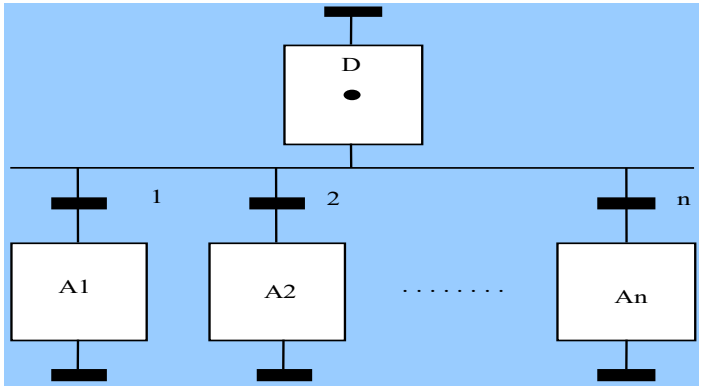


Figure 2.19. Divergence en OU

La convergence en OU (attribution, aiguillage) est telle que (figure 51) A peut être activée par n'importe quelle des transitions amont (OU exclusif en général car la simultanéité est peu probable).

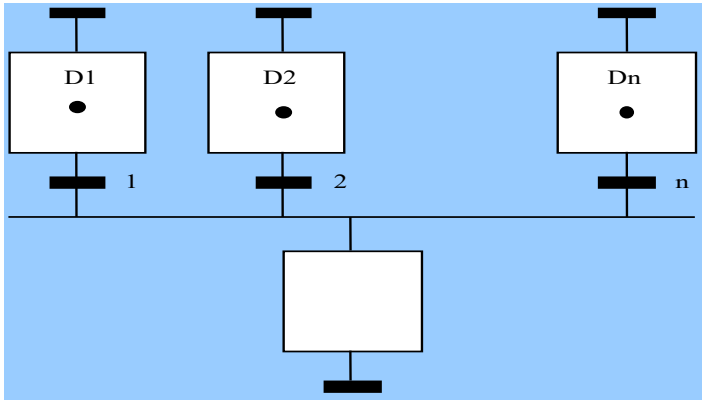
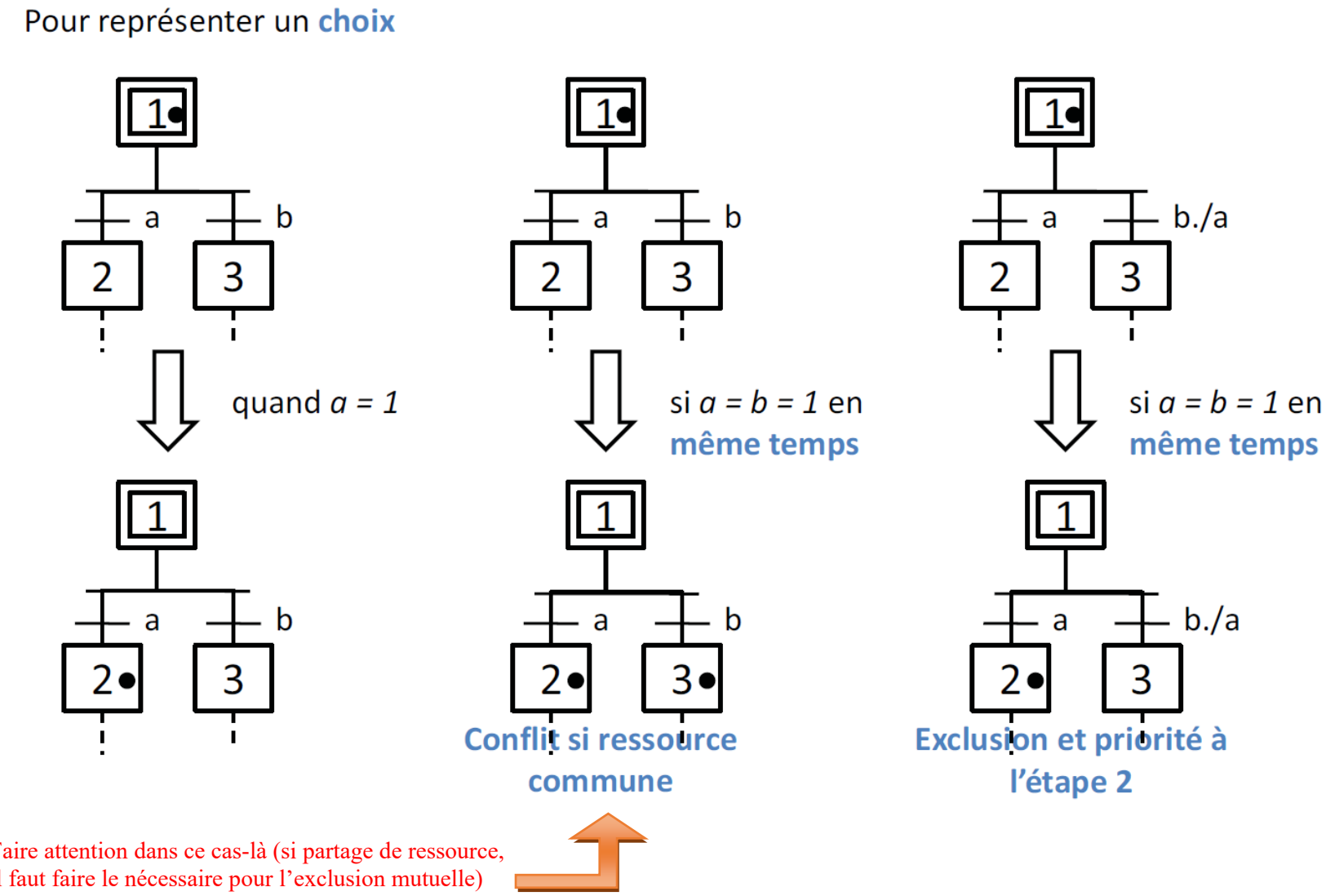


Figure 2.20. Convergence en OU

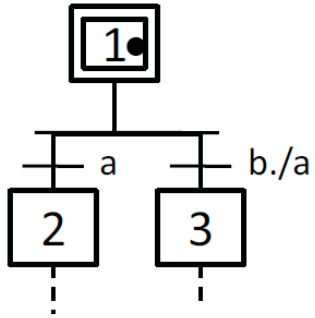
Exemple de la divergence en OU : c'est un OU inclusif.

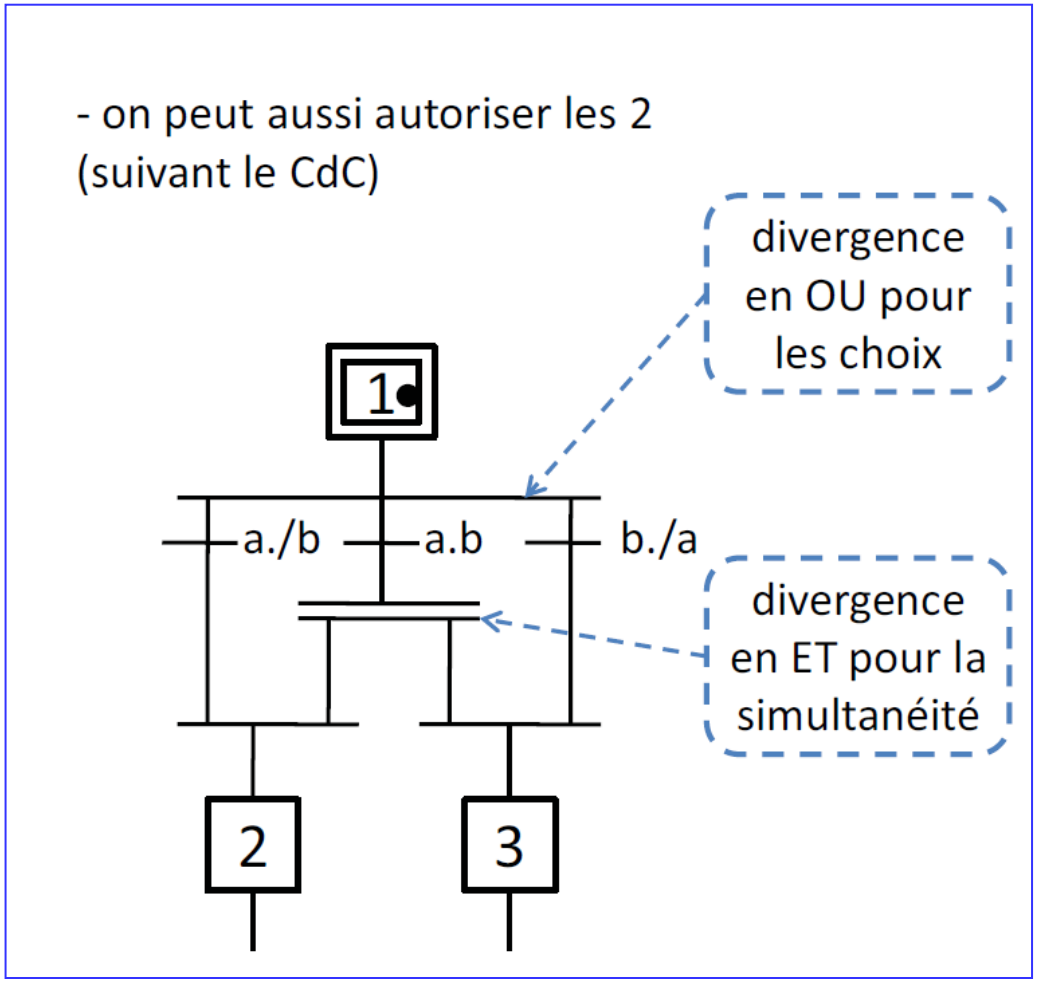


Faire attention dans ce cas-là (si partage de ressource, il faut faire le nécessaire pour l'exclusion mutuelle)

Exemple des divergences en ET et OU :

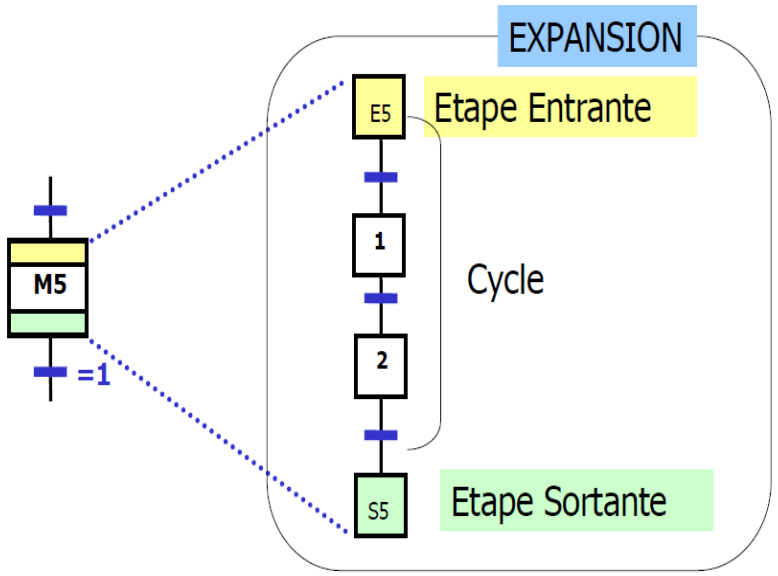
Retour sur la gestion du **conflit :**
- on avait proposé une exclusion, avec priorité à *a*





2.4 Les macro-étapes (Hiérarchisation)

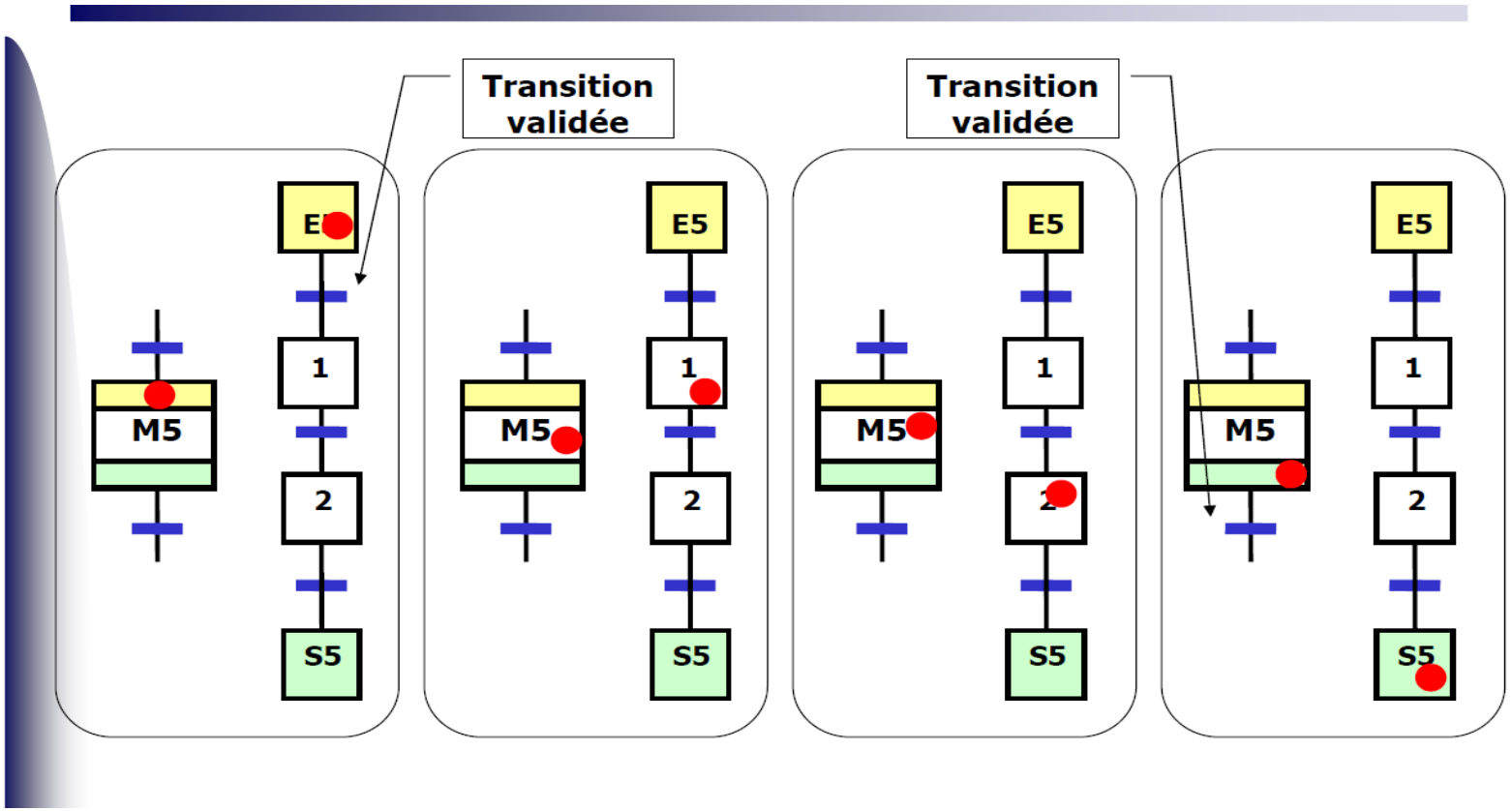
Une macro étape est une séquence ou un ensemble de séquences dans laquelle on peut entrer par une seule étape et de laquelle on ne peut sortir que par une seule autre étape. L'intérêt est d'éviter la répétition de cette séquence lorsqu'on a à l'utiliser plusieurs fois dans une SFC. Mais ce n'est pas un sous programme car cette séquence sera insérée chaque fois dans le programme lors de sa compilation dans la machine cible. La figure 2.21 donne un exemple de macro étape.



M5 représente un ensemble unique d'étapes transitions (expansion de M5). L'unique étape d'entrée est active dès q'une transition d'entrée de M5 est franchie. La ou les transitions de sortie de M5 ne sont validées que si l'étape de sortie S5 est active. Il n'existe dans la macro étape aucune liaison avec graphe principal autre que M5.

➔ *Comportement dynamique d'une macro-étape* :

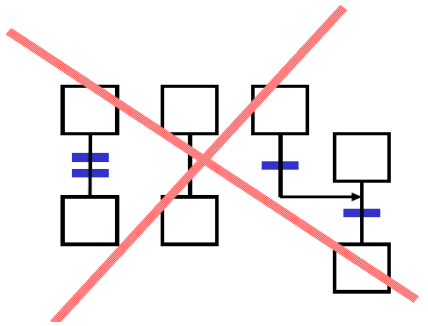
Une macro-étape est active si l'une de ses étapes 'internes' est active.



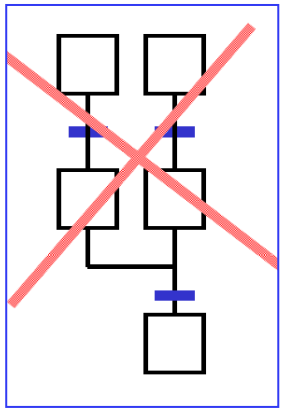
Résumé de ce qu'il ne faut pas faire

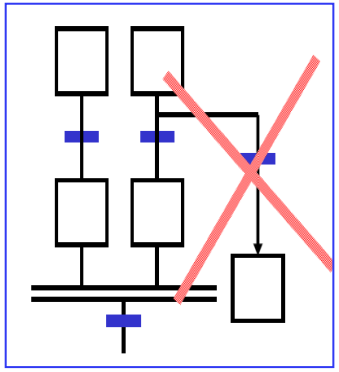
(1) Règles de structuration graphique

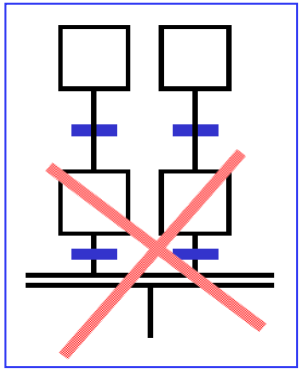
Respecter
l 'alternance
étape transition !

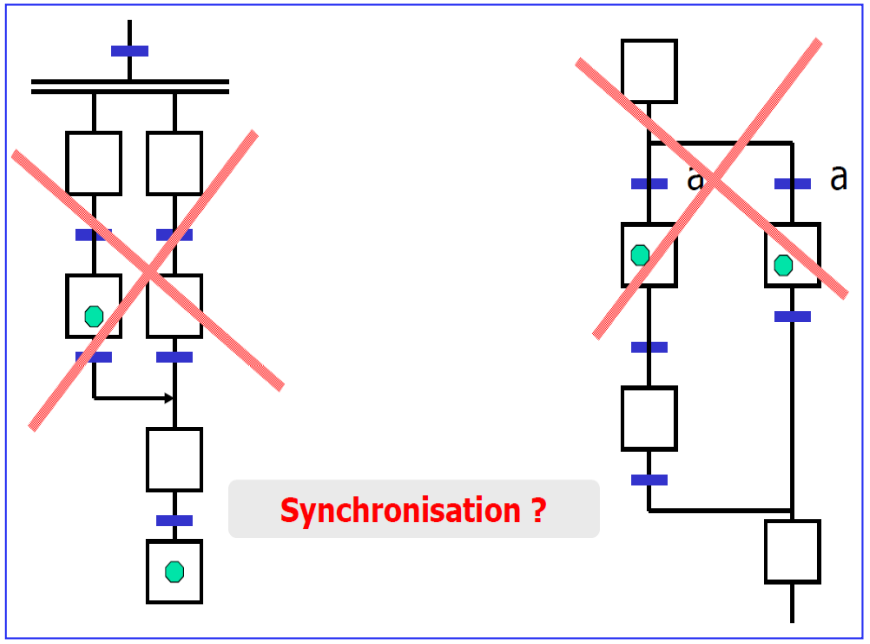


(2) Les convergences et divergences









2.5 Prise en compte du temps.

- **Simultanéité d'événements.**

La simultanéité d'occurrence d'événements externes non corrélés est exclue.

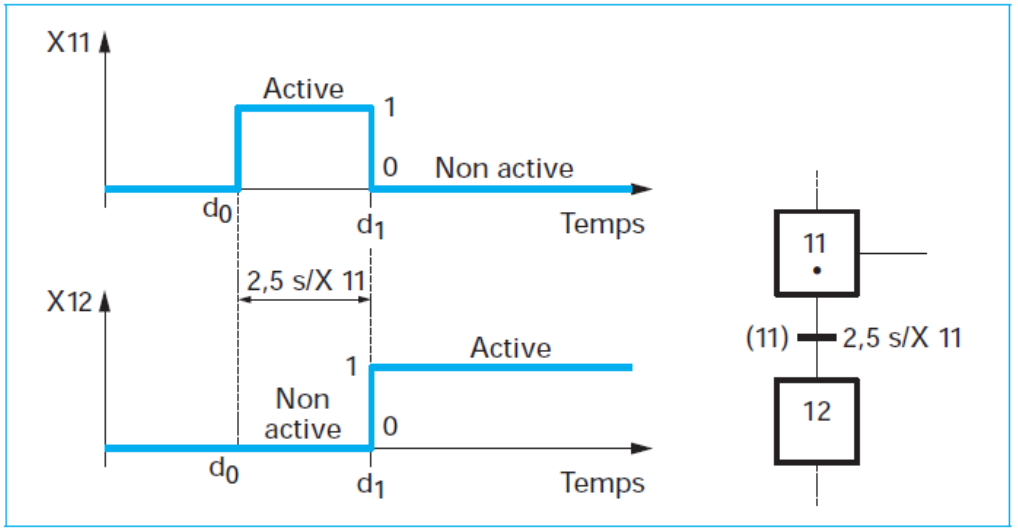
Des événements liés au changement d'état de variables non indépendantes sont bien entendu corrélés et peuvent donc être simultanés.

Des événements internes liés à l'évolution du graphe peuvent aussi être simultanés comme par exemple l'activation ou la désactivation de variables au franchissement simultané de plusieurs transitions franchissables.

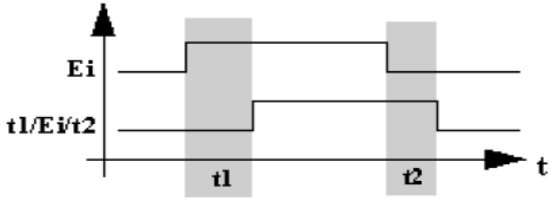
Cependant, cette notion peut souffrir de quelques **limitations technologiques** dues par exemple au caractère échantillonné des variables (difficulté à distinguer les dates de changement d'état de deux variables d'entrée dans un même cycle d'échantillonnage) et à l'impossibilité de garantir une durée nulle de franchissement des transitions.

• Réceptivité dépendant du temps (temporisation)

Le franchissement d'une transition après une durée donnée d'activité de l'étape immédiatement précédente est l'utilisation la plus courante. La réceptivité devient fausse dès la désactivation de l'étape temporisée (t1/Xi). L'étape temporisée doit rester active pendant un temps supérieur ou égal au temps indiqué pour que la réceptivité puisse être vraie (ici t1).



➔ Plus généralement, pour une étape *Ei* (*i* numéro de l'étape) on a la variable bouléenne *t1/Ei/t2* telle que :



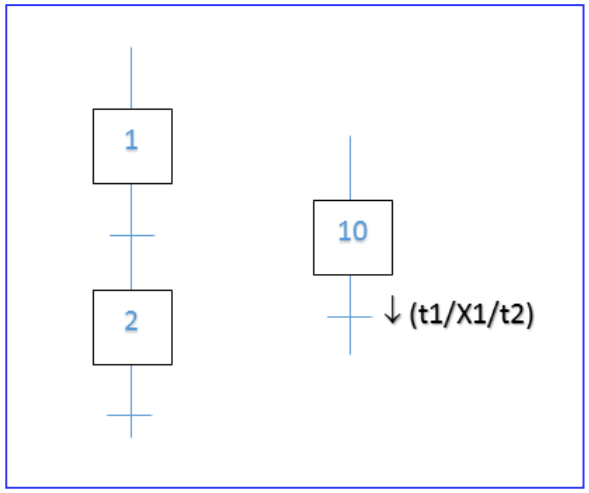
opérateur normalisé
"t1/En/t2" CEI/IEC 617-12

Notamment utilisé pour les actions conditionnelles

Une temporisation est une variable booléenne notée: $\mathbf{t_1 / X_i / t_2}$ qui est vraie (ou = 1) **depuis le temps t_1 après la dernière activation de l'étape i et jusqu'au temps t2 après sa désactivation.**

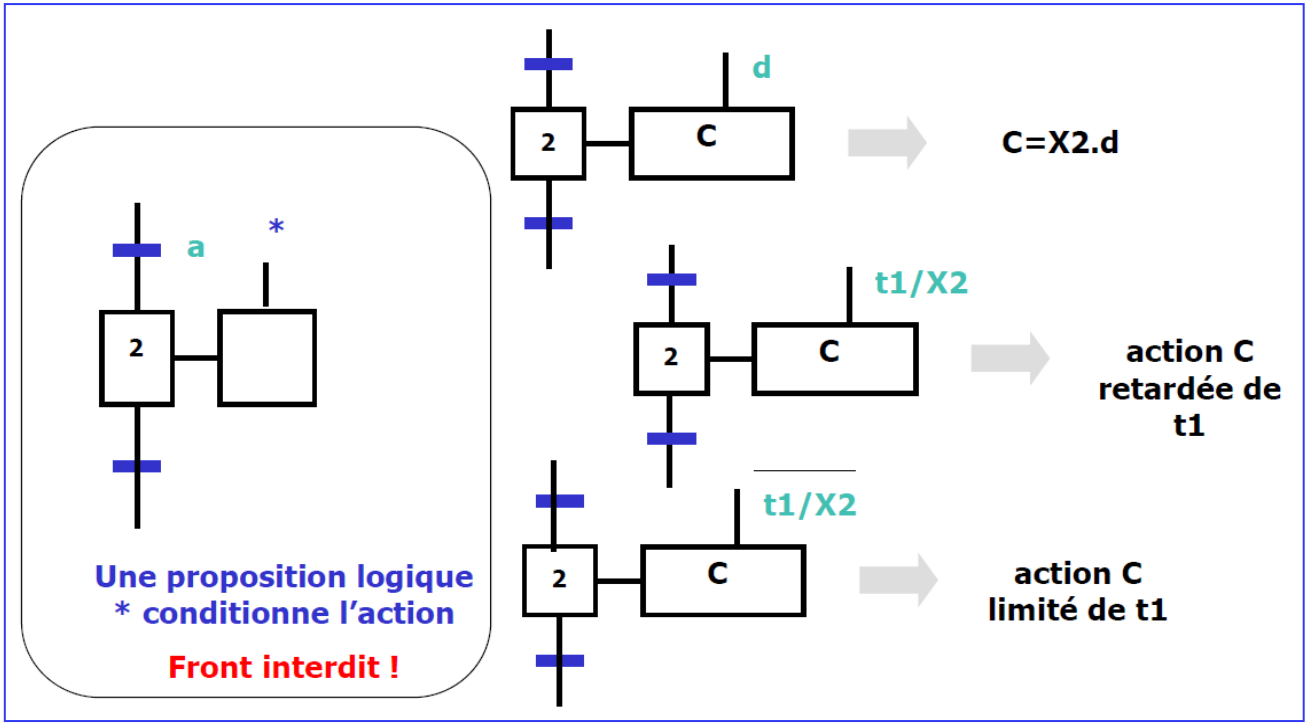
Le déclenchement d'une temporisation est assimilable à une action impulsionnelle. Une nouvelle activation de l'étape *i*, avant que le temps t_1 ne se soit écoulé, **réinitialise la temporisation**. La variable **($t_1 / X_i / t_2$)** étant une variable booléenne, on peut définir les **événements** relatifs à ses changements entre l'état 0 et l'état 1 que l'on note $\uparrow(\mathbf{t_1 / X_i / t_2})$ ou $\downarrow(\mathbf{t_1 / X_i / t_2})$.

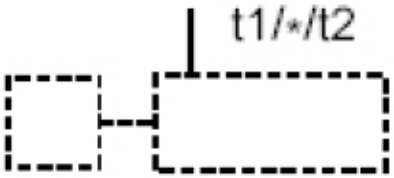
Exemple : La ci-dessous montre que la transition de sortie de l'étape 10 aura lieu (si validée), *t2* après la désactivation de l'étape 1 si cette dernière est restée active au moins *t1* unités de temps.



- **Action dépendante du temps (action conditionnelle)**

Une action conditionnelle existe si l'étape à laquelle elle est associée est vraie ET que la condition C est vraie. Celle-ci peut être une expression booléenne ou une temporisation.





Condition d'assignation dépendante du temps: La notation «t1/*/t2» indique que la condition d'assignation n'est vraie qu'après un temps t1 depuis l'occurrence du front montant (↑*, voir symbole 15) de la variable temporisée * et redevient fausse après un temps t2 depuis l'occurrence du front descendant (↓*, voir symbole 16).

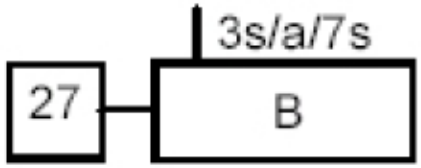
L'astérisque doit être remplacé par la variable que l'on désire temporiser, par exemple une variable d'étape ou une variable d'entrée.

t1 et t2 doivent être remplacés par leur valeur réelle exprimée dans l'unité de temps choisie.

La variable temporisée doit rester vraie pendant un temps égal ou supérieur à t1 pour que la condition d'assignation puisse être vraie.

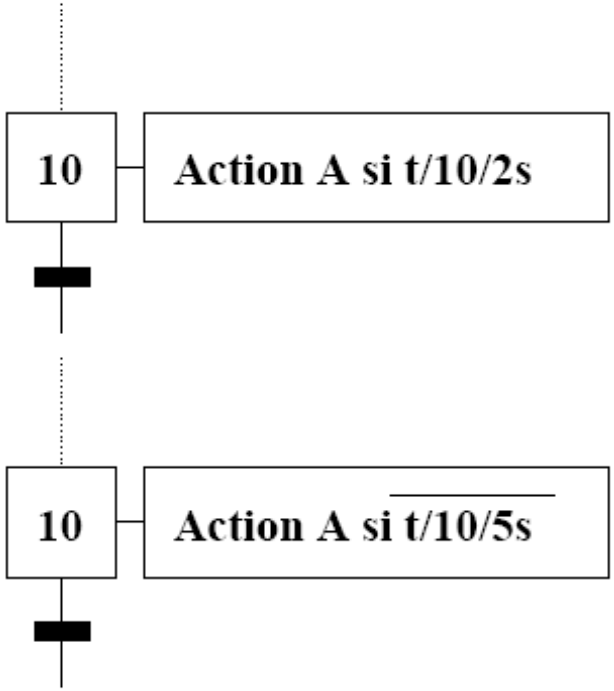
NOTE Cette notation est celle de l'opérateur à retard défini par la CEI 60617-12 (symbole n° 12-40-01).

EXEMPLE: La condition d'assignation n'est vraie que 3 s après que «a» passe de l'état «0» à l'état «1», elle ne redevient fausse que 7 s après que «a» passe de l'état «1» à l'état «0».



La valeur de la sortie B dépend de l'activité de l'étape 27 et de la valeur de la condition d'assignation (voir règles d'assignation en 4.8.1).

Exemples :



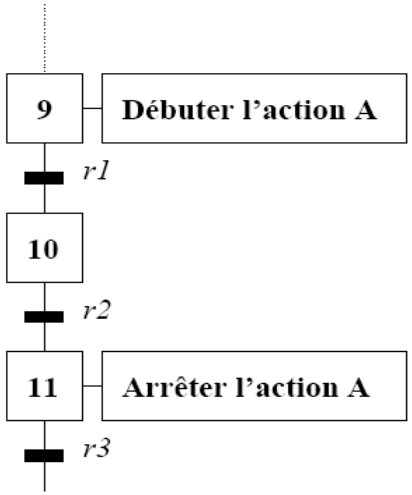
L'action A débute 2 s après l'activation de l'étape 10. En effet, la variable t/10/2s doit être vraie pour que l'action ait lieu. Or elle n'est vraie que 2 s après l'activation de l'étape 10. On remarque que si la durée de l'activation de l'étape 10 est inférieure à 2 s l'action A n'aura pas lieu.

L'action A débute dès l'activation de l'étape 10 et, à condition que cette étape reste active, dure 5 s. En effet, 5 s après l'activation de 10 la condition t/10/5s devient vraie donc son inverse t/10/5s devient fausse et l'action A s'arrête. On remarque que si la durée d'activation de l'étape 10 est inférieure à 5 s, l'action A ne durera que cette durée d'activation.

QUELQUES COMLEMENTS

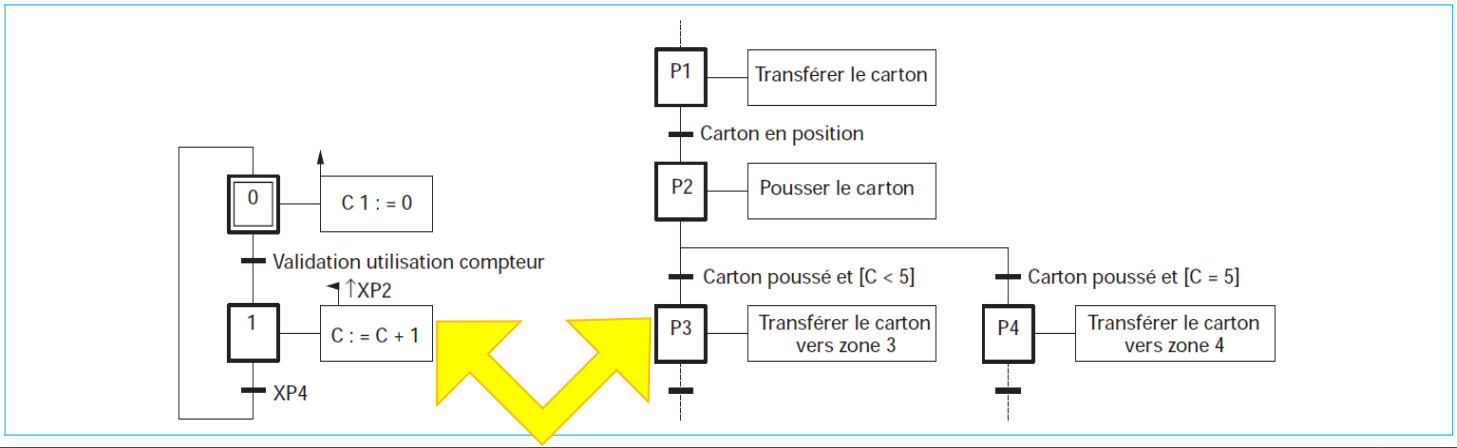
I. Action mémorisée sur événement:

Rappel :

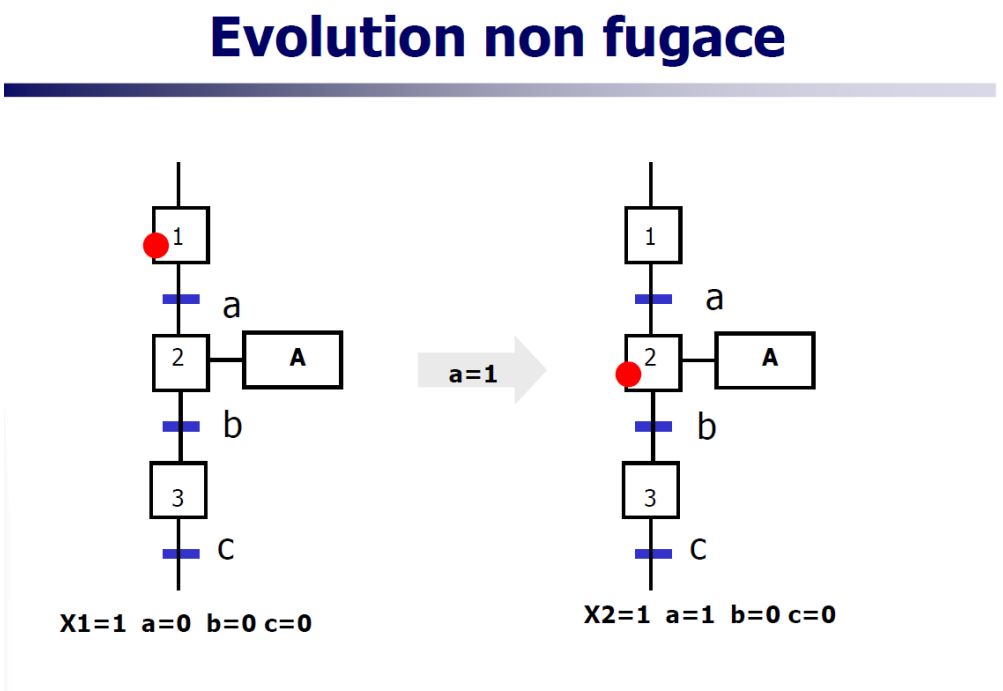


Si une action est « mémorisée », elle est l'objet de deux ordres différents qui sont « Débuter l'action » et « arrêter l'action ». Pendant toute la période s'écoulant entre l'envoi de ces deux ordres l'action à lieu.

Dans l'exemple ci-contre l'action A débute avec l'activation de l'étape 9. Elle se poursuit durant toute la durée de l'activation de l'étape 10. L'action A ne s'arrêtera qu'avec l'activation de l'étape 11



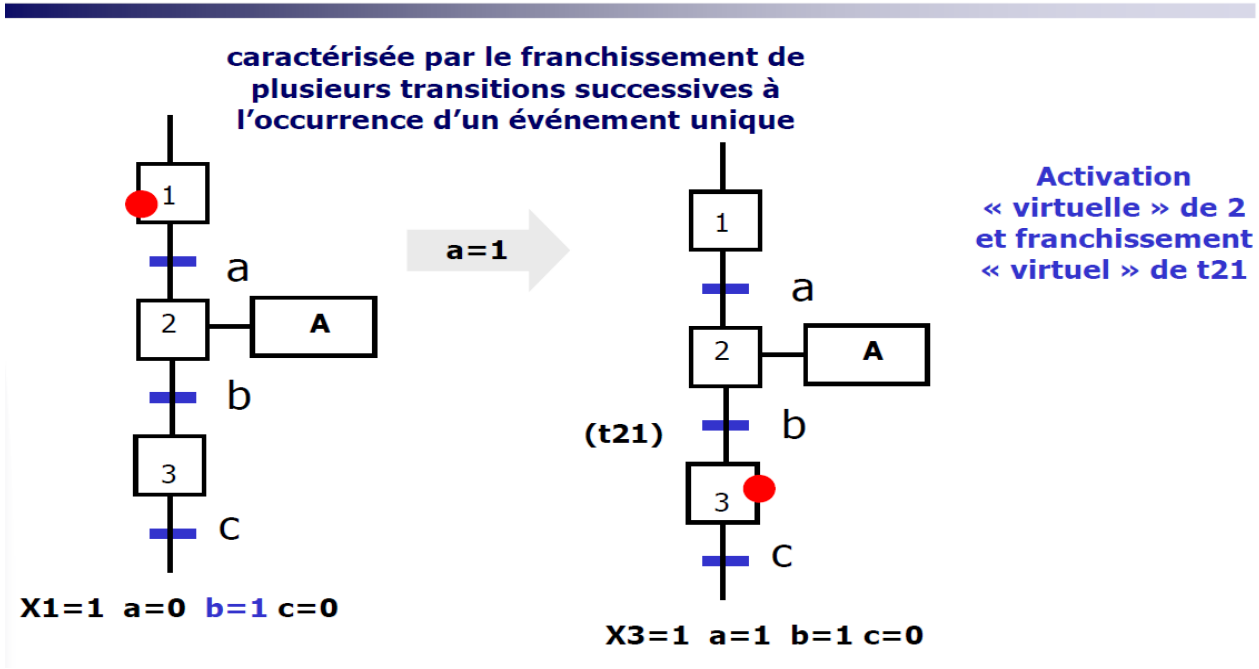
II. **Interprétation temporelle :** Evolutions fugace et non fugace.



Franchissements successives des transitions ➔ les étapes intermédiaires sont ***stables*** (=marquage stable).

II. **Interprétation temporelle :** Evolutions fugace et non fugace.

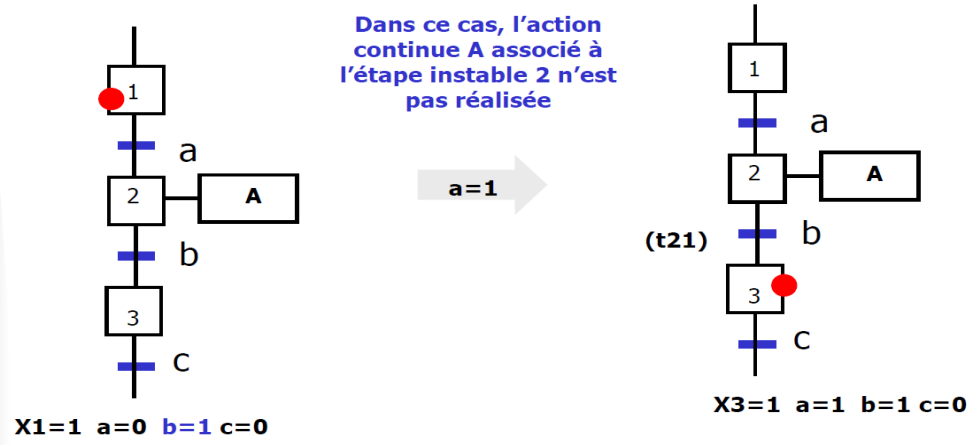
Evolution fugace



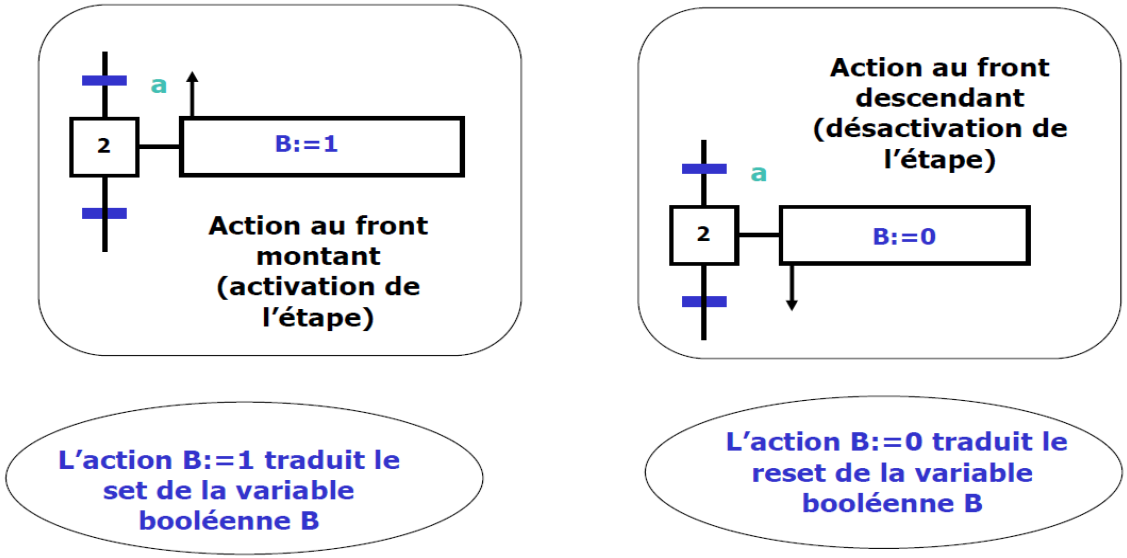
Au moment de franchir certaines transitions, des transitions intermédiaires sont déjà vraies ➔ étapes intermédiaires non activées mais considérées comme virtuellement activées et désactivées. Elles sont ***instables*** (=marquage instable). De même pour les transitions correspondantes (franchissement virtuel).

⇨ **Conséquence sur les assignations (selon le type d'actions associées à ces étapes instables)**

II. Interprétation temporelle : (a) Conséquence évolution fugace sur action continue (permanente)

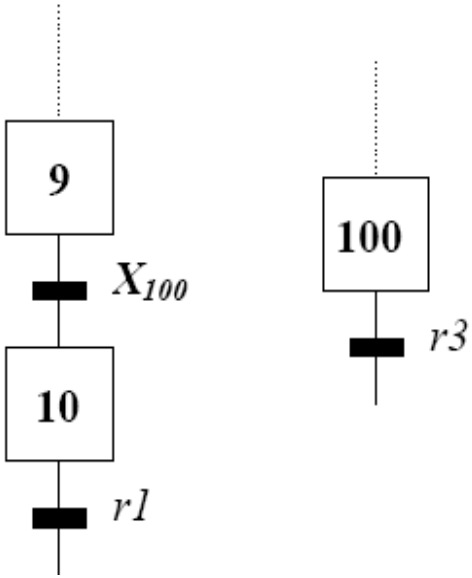


(b) Conséquence évolution fugace sur action mémorisée (impulsionnelle)



III. Réceptivité faisant intervenir une variable d'étape

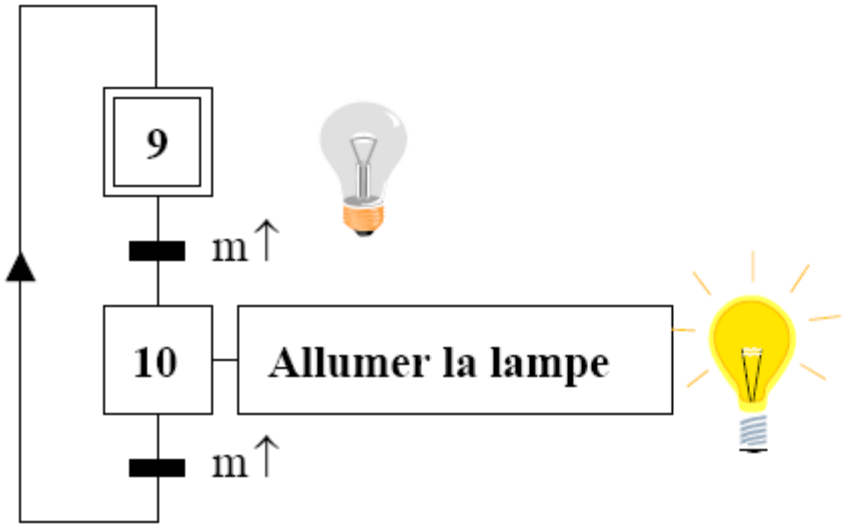
On peut utiliser, dans la fonction booléenne d'une réceptivité la variable X_i. Cette variable vaut « 1 » (ou est vraie) lorsque l'étape i est active. Elle vaut « 0 » (ou est fausse) dans le cas contraire.



Cette variable Xi est très importante car elle permet, lorsqu'un grafcet est constitué de plusieurs graphes, de faire des synchronisations entre ces graphes. Ainsi, dans l'exemple ci-contre, pour activer l'étape 10 il faut que les étape 9 et 100 soient actives.

IV. Réceptivité faisant intervenir les fronts d'une variable

Il y a deux façons de considérer une variable booléenne dans une réceptivité :
On peut prendre en compte son état. La réceptivité est vraie lorsque, et tant que, l'état de la variable booléenne est à « 1 ».
On peut prendre en compte les fronts de la variable. Dans le cas d'un « front montant », la réceptivité noté r = a↑ n'est vraie qu'a l'instant du passage de a = 0 à a = 1. Elle est fausse avant et après cet instant. Elle n'est vraie que pendant la durée nécessaire au franchissement de la transition à laquelle elle est associée. De même, dans le cas d'un « front descendant », la réceptivité noté r = b↓ n'est vraie qu'à l'instant du passage de b = 1 à b = 0. Elle est fausse avant et après cet instant. Elle n'est vraie que pendant la durée nécessaire au franchissement de la transition à laquelle elle est associée. La notion de front permet de simplifier un grafcet. Reprenons l'exemple du chapitre 4.2. Le grafcet suivant est équivalent au grafcet donné dans l'exemple.



La lampe s'allume lors d'un front montant de m. elle s'éteint lors du font montant suivant de m. On remarque bien que le front montant de m ne permet que le franchissement d'une seule transition.

On peut associer dans une même réceptivité plusieurs variables sur front ou des variables sur front et des variable sur état. Quelques exemples :

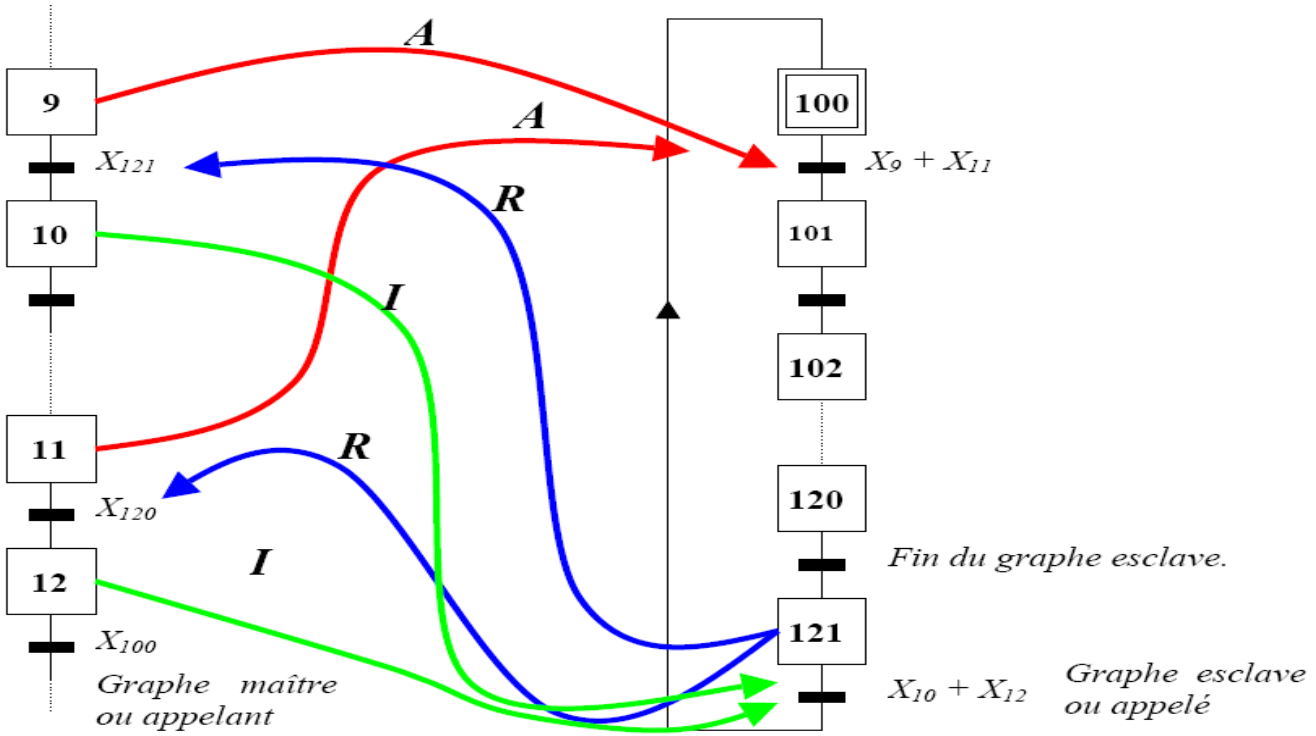
Réceptivité	Signification
(a+b) ↑	La réceptivité est vraie lorsque la fonction a+b subit un front montant. Il faut donc que l'on ait simultanément a=0 et b=0 puis que a ou b passe à « 1 ».
(a.b) ↑	La réceptivité est vraie lorsque la fonction a.b subit un front montant. Cela signifie que l'on à un front montant sur a alors que b=1 ou que l'on a un front montant sur b alors que a=1.
c+a ↑	La réceptivité est vraie soit si c est à l'état « 1 » soit si on a un front montant sur a.
c.a ↑	La réceptivité est vraie si on a simultanément c à l'état « 1 » et un front montant sur a.
a ↑ +b ↑	La réceptivité est vraie si on a un front montant sur a ou sur b.
a ↑ .b ↑	L'occurrence simultanée de deux événements étant théoriquement impossible cette réceptivité est toujours fausse.

V. Grafcet multi-graphes

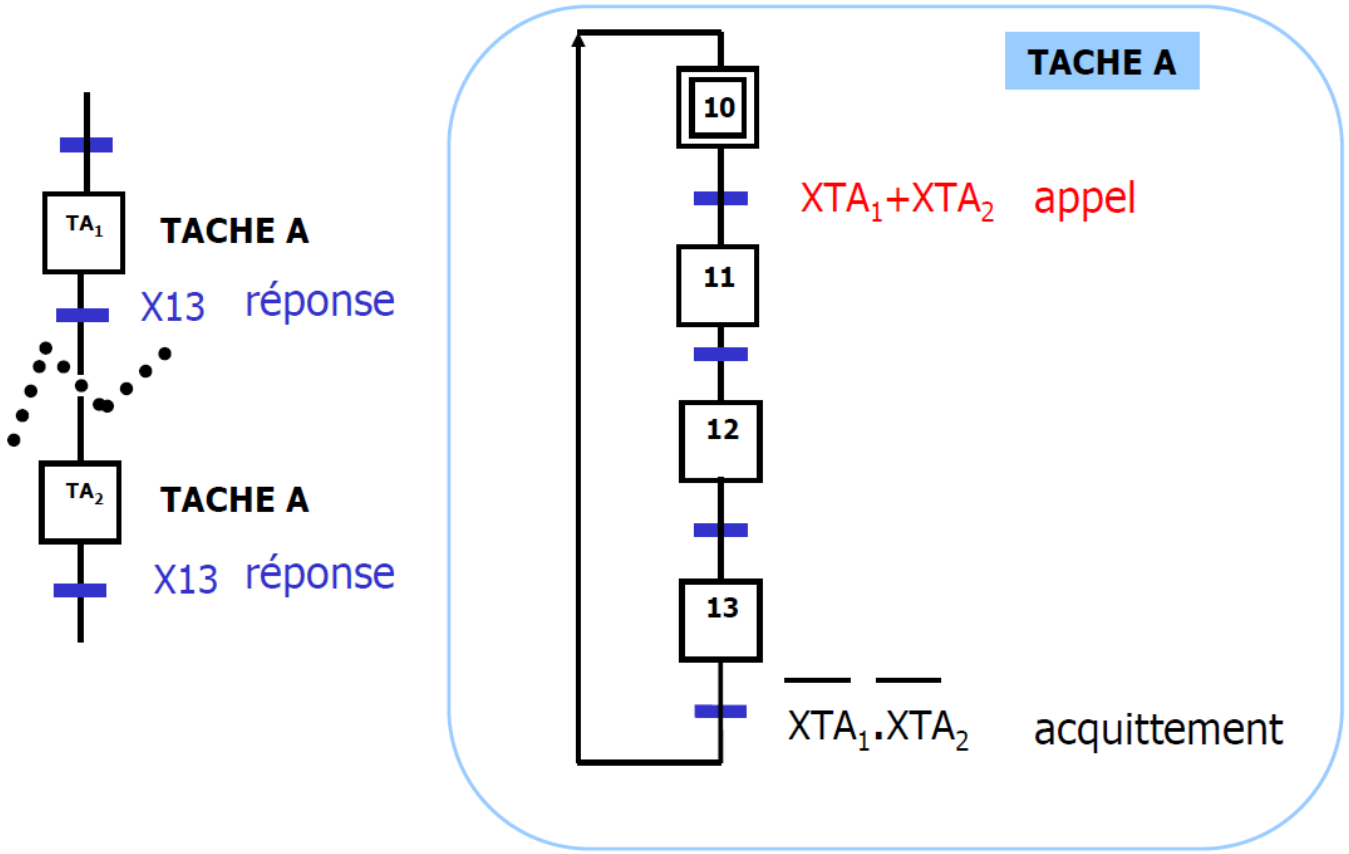
Un grafcet est dit multi-graphes s'il est constitué de plusieurs graphes. C'est une possibilité très importante qui permet de :

- Structurer un grafcet complexe (le rendre lisible) ;
- Construire un G7 de grande envergure en le décomposant en plusieurs parties.

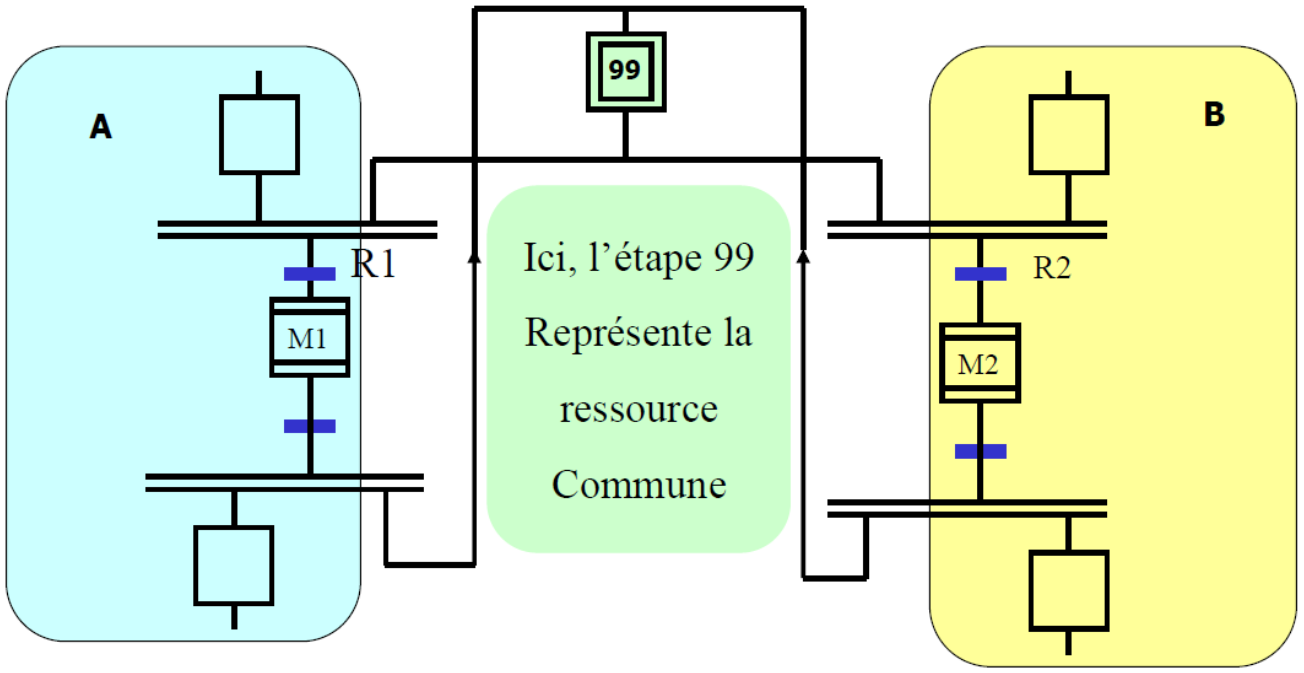
La mise en œuvre de plusieurs graphes nécessite des synchronisations entre ceux-ci. La technique décrite ici est dite « Appel/Réponse » : les étapes 9 et 11 sont des « étapes d'appel » car elles permettent de lancer le graphe esclave (*A*) au moyen des réceptivités *X9+X11*. Lorsque le graphe esclave a terminé son évolution, il informe (*R*) le graphe maître au moyen de l'étape 121. Le graphe maitre réarme alors le graphe esclave (*I*) au moyen des étapes 10 et 12. Cette structure interdit relancer le graph esclave s'il n'est pas initialisé.



Exemple d'utilisation en sous-programmes :

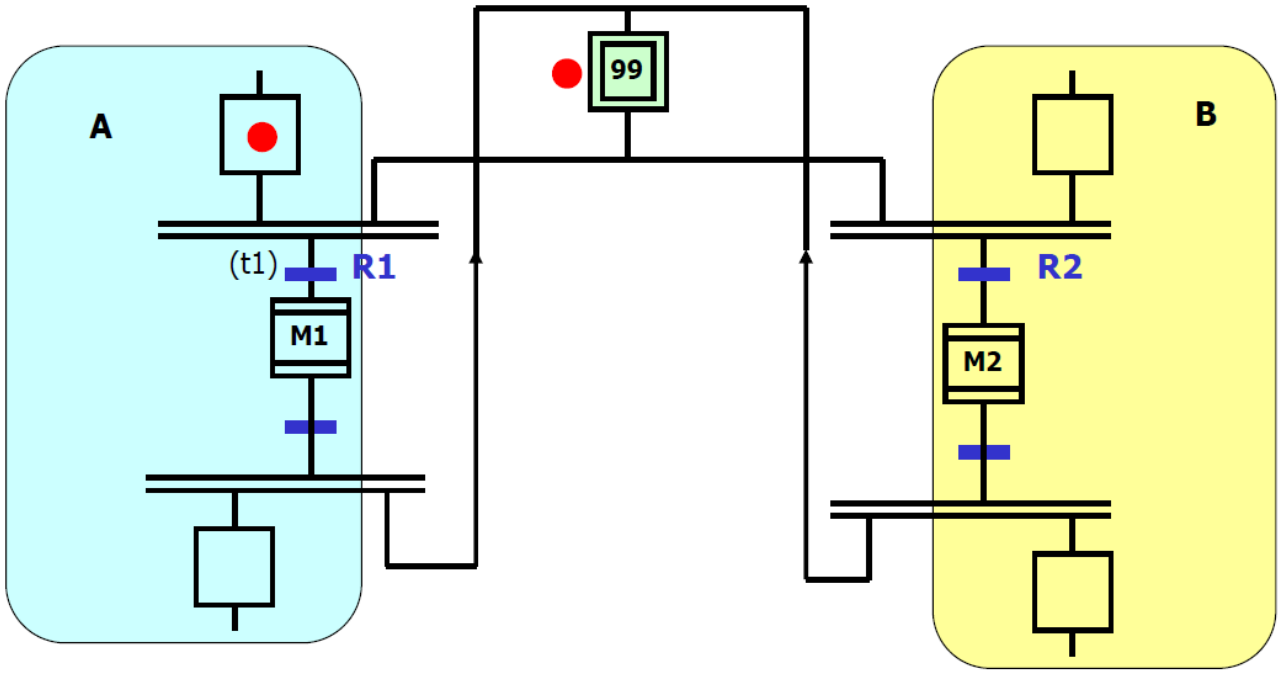


Ressource Commune



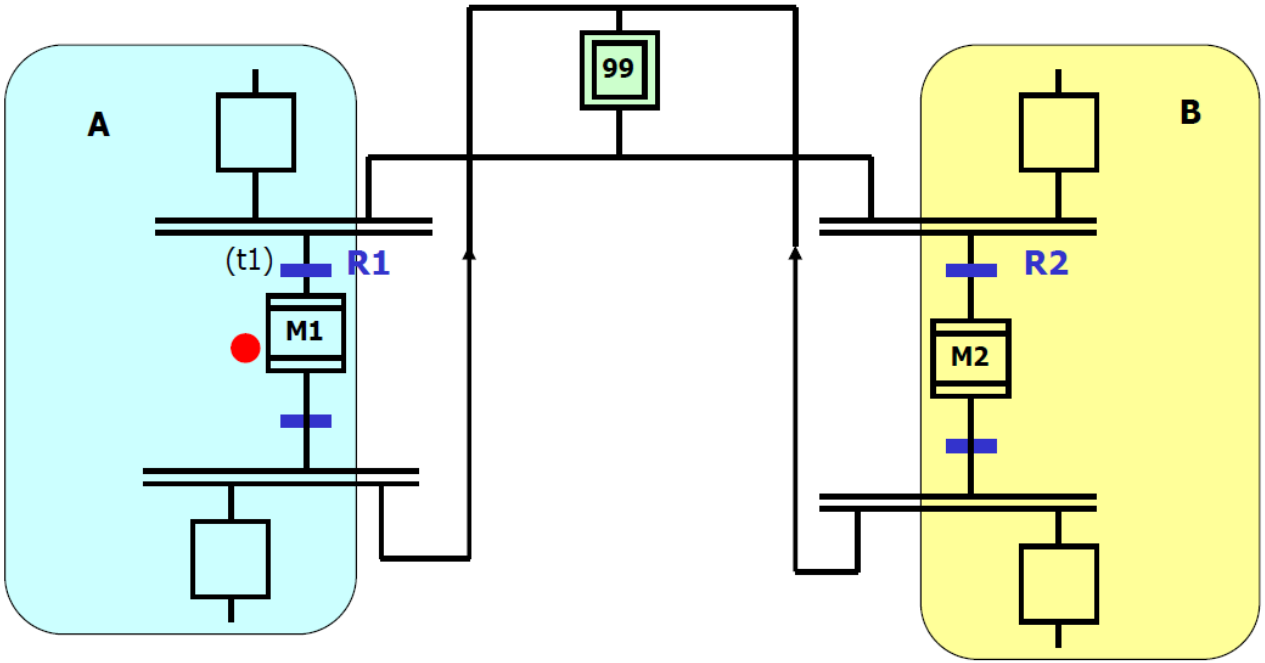
La ressource peut être utilisée par A (macro M1) ou B (macro M2)

Comportement Dynamique



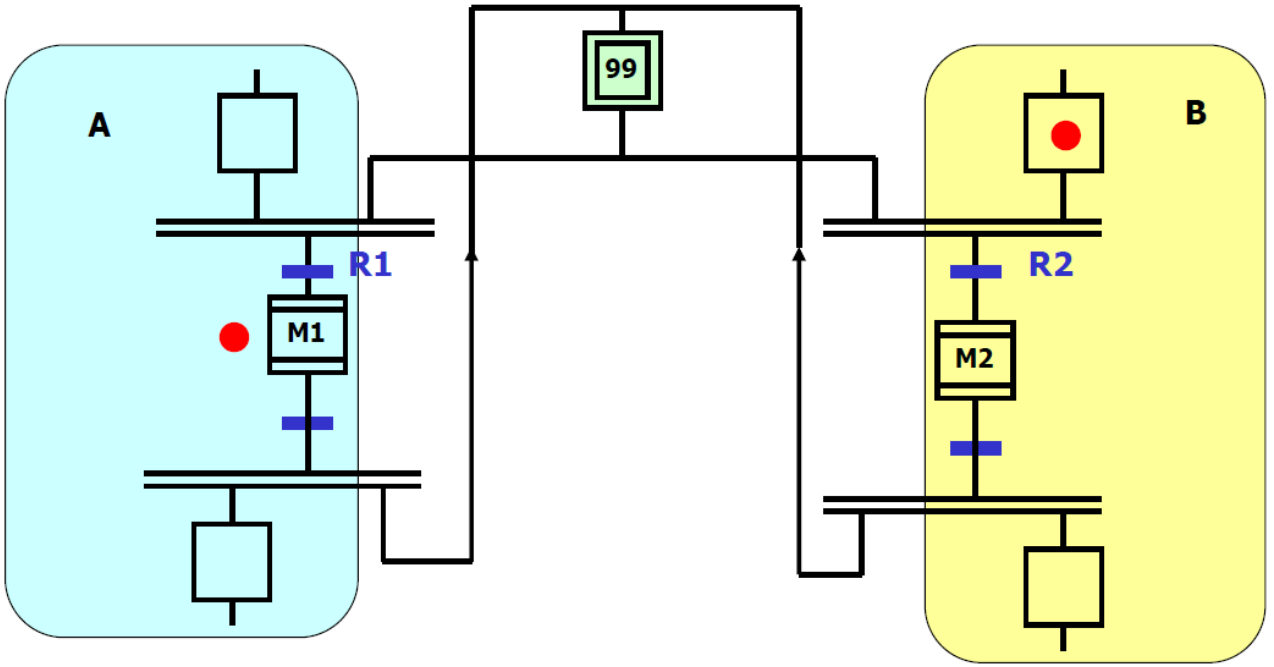
L'étape 99 est active, la « ressource » est libre

Comportement Dynamique



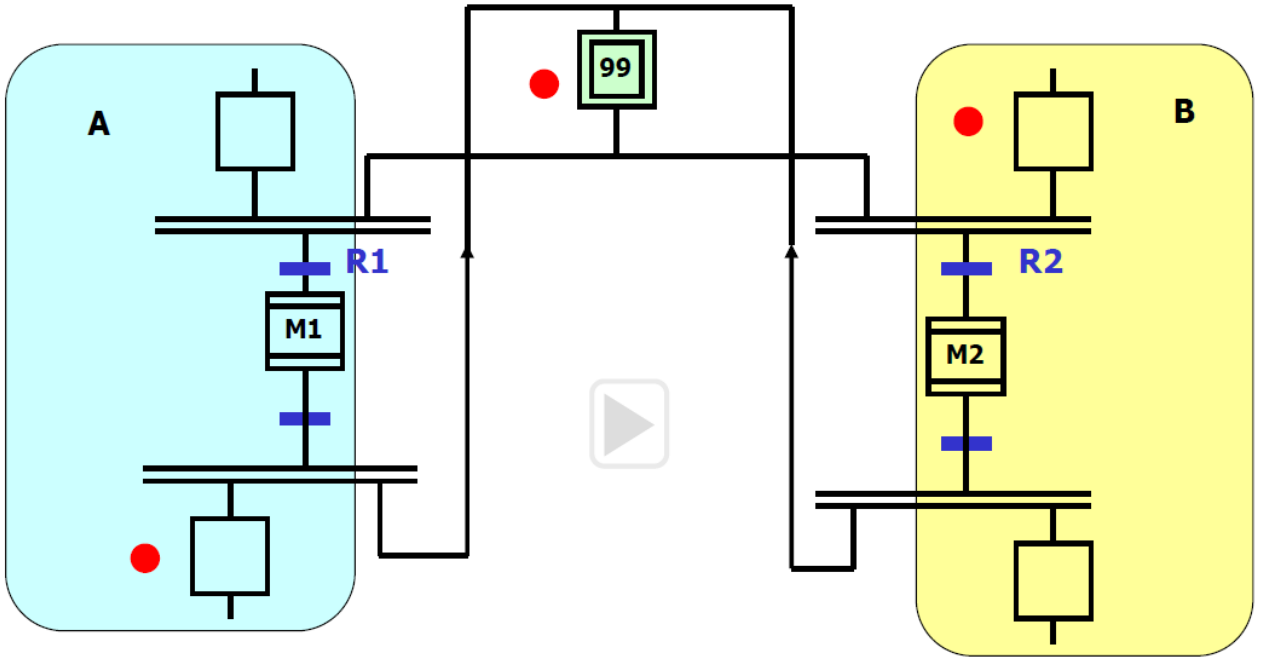
Le franchissement de la transition (t1) entraîne la désactivation de l'étape 99. La ressource est donc utilisée par M1

Comportement Dynamique



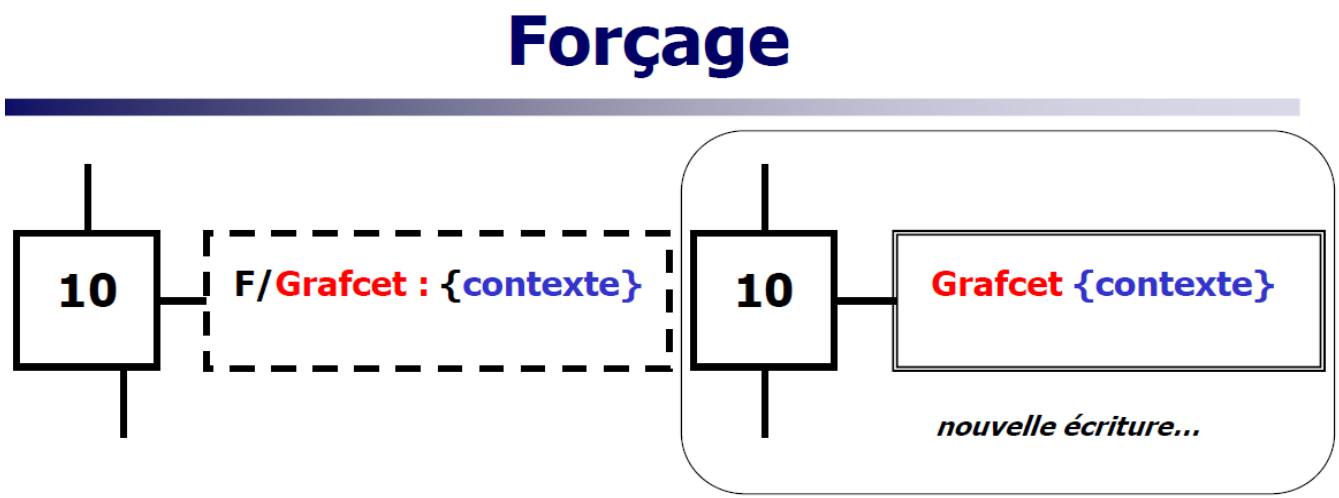
Le processus B doit attendre la fin du processus A (M1) pour utiliser la ressource qui n'est plus disponible (étape 99 inactive)

Comportement Dynamique



La fin de la macro M1 entraîne la réactivation de l'étape 99. La ressource est de nouveau disponible pour B par exemple

Toujours concernant le grafcet multi-graphes…



Le forçage est un ordre interne consécutif à une évolution.

L'application du forçage est prioritaire par rapport à toute évolution.

Les actions associées aux étapes des grafcets forcés sont maintenues pendant la durée du forçage !

Le grafcet forcé ne peut évoluer tant que l'ordre de forçage est présent.

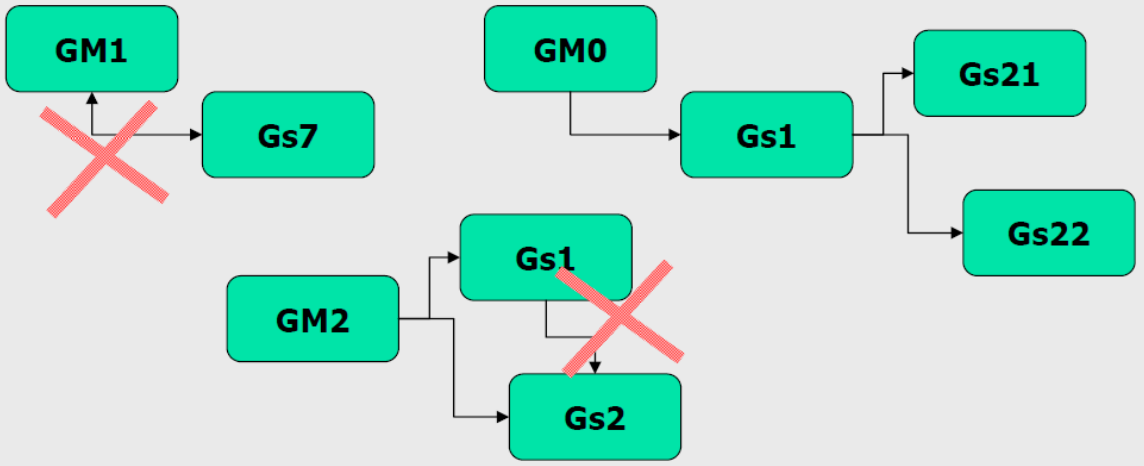
Toujours concernant le grafcet multi-graphes…

Cohérence

La cohérence de la hiérarchie impose que :

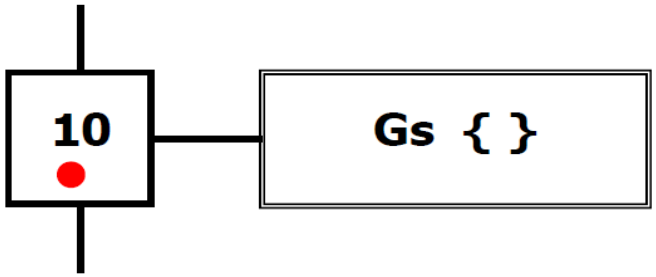
Si un grafcet force un autre grafcet, la réciproque est impossible

Un grafcet ne peut être forcé que par un et un seul grafcet

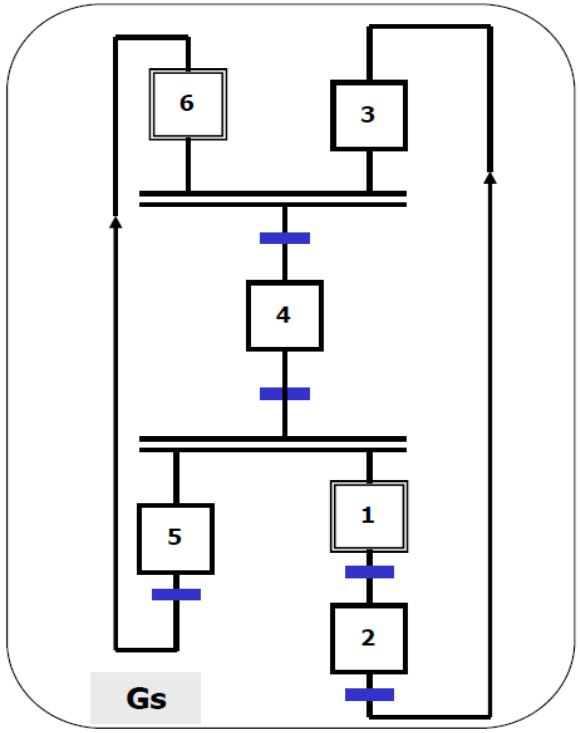


Toujours concernant les grafcet multi-graphes…

Forçage Dans La Situation Vide (Désactivation)

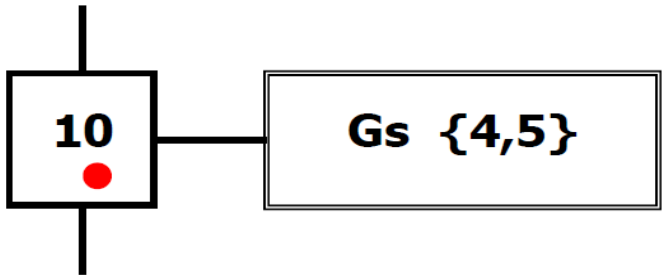


L'activation de l'étape 10 entraîne la désactivation de toutes les étapes du grafcet Gs

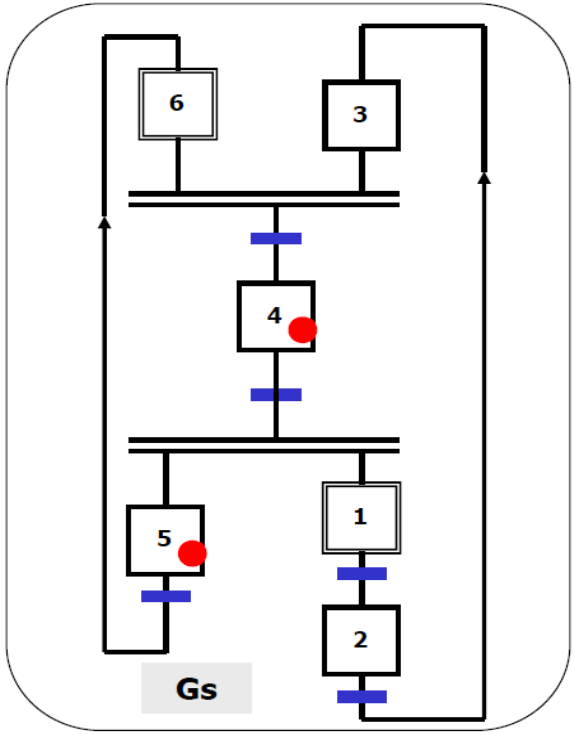


Toujours concernant les grafcets multi-graphes…

Forçage Dans Une Situation Donnée

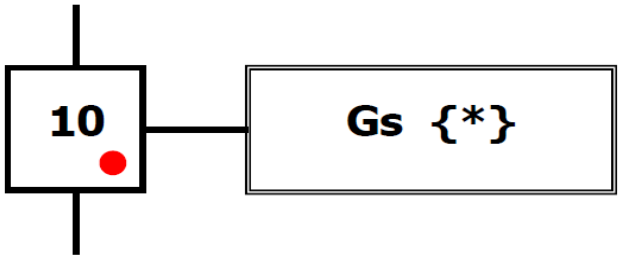


L'activation de l'étape 10 entraîne l'activation des étapes 4 et 5 du grafcet Gs et le maintient dans ce contexte tant que l 'ordre de forçage est émis

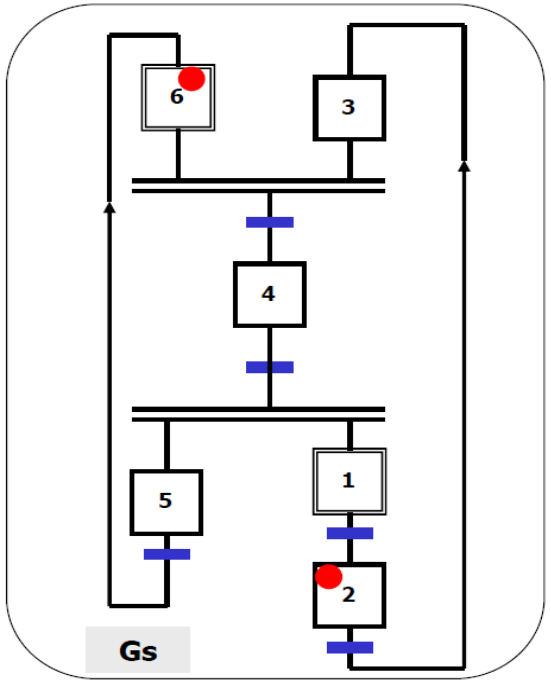


Toujours concernant les grafcets multi-graphes…

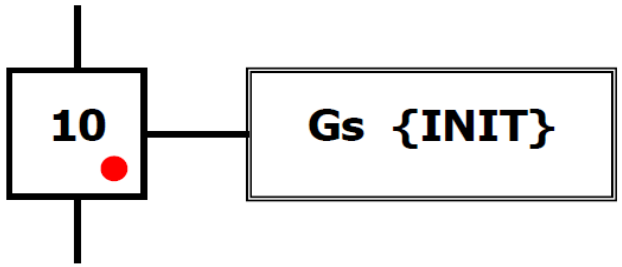
Forçage Dans La Situation Courante :Figeage



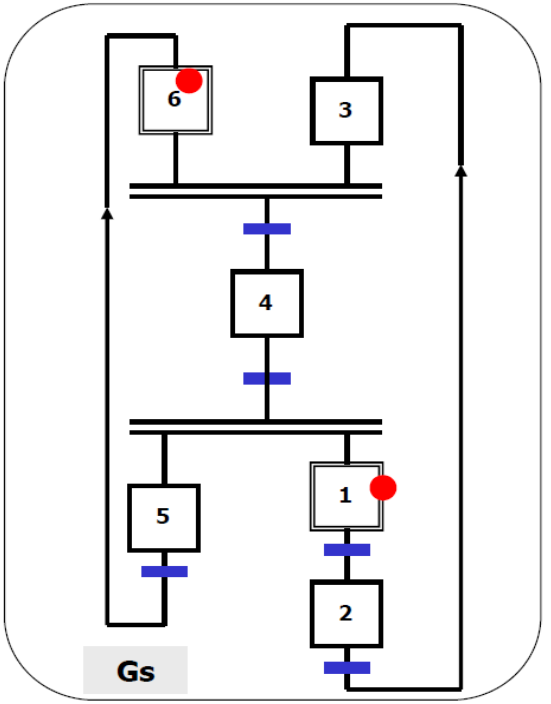
L'activation de l'étape 10 entraîne le figeage du grafcet Gs dans la situation courante et le maintient dans ce contexte tant que l 'ordre de forçage est émis



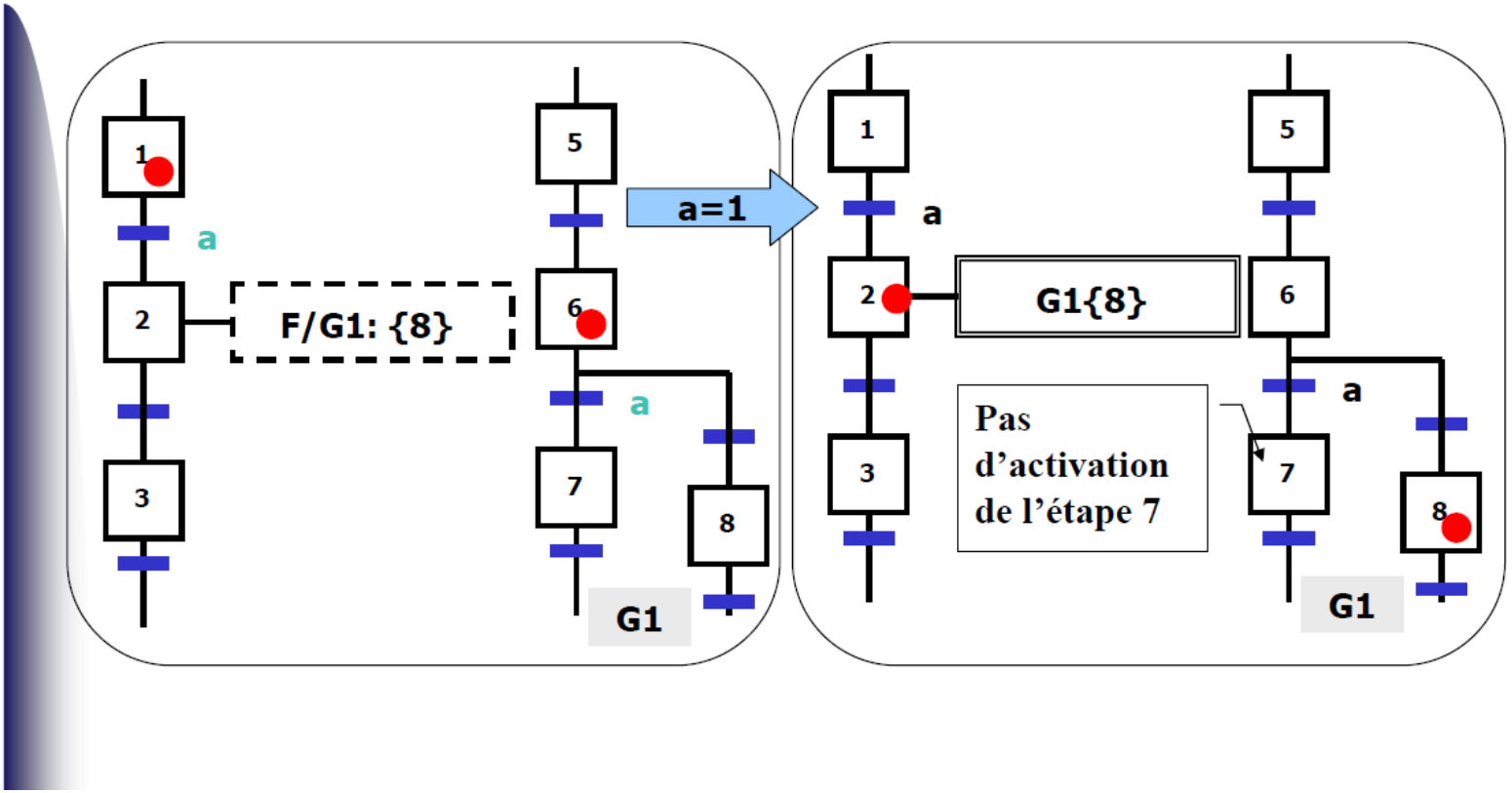
Forçage Dans La Situation Initiale



L'activation de l'étape 10 entraîne l'initialisation du grafcet Gs et le maintient dans ce contexte tant que l 'ordre de forçage est émis



Exemple


1
a
2
F/G1: {8}
3
5
6
a
7
8
G1
a=1
1
a
2
G1{8}
3
Pas
d'activation
de l'étape 7
5
6
a
7
8
G1

Récapitulatif de la sémantique d'évolution du Grafcet
(5 Règles)

Règle 1 : SITUATION INITIALE
La situation initiale est caractérisée par les étapes actives au début du fonctionnement.

Règle 2 : FRANCHISSEMENT D'UNE TRANSITION
Une transition est dite **VALIDEE** lorsque toute les étapes immédiatement précédentes reliées à cette transition sont actives. Une transition est **FRANCHISSABLE** lorsque la transition est validée ET lorsque la réceptivité associée à cette transition est vraie. Une transition franchissable est obligatoirement franchie.

Règle 3 : ÉVOLUTION DES ÉTAPES ACTIVES
Le franchissement d'une transition entraîne **SIMULTANÉMENT** l'ACTIVATION de toutes les étapes immédiatement suivantes et la DÉSACTIVATION de toutes les étapes immédiatement précédentes reliées à cette transition.

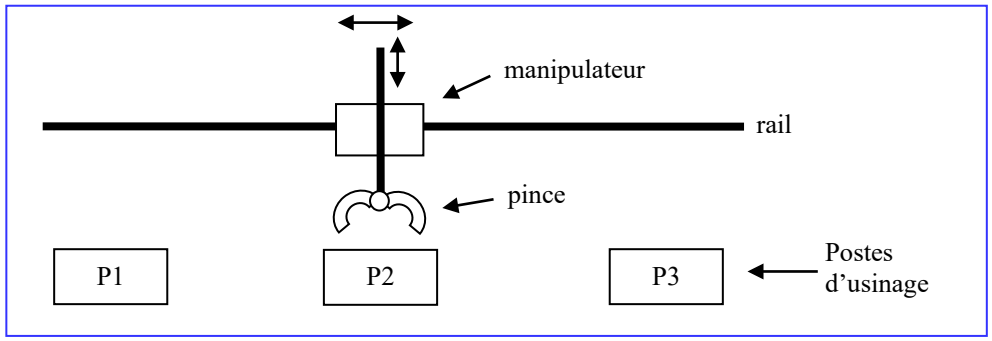
Règle 4 : ÉVOLUTIONS SIMULTANÉES
Plusieurs transitions **SIMULTANÉMENT** franchissables sont simultanément FRANCHIES.

Règle 5 : ACTIVATION ET DÉSACTIVATION SIMULTANÉE D'UNE ÉTAPE
Si une même étape est **SIMULTANÉMENT** ACTIVÉE et DÉSACTIVÉE, elle reste ACTIVE.

Exercice : Commande d'un bras manipulateur

La fabrication d'un objet manufacturé nécessite des usinages multiples exécutés par 3 postes alignés (machines) P1, P2 et P3. Pour déplacer cet objet d'un poste à l'autre, on utilise un manipulateur M muni d'une pince P, qui se déplace sur un rail.

A l'arrêt, le manipulateur se trouve devant le poste où se trouve la pièce.



Le manipulateur possède les commandes suivantes :

- Fermer la pince : **fp**
- retirer l'objet : **ro**
- Déposer l'objet : **do**
- Ouvrir la pince : **op**
- Déplacement à droite : **dd**
- Déplacement à gauche : **dg**

On dispose des capteurs suivants :

- Objet déposé : **od**
- Objet sur manipulateur : **om**
- Pince ouverte : **po**
- Pince fermée : **pf**
- Manipulateur devant P1 : **mp1**
- Manipulateur devant P2 : **mp2**
- Manipulateur devant P3 : **mp3**

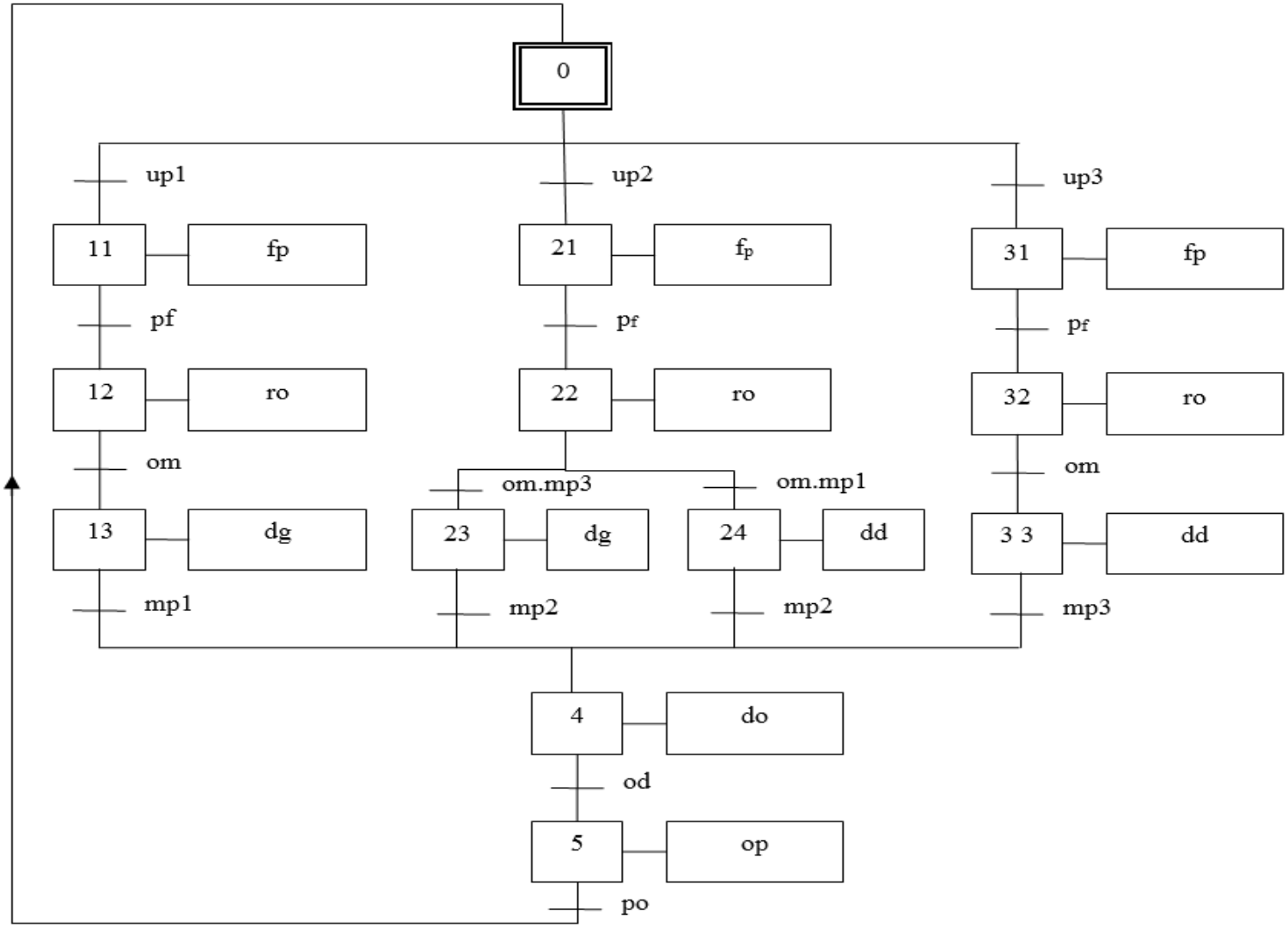
Le programme de fabrication fournit les demandes d'usinage :

- Usinage par le poste ***i*** : **upi**

On suppose que l'objet étant en position j (et le bras devant l'objet à transporter), il ne peut y avoir de demande upj.

Construire le GRAFCET de commande du manipulateur.

Corrigé :


0
up1
up2
up3
11
fp
21
fp
31
fp
pf
pf
pf
12
ro
22
ro
32
ro
om
om.mp3
om.mp1
om
13
dg
23
dg
24
dd
3 3
dd
mp1
mp2
mp2
mp3
4
do
od
5
op
po

6.2 Les niveaux de sécurité d'un système automatisé :

La sécurité de fonctionnement est assurée à trois niveaux :

6.2.1 Sécurité de niveau 2 :

C'est la sécurité **directement assurée par la partie commande**. Lors de la définition du cahier des charges de cette partie commande, on prévoit une comportement de la machine qui en toute occasion permet d'assurer qu'il n'y a aucun risque pour l'opérateur ou pour la machine.

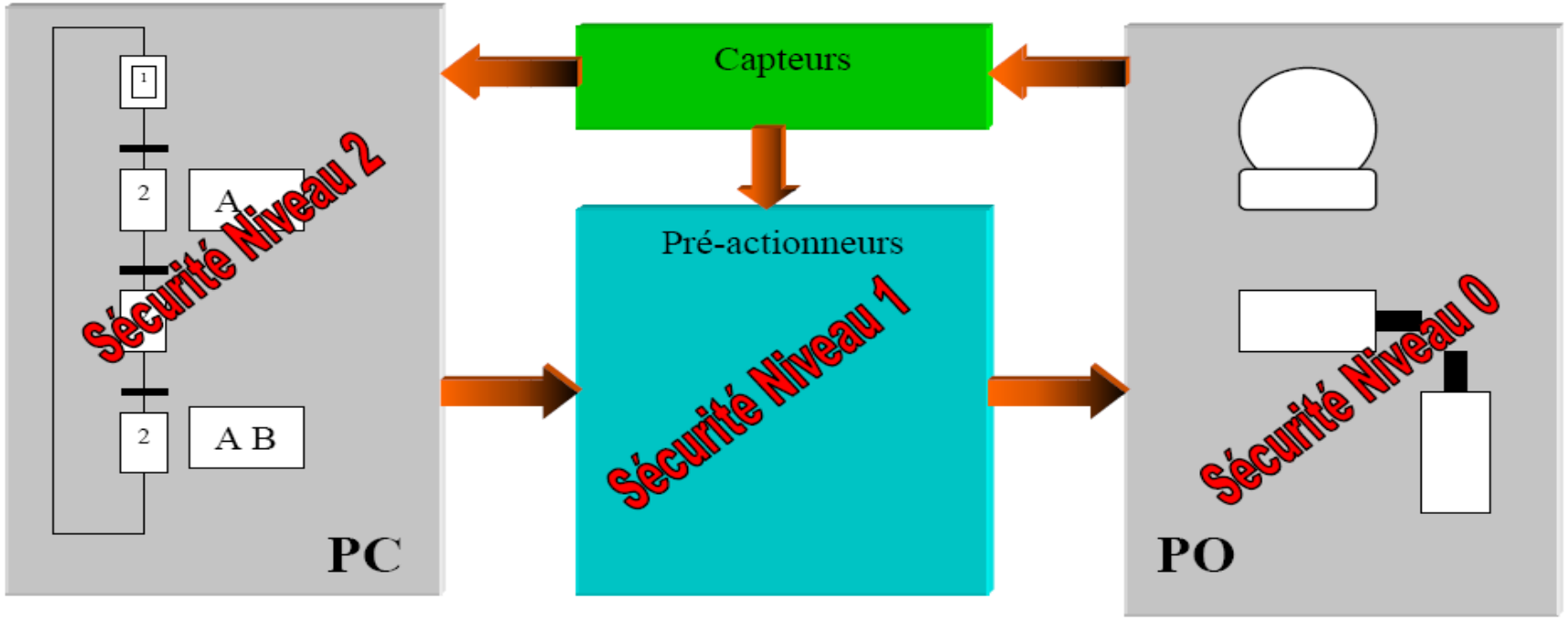
Mais bien évidemment, la partie commande elle même peut être sujette à défaillances. Il faut donc prévoir d'autres dispositif de sécurité.

6.2.2 Sécurité de niveau 1

On place des dispositifs entre la partie commande et la partie opérative, qui permettent d'assurer un minimum de sécurité en cas de défaillance de la partie commande. C'est l'un des rôle des pré-actionneurs.

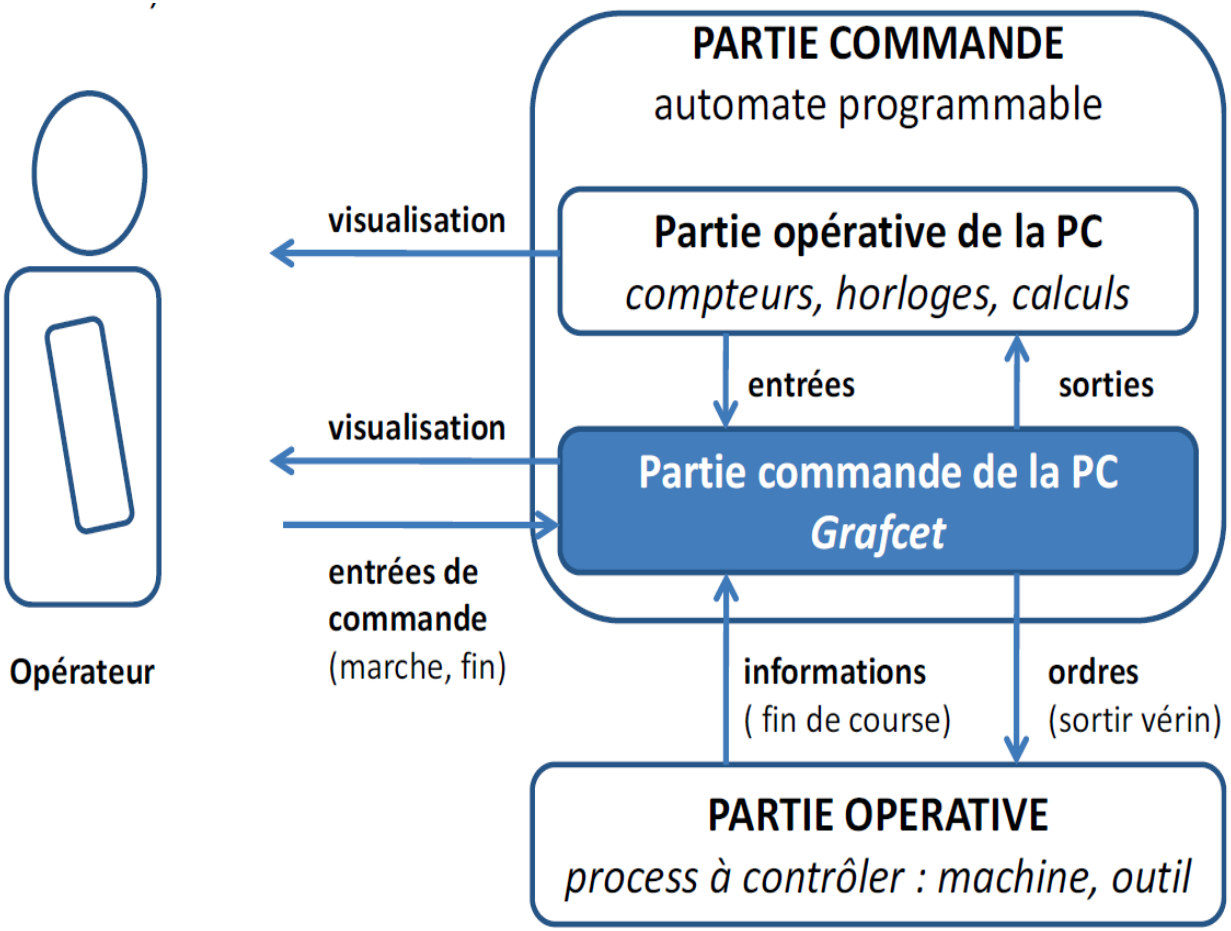
6.2.3 Sécurité de niveau 0

S'il y a défaillance simultanée de la partie commande et des pré-actionneurs , on prévoit directement des dispositifs de sécurité au niveau de la partie opérative. Ce sont souvent des dispositifs mécaniques qui empêchent des conséquences grave en cas de défaillance.



Comme les modes de marche ont pour objectif de préciser le cahier des charges comportemental de la partie commande, ils ne considèrent que la sécurité de niveau 2.

3. L'automate Programmable Industriel (API)



Programmer un automate, c'est indiquer à celui-ci le cahier des charges de la partie commande qu'il matérialise.

7.4.1 Les objets manipulés :

Le programme utilise des variables constituées de bits ou de mots. Un mot est un ensemble de bits permettant des stocker des valeurs numériques. En général un mot est constitué de 16 bits. Chaque automate est caractérisé par le nombre de mots et de bits utilisables et par leur mode d'adressage.

7.4.1.1 Les objets Bits :

Bits d'entrée/sortie : C'est l'adresse de la zone mémoire ou est stockée l'état d'une entrée ou d'une sortie lors des phases de lecture des entrées ou d'affectation des sorties du cycle de l'automate.

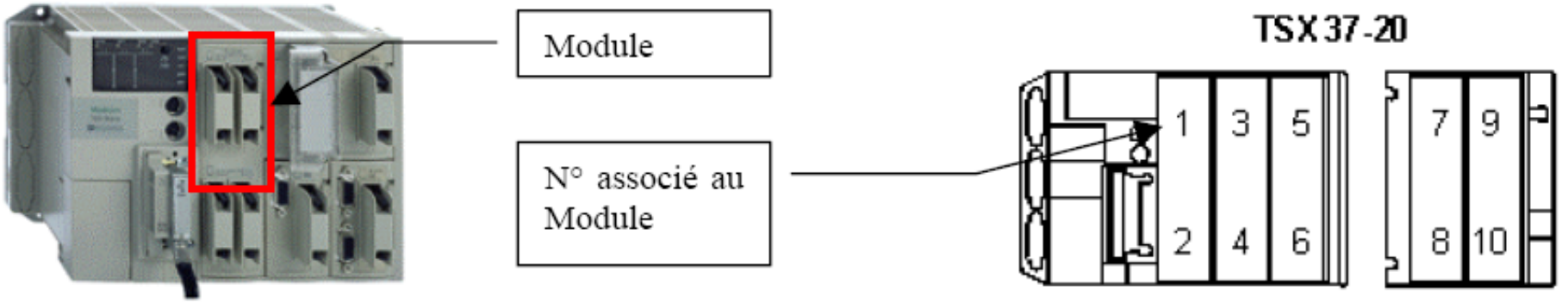
Bits d'information :Ce sont des adresses en mémoire qui permettent de stocker une information.

Exemple d'adressage du TSX 3720 :

TSX Micro

%	I ,Q,M ou K	X, W, D ou F	x.i	.r
Symbole	**Type d'objet** **I** = Entrée **Q** =Sortie **M** = information en lecture écriture **K**=Information de configuration	**Format** **X**=booléens **W**=mots **D**=double mots **F**=flottants	Position (x) et numéro de voie (i) du module TSX Micro	**Rang** r=0 à 127 ou ERR

Le TSX37 étant un automate modulaire, x désigne le numéro du module et i le numéro de l'entrée/sortie dans le module :



Ainsi, si le module 1 est un module d'entrée et le module 2 un module de sortie,
%I1.3 désigne la quatrième entrée du module 1 (On commence à 0…).
%Q2.5 désigne la sixième sortie du module 2.
Le TSX37 possède 256 bits internes d'information notés de **%M0** à **%M255**.
Le TSX37 possède 128 bits de N° d'étapes notés **%X0** à **%X127**.

7.4.2 Structure d'un programme automate.

Le programme automate est exécuté à chaque cycle de l'automate. Il est en général organisé en trois parties ou sections :

7.4.2.1 Le traitement préliminaire.

Cette partie du programme est traitée en premier. Elle permet en général la description des modes de marche et notamment du comportement de l'Arrêt d'Urgence.

7.4.2.2 Le traitement séquentiel.

C'est le corps du programme. En général programmé en langage GRAFCET, cette partie décrit la partie séquentielle du cahier des charges.

7.4.2.3 Le traitement postérieur.

Cette partie est exécutée après le traitement séquentiel. Elle permet le traitement particulier des actions du GRAFCET.

7.4.3 Langages de programmation des automates :

La norme CEI 1131 précise cinq langages possibles pour la programmation des automates :

- Instruction List (IL) *liste d'instructions*
- Ladder Diagram (LD) *langage à contacts*
- Sequential Function Chart (SFC) *diagramme fonctionnel en séquence*
- Structured Text (ST) *littéral structuré*
- Function Block Diagram (FBD) *langage en blocs fonctionnels*

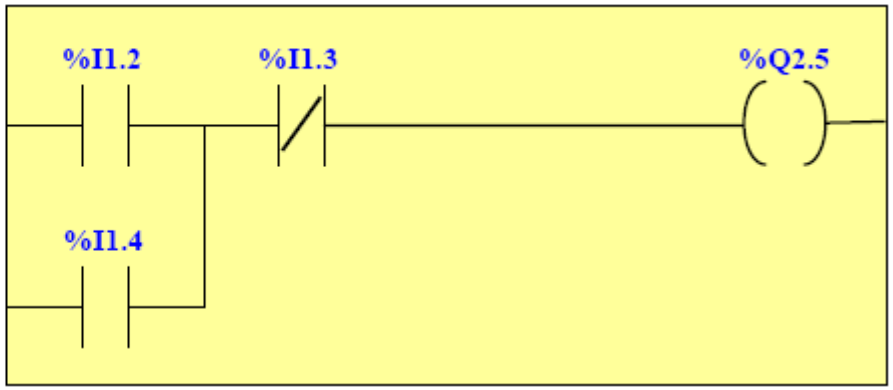
7.4.3.1 Le langage « Liste d'instruction » (IL) :

C'est le langage le plus répandu, notamment pour les automates « bas de gamme ».
Il s'agit d'un ensemble d'instruction simples permettant de manipuler des bits ou des mots.

```
%L1
        LD    %I1.2
        OR    %I1.4
        ST    %M0
        LDN   %I1.3
        AND   %M0
        ST    %Q2.5
```

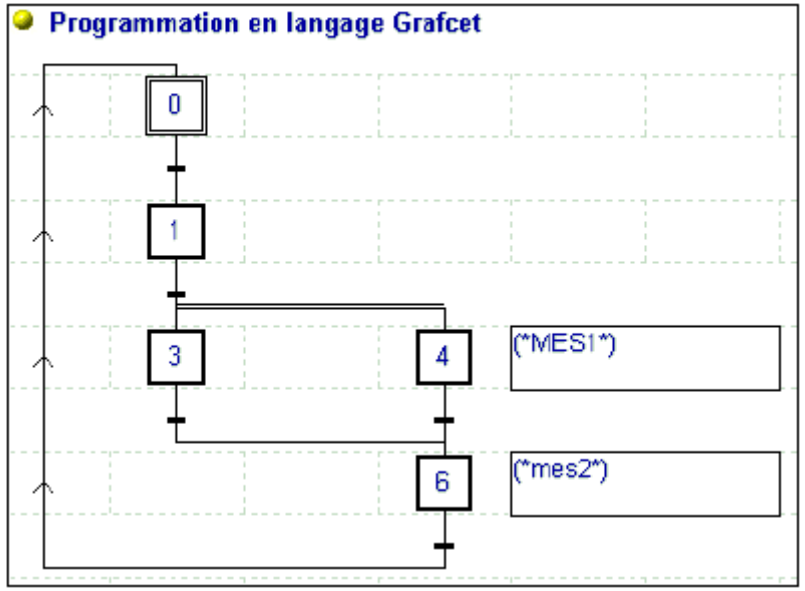
7.4.3.2 Le Langage “A contacts” (LD) :

Pendant longtemps, les systèmes automatisés ont été réalisé à partir de contacts électriques. Ce langage est dérivé de cette habitude. Il s’agit d’un langage graphique. On dessine un réseau de contacts électriques traduisant le cahier des charges du système.



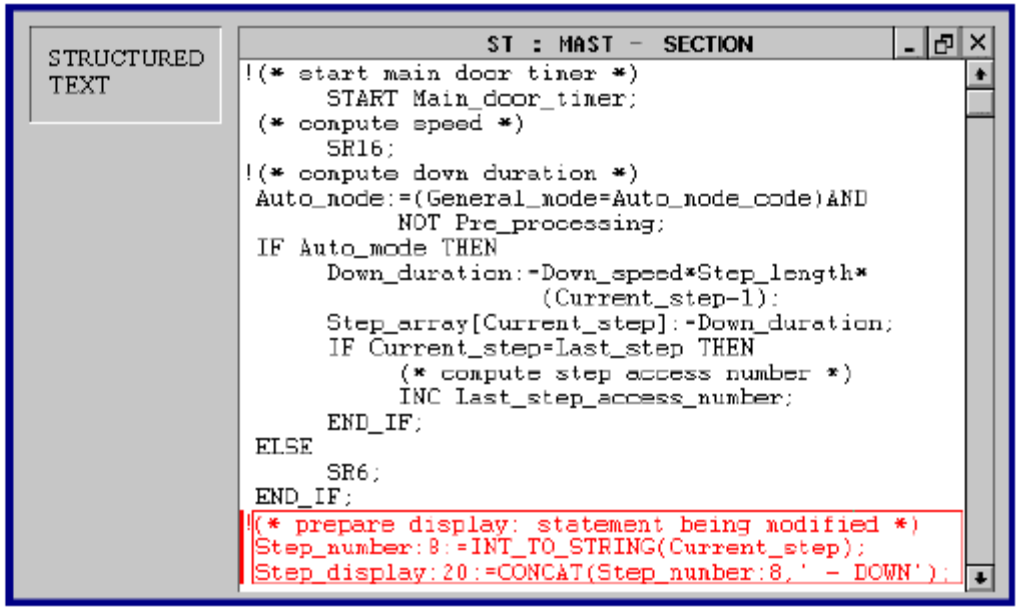
7.4.3.3 Le langage « Diagramme fonctionnel en séquence »(SFC) :

La version la plus répandue de ce type de langage est le GRAFCET. On programme l'automate en dessinant le GRAFCET traduisant le comportement des sorties en fonction des entrées.



7.4.3.4 Le langage « Littéral Structuré »(ST) :

Ce langage est proche d'un langage informatique classique (Pascal, Visual Basic,…). Certains cahiers des charges nécessitent une part importante de traitement de mots. Le langage ST est alors très bien adapté. Il permet la programmation de calculs complexes et d'algorithmes comportant des boucles de traitement.

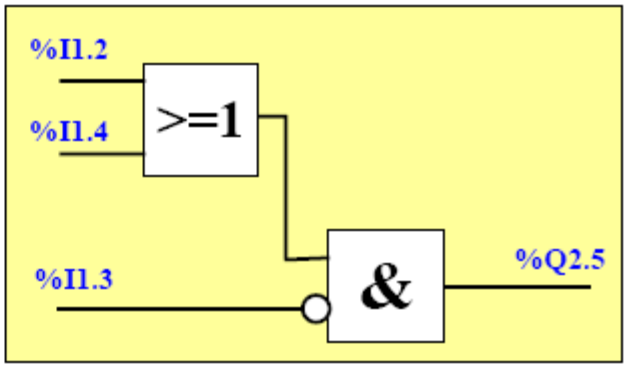


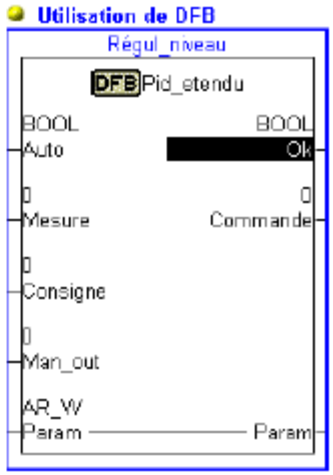
7.4.3.5 Le langage « en blocs fonctionnels »(FDB) :

Ce langage est très répandu aux Etats-Unis. C'est un langage graphique. La programmation se fait en plaçant des « boites » fonctionnelles et en les reliant entre elles.

Un grand intérêt de ce langage est la possibilité pour l'utilisateur de définir lui-même ses propre boites fonctionnelles.

Cela permet de structurer de façon efficace les programmes.





7.5 Interfaçage d'un automate :

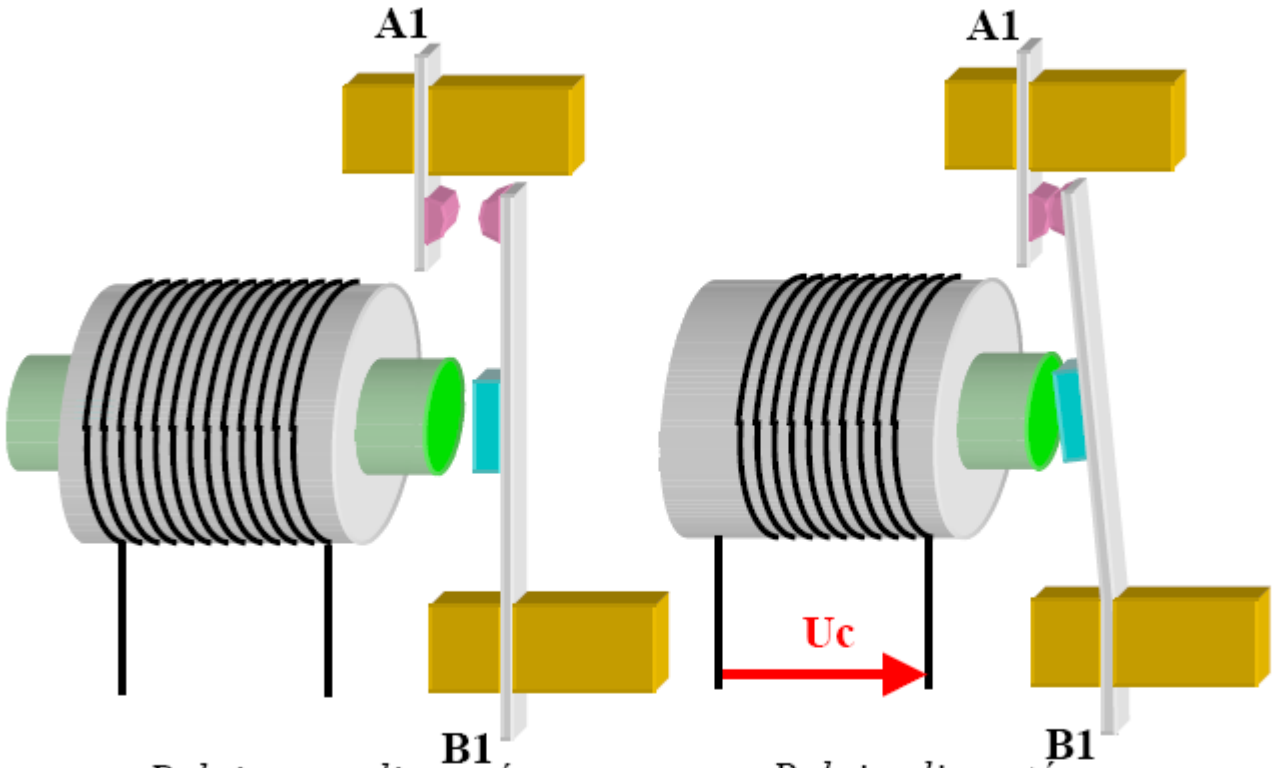
L'Interfaçage entre l'automate et la partie opérative à 3 fonctions qui sont :

- L'adaptation du type d'énergie (Ex : l'automate délivre une énergie électrique alors que le vérin à commander utilise une énergie pneumatique).
- L'adaptation du niveau d'énergie (L'automate délivre une énergie de quelque Watt alors que le moteur à commander utilise plusieurs milliers de Watt).
- Assurer la sécurité de niveau 1. (cf. chapitre 6.2.2.)

Il existe plusieurs type de sorties d'automate. Nous ne nous intéressons qu'aux automates à sortie à relais.

7.5.1 Notion de relais :

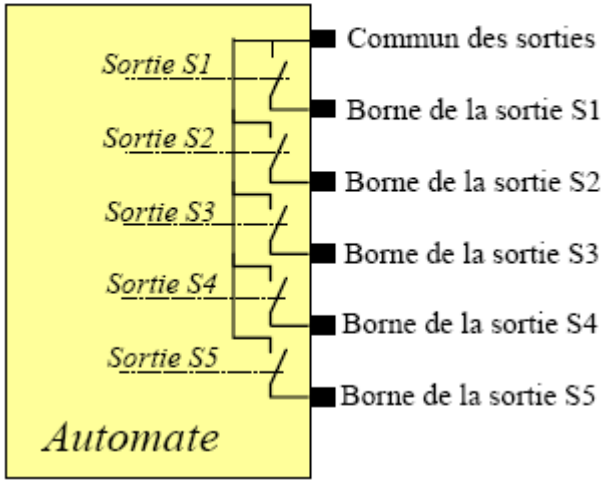
Un relais est un dispositif électromagnétique permettant de fermer un contact électrique. Le



Relais non alimenté *Relais alimenté*

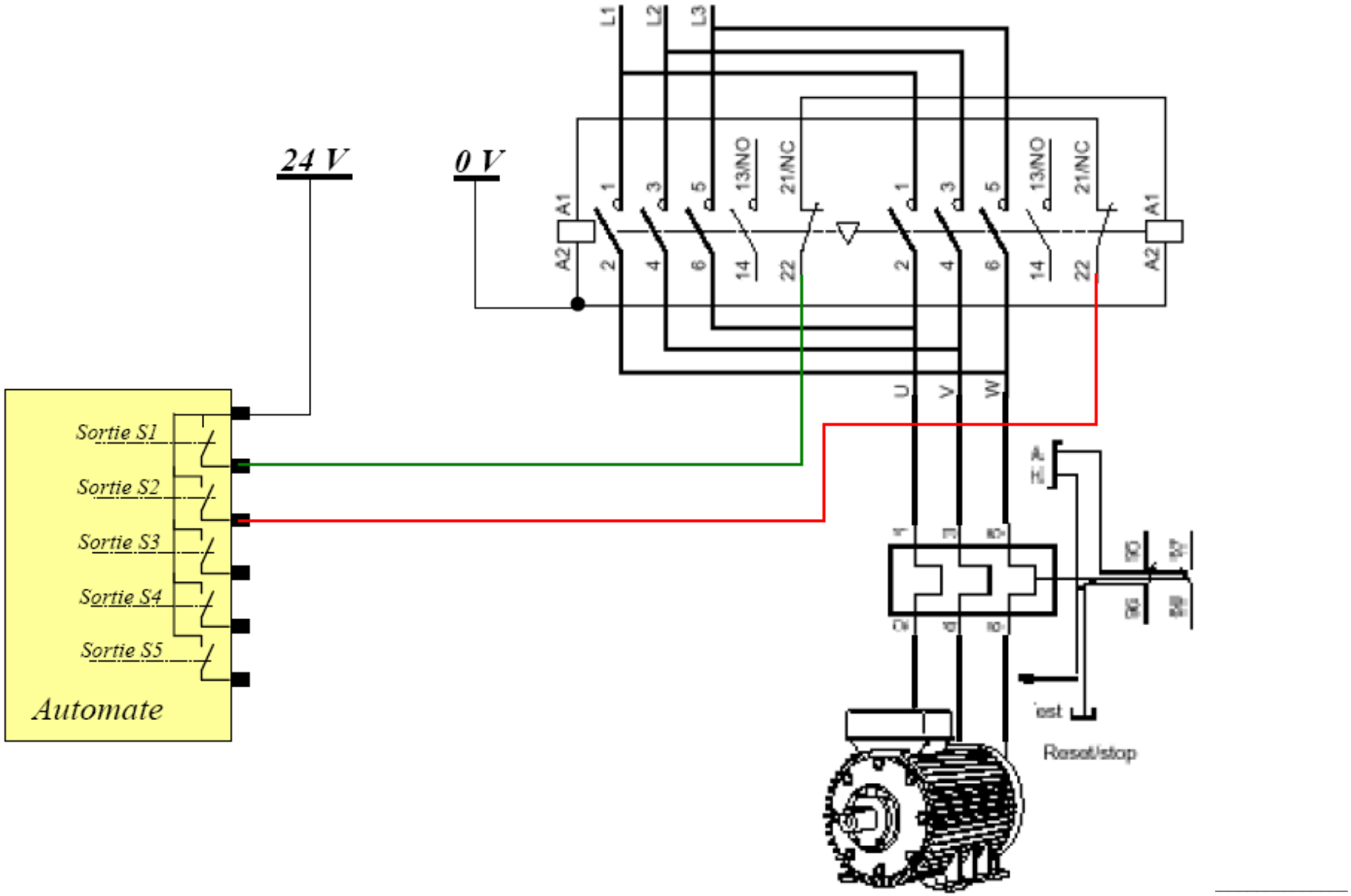
relais comporte un électro-aimant qui, lorsqu'il est alimenter, ferme (ou ouvre) un contact électrique.

Lorsque un automate possède des sorties à relais, cela signifie qu'il possède un relais par sortie. Lorsque la sortie « passe à 1 », l'automate alimente le relais correspondant qui relie deux bornes électriques accessibles à l'extérieur de l'automate.

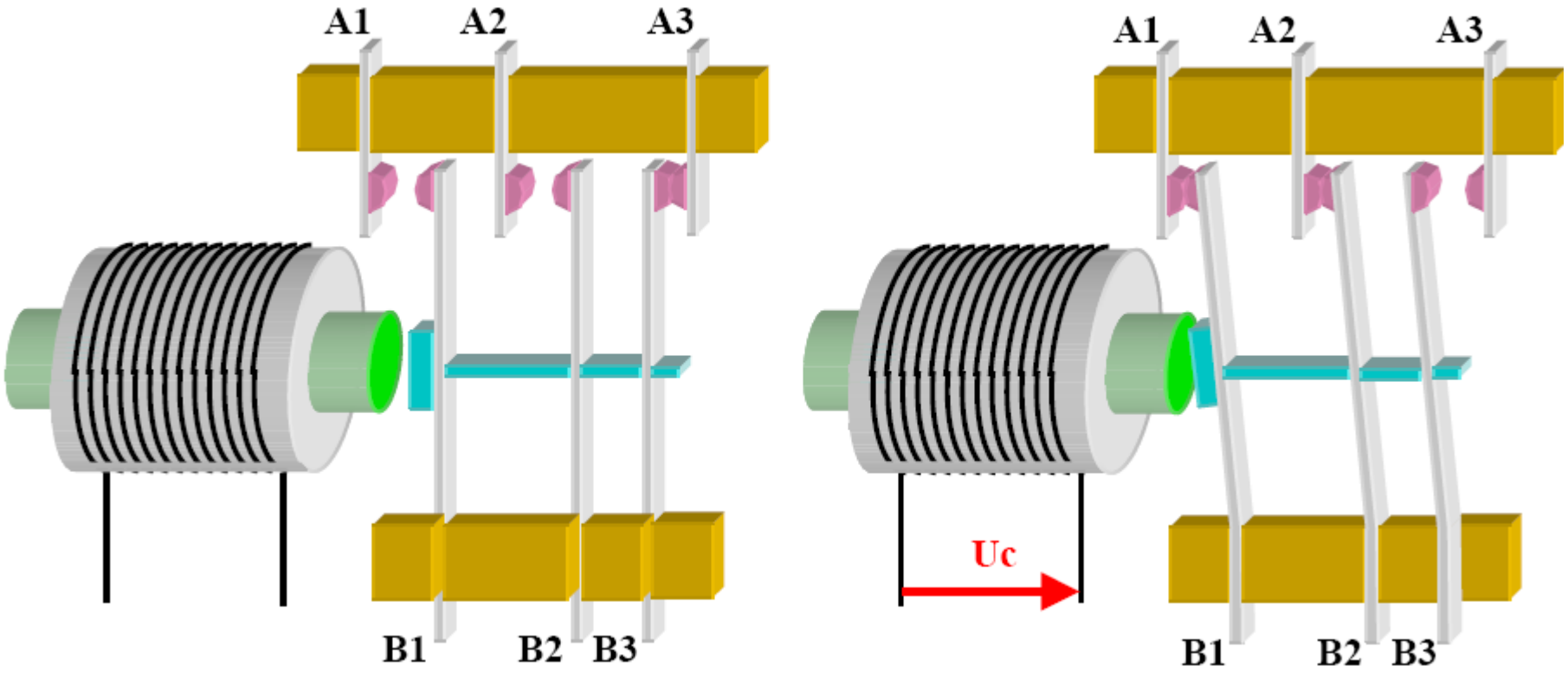


En général, plusieurs sorties possèdent une borne commune appelée « commun des sorties ». Ainsi, sur le schéma ci-contre, lorsque la sortie S_i passe à « 1 », la borne de la sortie S_i se trouve reliée à la borne « Commun des sorties ».

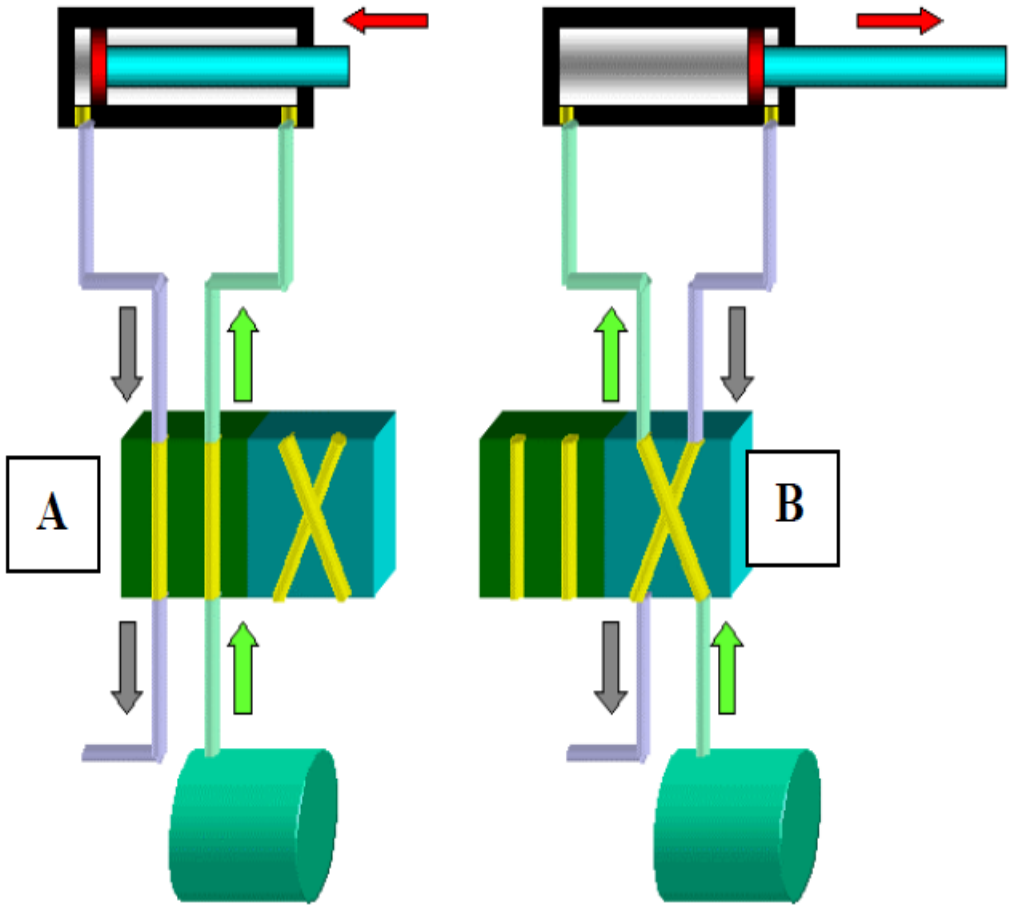
Voir exemple ci-dessous de la commande du sens de rotation d'un moteur asynchrone triphasé


24 V
0 V
L1
L2
L3
A1
A2
13/NO
21/NC
14
22
U
V
W
Sortie S1
Sortie S2
Sortie S3
Sortie S4
Sortie S5
Automate
Reset/stop

Exemple d'un relai :



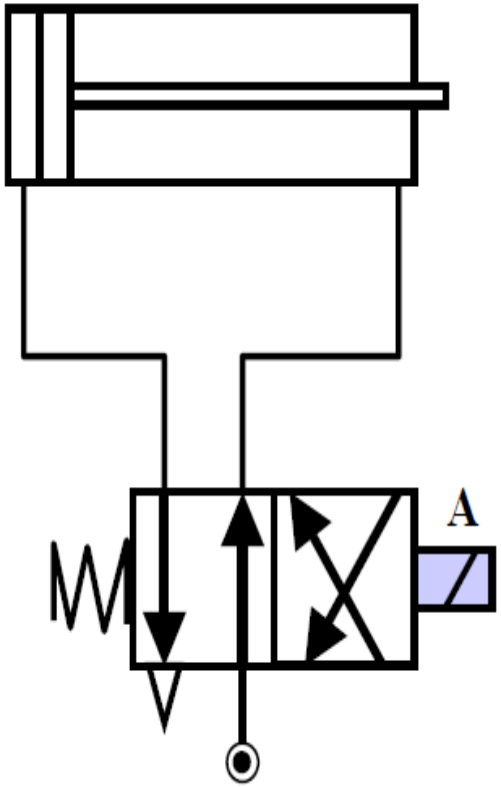
7.5.3 Commande de vérins pneumatiques :

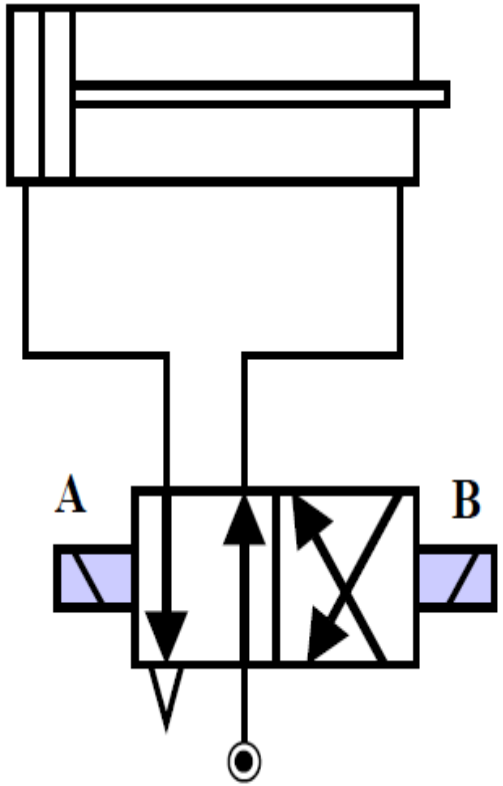


La commande d'un vérin pneumatique se fait au moyen d'un distributeur pneumatique. Le rôle de ce distributeur est de relier les chambres du vérin, tantôt à une source d'air comprimé, tantôt à un échappement.

Dans l'exemple de distributeur ci-contre, on trouve un tiroir à deux positions. Dans la position A la chambre avant du vérin est reliée à la source de pression et la chambre arrière à l'échappement. Si le tiroir se translate vers la gauche, on obtient la position B qui fait que la chambre avant est reliée

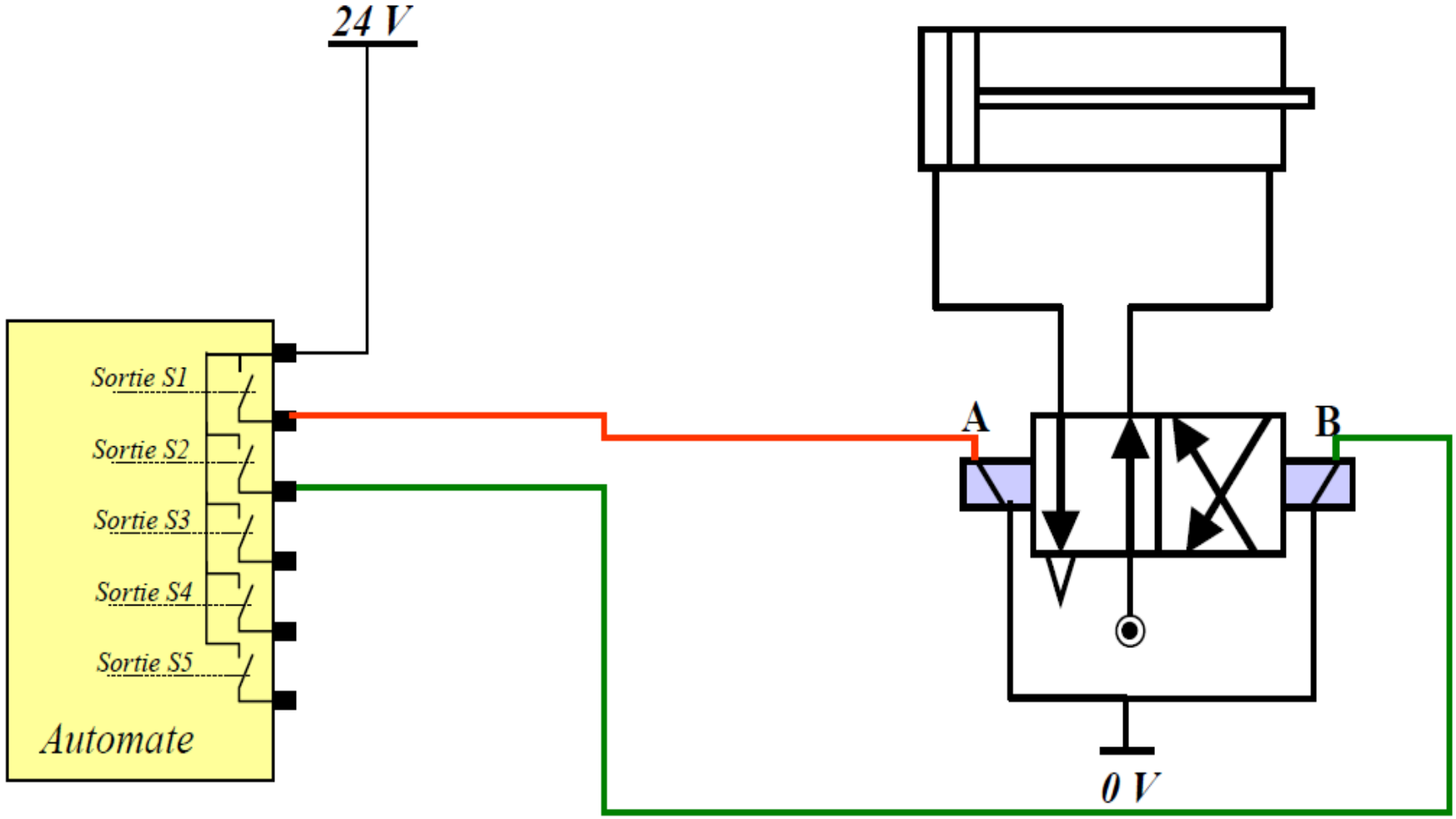
à l'échappement et la chambre arrière à la source de pression. La translation du tiroir peut se faire au moyen de deux électro-aimants : on a alors un distributeur à commande électrique à deux positions stables. L'alimentation de l'électro-aimant A commande la rentrée de la tige du vérin et l'alimentation de l'électro-aimant B commande la sortie de la tige du vérin.





Dans l'exemple de gauche, le distributeur possède un seul électro-aimant. Si celui-ci est alimenté la tige du vérin sort, sinon elle rentre car le tiroir revient en position grâce à un ressort.

Dans l'exemple de cablage donné ci-dessous, la sortie S1 de l'automate commande la rentrée de la tige du vérin et la sortie S2 commande la sortie de la tige.



Annexes : Les normes CEI 60848 et CEI 61131

La norme internationale CEI 60848 est destinée principalement aux spécificateurs, concepteurs, réalisateurs, agents d'exploitation et de maintenance qui ont besoin de spécifier le comportement d'un système (commande d'une machine automatique, composant de sûreté, etc). Ce langage de spécification est un support de communication privilégié entre les concepteurs et les utilisateurs de systèmes automatisés.

Le langage de spécification GRAFCET permet la description fonctionnelle du comportement de la partie séquentielle des systèmes de commande.

La norme définit les symboles et les règles nécessaires à la représentation graphique de ce langage, ainsi que l'interprétation qui en est faite. Elle ne donne pas de méthodes d'utilisation des notions et concepts définis.

Elle a été établie pour les systèmes automatiques qui relèvent d'applications industrielles ou grand public, mais aucun champ d'application n'est exclu.

Les méthodes permettant le passage d'une spécification utilisant le GRAFCET à une réalisation câblée et (ou) programmée ne font pas partie du domaine d'application de cette norme.

Dans le cas des systèmes de commande intégrant un automate programmable, il est important de remarquer que depuis 1993, la norme CEI 61131-3 [2] définissant un ensemble de langages de programmation destinés aux automates programmables peut être utilisée. Il est possible alors de faire appel au langage « SFC » décrit dans cette norme comme méthode de réalisation de la spécification GRAFCET.

Norme CEI 61131-3

Les langages de programmation des systèmes automatisés sont décrits dans deux normes complémentaires : les normes CEI 61131-3 et CEI 61499.

La norme CEI 61131-3 définit entre autres choses, cinq langages qui peuvent être utilisés pour la programmation d'applications d'automates programmables industriels (API). Les cinq langages sont :

- **SFC** (« Sequential Function Chart ») : issu du langage GRAFCET, ce langage, de haut niveau, permet la programmation aisée de tous les procédés séquentiels ;
- **FBD** (« Function Block Diagram », ou schéma par blocs) : ce langage permet de programmer graphiquement à l'aide de blocs, représentant des variables, des opérateurs ou des fonctions. Il permet de manipuler tous les types de variables ;
- **LD** (« Ladder Diagram », ou schéma à relais) : ce langage graphique est essentiellement dédié à la programmation d'équations booléennes (true/false) ;
- **ST** (« Structured Text » ou texte structuré) : ce langage est un langage textuel de haut niveau. Il permet la programmation de tout type d'algorithme plus ou moins complexe ;
- **IL** (« Instruction List », ou liste d'instructions) : ce langage textuel de bas niveau est un langage à une instruction par ligne. Il peut être comparé au langage assembleur.

Référence :
https://www.techniques-ingenieur.fr/base-documentaire/automatique-robotique-th16/supervision-des-systemes-industriels-42396210/langages-de-programmation-pour-systemes-automatises-norme-cei-61131-3-s8030/

http://lgt.garnier.free.fr/espace_ee_fichiers/automatisme/grafcet#S%E9quences+parall%E8les