SYSTEMES LOGIQUES et GRAFCET

Prof. Belkacem OULD BOUAMAMA

Responsable de l'équipe MOCIS
Méthodes et Outils pour la conception Intégrée des Systèmes
http://www.mocis-lagis.fr/membres/belkacem-ould-bouamama/
Laboratoire d'Automatique, Génie Informatique et Signal
(LAGIS - UMR CNRS 8219

et Directeur de la recherche à École Polytechnique de Lille (Poltech' lille)

---

mèl : Belkacem.ouldbouamama@polytech-lille.fr, Tel: (33) (0) 3 28 76 73 87 , mobile : (33) (0) 6 67 12 30 20

- *Ce cours est dispensé aux élèves de niveau IUT*
- *Toutes vos remarques pour l'amélioration de ce cours sont les bienvenues.*

1. NOTION DE SYSTEMES LOGIQUES

1.1. Définition d'une variable logique :

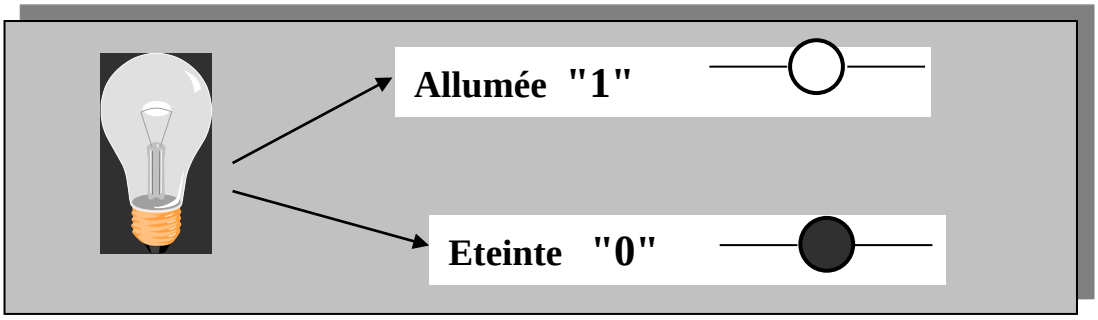
Une variable logique est définie par deux états logiques.

1. "**0**" **non actif , faux** (0 volt)

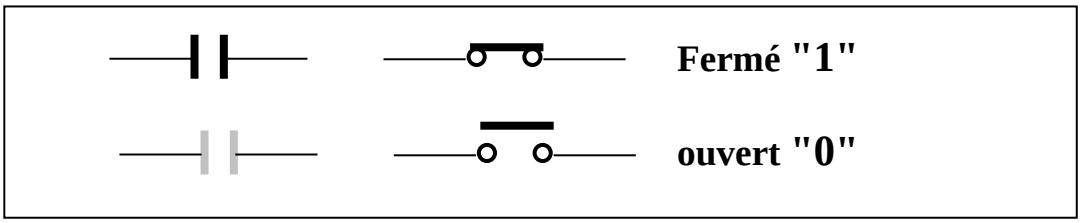
2. "**1**" **actif, vrai** (5 volts)

Exemples :

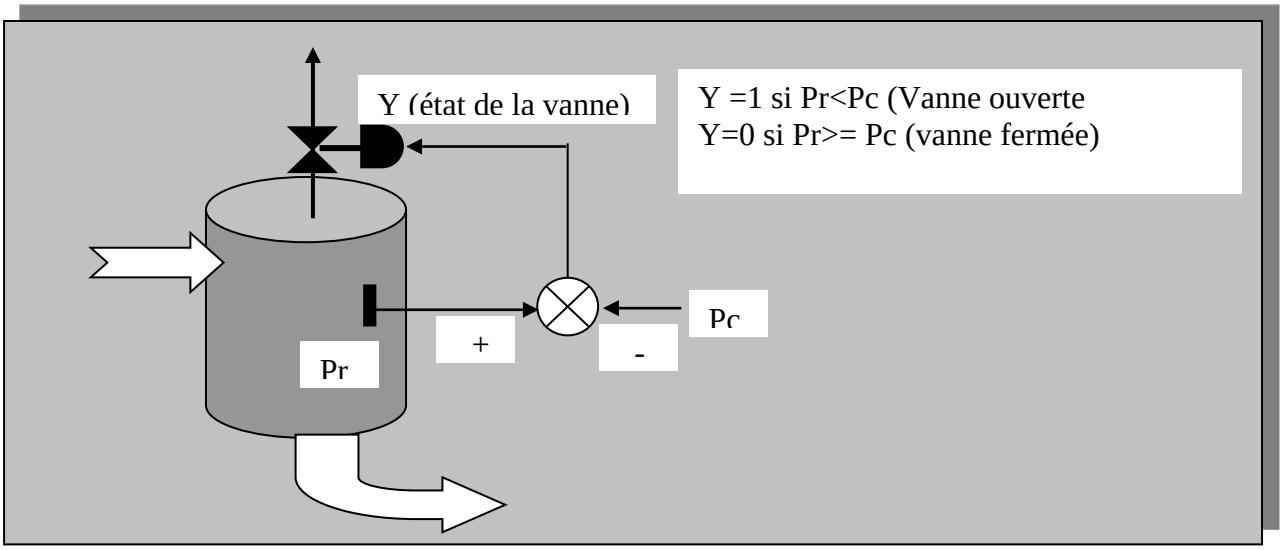
- **Etat d'une lampe**



- **Contacts**

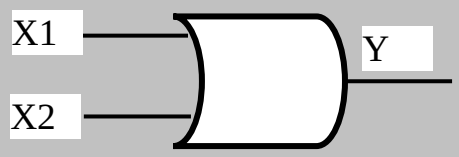


- Sécurité dans un réacteur (Soupape de sécurité)



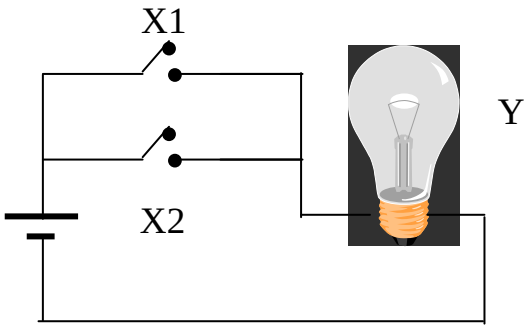
1.2 OPERATIONS LOGIQUES

♦OU (OR) logique

SYMBOLE	EQUATION	TABLE DE VERITE
	**Y= X1+X2**	

X1	X2	Y
1	0	1
1	1	1
0	0	0
0	1	0

♦ Cablage

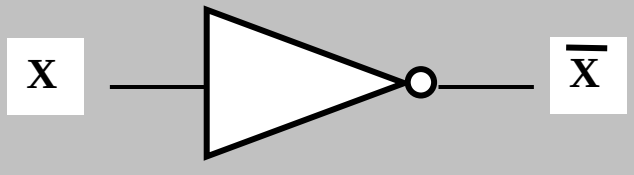


♦ COMPLEMENT OU INVERSE NON (NOT) D'UNE VARIABLE LOGIQUE

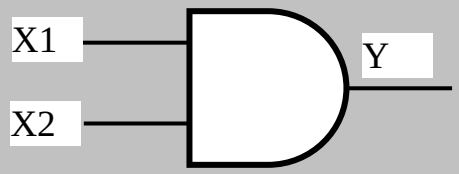
Le Complément de **X** est noté $\overline{X}$

Table de vérité	
X	$\overline{X}$
1	0
0	1

Symbole

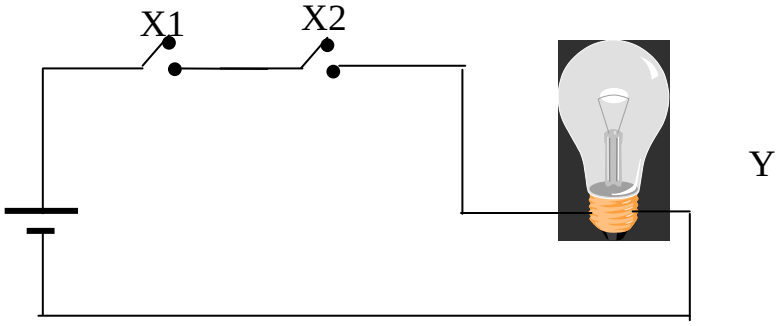


♦ ET (AND) logique

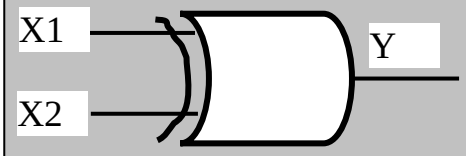
SYMBOLE	EQUATION	TABLE DE VERITE
	**Y= X1.X2**	

X1	X2	Y
1	0	0
1	1	1
0	0	0
0	1	0

♦ Câblage

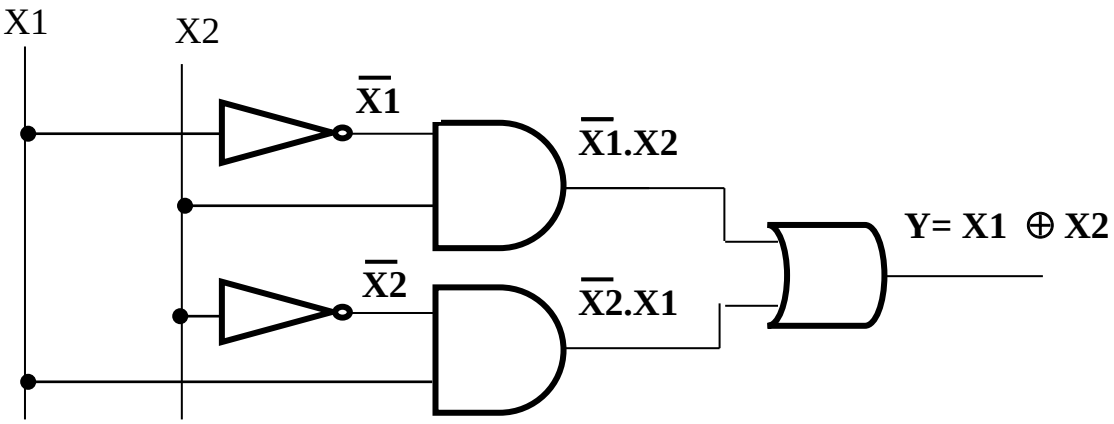


♦ OU Exclusif (XOR) logique

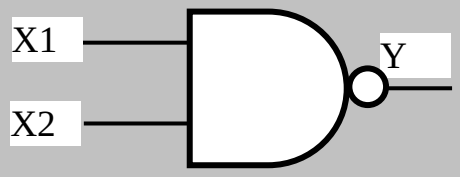
SYMBOLE	EQUATION	TABLE DE VERITE
	**Y= X1 ⊕ X2** **=$\overline{X1}$.X2+X1.$\overline{X2}$**	

X1	X2	Y
1	0	1
1	1	0
0	0	0
0	1	1

♦ Câblage



♦ NON ET (NAND)

SYMBOLE	EQUATION	TABLE DE VERITE
	$Y = \overline{X1.X2} = \overline{X1} + \overline{X2}$	

X1	X2	Y
1	0	1
1	1	0
0	0	1
0	1	1

1.3 SIMPLIFICATION DE FONCTIONS LOGIQUES

$X+0=X$	$X+X=X$	$X.0=0$	$X.X=X$
$X+0=1$	$X+\overline{X}=1$	$X.1=X$	$X.\overline{X}=0$

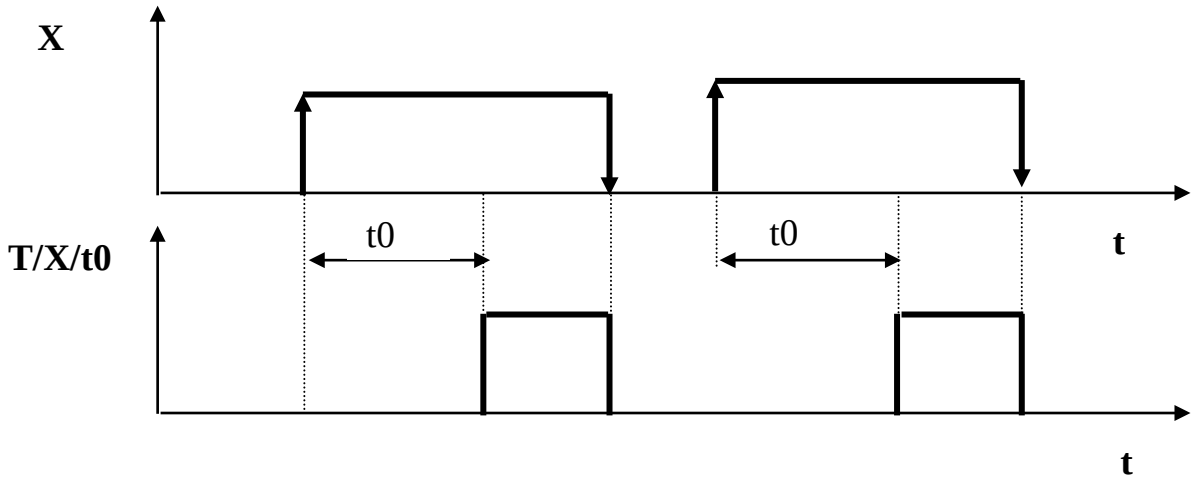
1.4 FRONTS

Front montant (FM), noté ↑**m**, est l'instant où **m** passe de **0** à **1**

Front descendant (FDM), noté ↓**m**, est l'instant où **m** passe de **1** à **0**

Exemple : Bouton poussoir marche ou arrêt

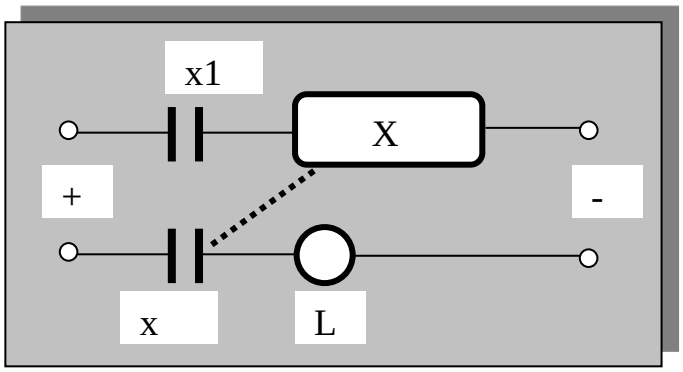
1.5 TEMPORISATION



∗1.6 FONCTIONS PECIALES

♦ Relais

Un relais est constitué d'une bobine X qui lorsqu'elle est traversée par un courant, modifie l'état des contacts x qui lui sont associées.



♦ Fonctions à mémoire

Les fonctions à mémoire conservent l'information selon laquelle un contact a été à l'état actif même si celui ci est repassé entre temps à l'état non actif.

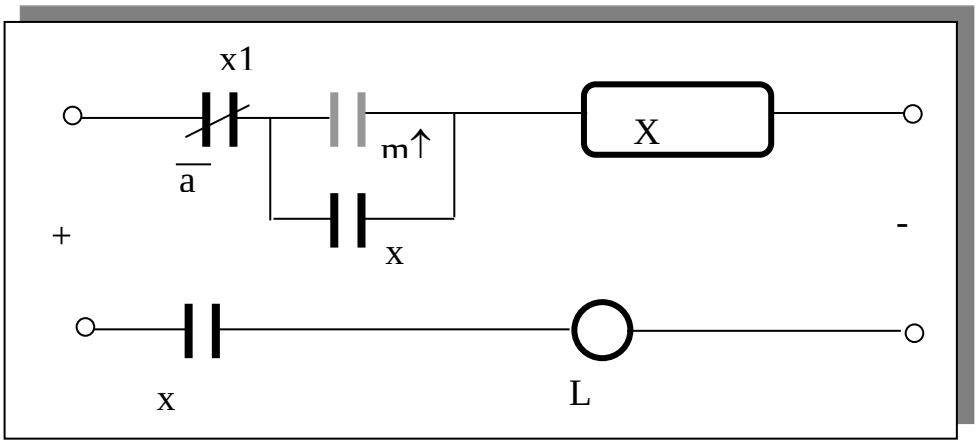
Il faut prévoir une fonction de remise à l'état non actif de la mémoire

Exemple : Marche arrêt prioritaire

On veut réaliser une machine qui se mette en route suite à une impulsion sur un contact. Un autre contact permet de l'arrêter. Si les deux contacts sont actifs en même temps la machine ne démarre pas.

	t1	t2	t3	t4	t5	t6	t7
m	0	0	0	0	0	1	0
a	0	0	0	1	0	1	0
L	0	1	1	0	0	0	0

Cablage



Equation logique :

$$x = \overline{a}.(m+x) = \overline{a}\ \text{ET}.(m\ \text{OU}\ x)$$
$$L = x$$

1.7 NOTION DE SYSTEMES DE NUMERATION

Tout nombre **N** de base **b** est décomposable en fonction des puissances entières de **b**

$$N \equiv \sum_{i=0}^{i=n} a_i . b^i$$

$a_i \in 0,1;2,\ldots,(b-1)$

i sont des entiers ≥ 0

n est l'exposant de b du chiffre de poids fort

Exemple.

$(54321)_6 \equiv 5*6^4 + 4*6^3 + 3*6^2 + 2*6^1 + 1*6^0$

$(27674)_{10} \equiv 2*10^4 + 7*10^3 + 7*10^3 + 6*10^2 + 7*10^1 + 4*10^0$

♦ SYSTEME BINAIRE

si b=2, le système est appelé binaire

$$N \equiv \sum_{i=0}^{i=n} a_i . 2^i$$

$a_i \in \{0,1\}$

On aura par exemple

$$(101101)_2 \equiv 1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$$
$$\equiv 32 + 8 + 4 + 1$$
$$\equiv (45)_{10}$$

♦ Présentation d'un nombre de base > à 10

1. Base duodécimal (base 12)

$a_i = (0,1,2,3,4,5,6,7,8,9,A,B)$

$(9A73B)_{12} \equiv 9*12^4 + A*12^3 + 7*12^2 + 3*12^1 + B*12^0$

2. Base Hexadécimal (base 16)

$a_i = (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)$

$(A73)_{16} \equiv A*16^3 + 7*16^2 + 3*16^1 + 2*16^0$

$\equiv 10*4096 + 7*256 + 3*16 + 2$

$\equiv (42802)_{10}$

♦ Nombre fractionnaire

$$N \equiv \sum_{i=1}^{i=n} a_i . b^{-i}$$

$a_i \in 0,1;2,\ldots,(b-1)$

i sont des entiers > 0

-n est l'exposant de b du chiffre de poids faible

* 1.8 PRINCIPAUX CODES

1. Code binaire

Le plus utilisé (0,1)

Quelques opérations en binaire		
Addition	**Soustraction**	**Multiplication**
0+0=0 1+0=1 0+1=1 1+1=0 (report de 1)	0-0=0 1-0=1 0-1=1 (retenue de 1) 1-1=0	0x0=0 0x1=0 1x0=0 1x1=1

Exemple

```
      1 0 1 1 0 1
    x       1 0 1
  ---------------
      1 0 1 1 0 1
  1 0 1 1 0 1      <-- décalage dû au zéro de 101
  ---------------
  1 1 1 0 0 0 0 1
```

2. Code GRAY (code binaire réfléchi)

Les grandeurs successives ne différent que d'un caractère.

Equivalent du code Gray:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000

3. Code BCD (Binary Coded decimal)

Ce code conserve les avantages d'un système décimal et du code binaire. Il est utilisé par les calculateurs.

Code décimal	0	1	2	3	4	5	6	7	8	9
Code DCB	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Exemple Pour coder 86387

$\underset{8}{1000}\ \underset{6}{0110}\ \underset{3}{0011}\ \underset{8}{1000}\ \underset{7}{0111}$

U est $(55)_{16}$ soit 1010101 en binaire

F est $(46)_{16}$ soit 1000110

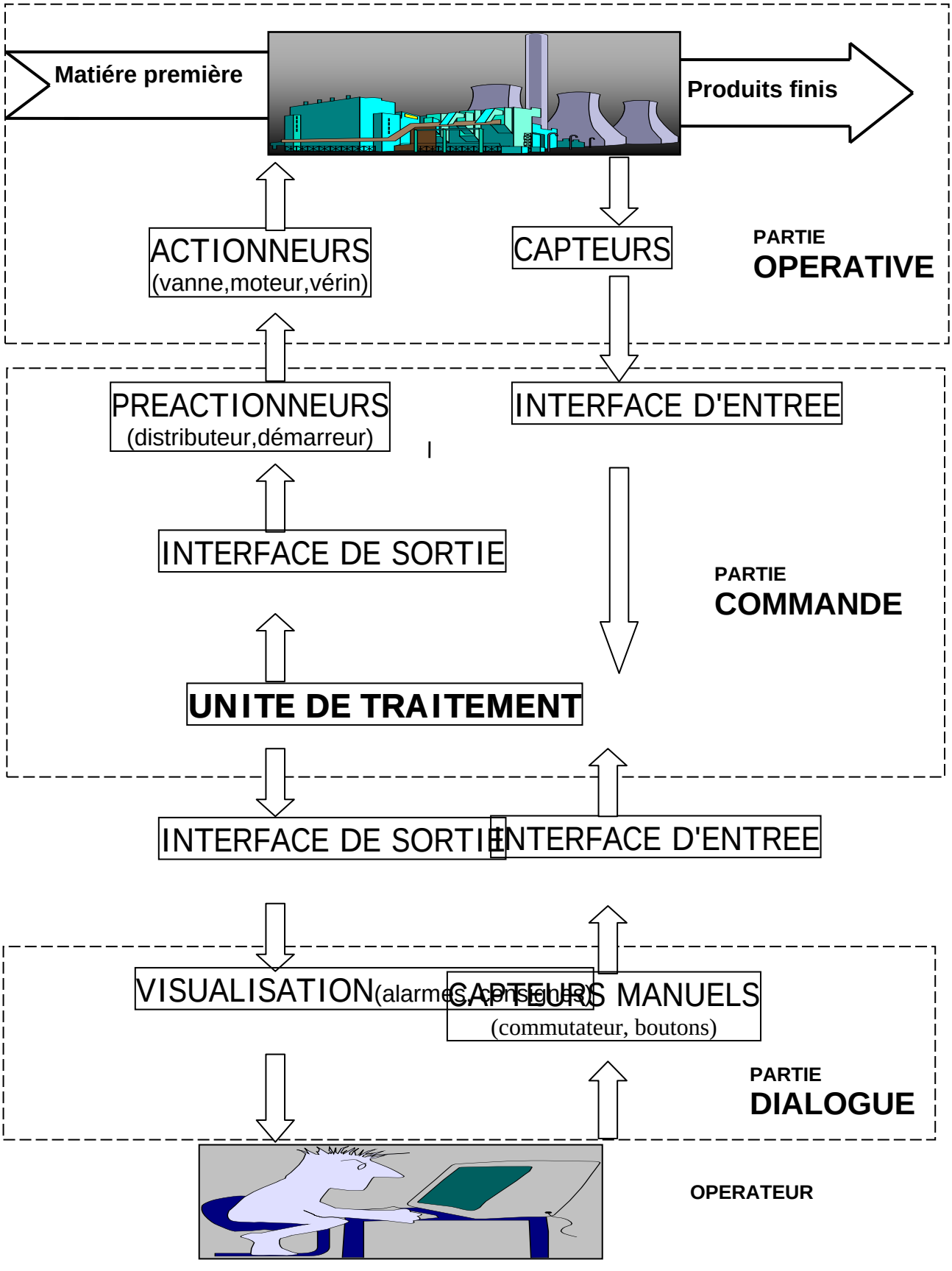
K est $(4B)_{16}$ soit 1001011

♦ **Code ASCII (American Standard Code for Information Interchange)**

Ce code est une norme universelle dans les transmissions. Il comprend 7 ou 8 caractères. le huitième caractère est dit de **Parité** sert à détecter les erreurs de transmissions.

SYSTEME AUTOMATISE

♦ 2.1 Architecture d'un système automatisé



2.2 Le cahier des charges

♦ Définition :

Le cahier des charges est un document ou sont spécifiées toutes les fonctions, toutes las valeurs des grandeurs physiques et tous les modes d'utilisation du matériel.

♦ Niveaux de description du matériel

1. Spécifications fonctionnelles

correspond aux spécifications fonctionnelles qui décrivent l'automatisme indépendamment de la technique utilisée (pneumatique, hydraulique, électrique). Ces spécifications peuvent permettre au concepteur d'en comprendre le rôle, de définir les actions à réaliser et leur enchaînement séquentiel.

2. Spécifications techniques ou opérationnelles.

Les spécifications techniques énumèrent les caractéristiques physiques des capteurs et des actionneurs, les conditions d'environnement et les conditions pour assurer la sécurité de fonctionnement de l'automatisme. Les spécifications opérationnelles assurent l'optimisation de l'exploitation du process (modes de marche et d'arrêt, maintenabilité, absence de pannes dangereuses).

3. Documentation à propos de l'utilisation

il concerne (câblage, programmation), de l'entretien et du dépannage du matériel. cette documentation doit être à jour.

4. Niveaux supérieurs

Ils concernent les clauses juridiques, le service après-vente, les garanties, les conditions financières...

2.3 Représentation graphique du fonctionnement d'un automatisme

♦ **L'organigramme** :

Elaboré en 1948 il est appliqué essentiellement à l'informatique (logiciels) puis à l'électronique (architecture des ordinateurs et API numériques).

♦ **Les langages à contact** : ladder, relais.

♦ **Les langages booléens** : logigrammes.

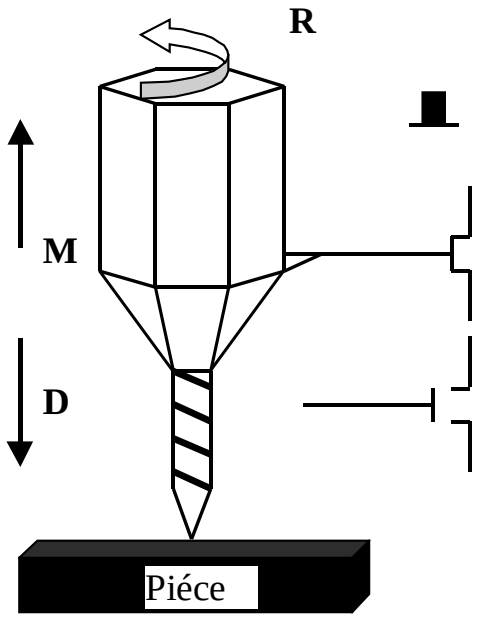
♦ **Le Grafcet** :

(**G**raphe de **C**ommande **É**tapes - **T**ransitions), élaboré en 1977 par l'AFCET, il est appliqué exclusivement aux automates.

♦ **Les réseaux de Pétri :**

Elaborés en 1970, ils sont particulièrement bien adaptés à l'étude des fonctionnements simultanés, ou pour des circuits complexes.

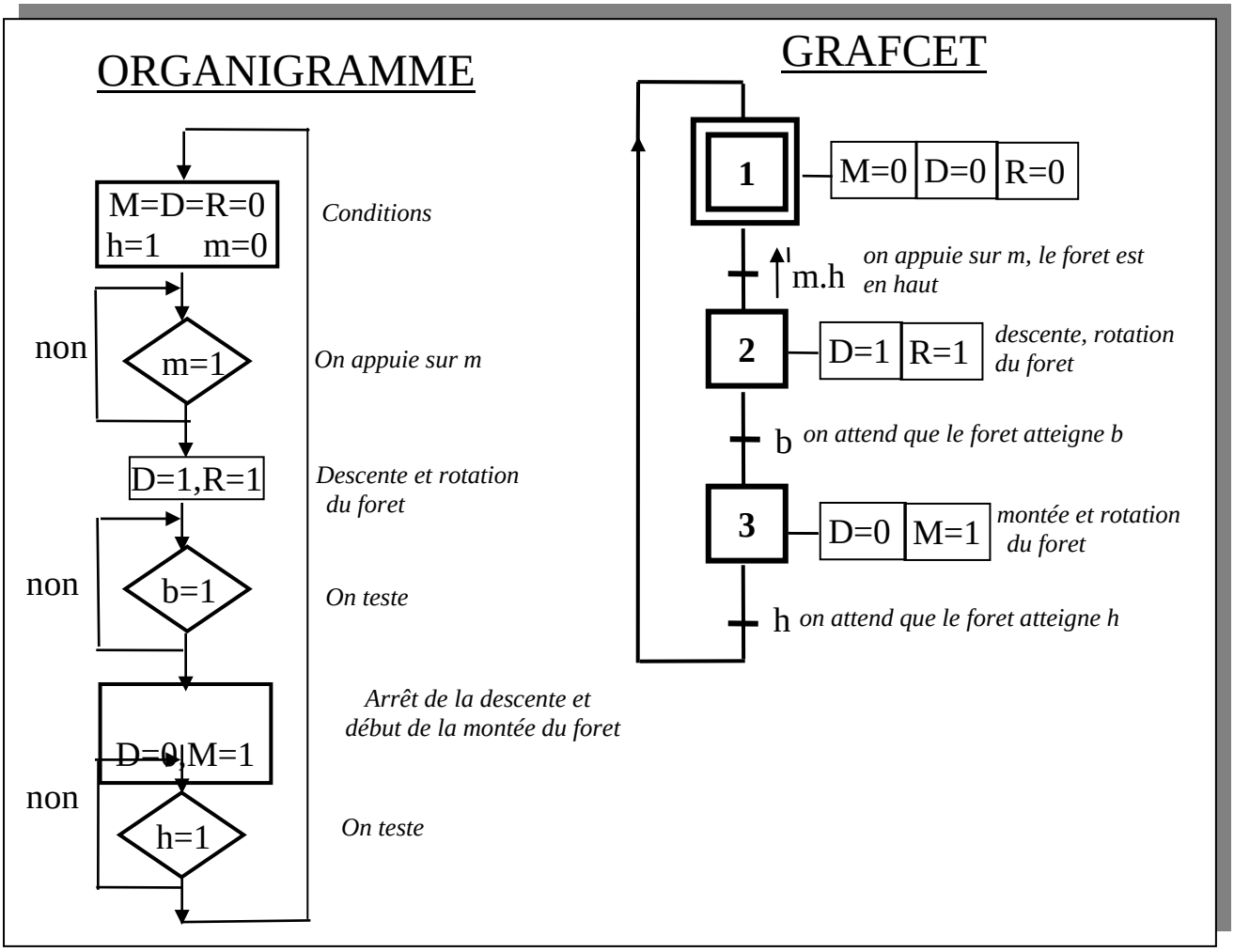
2.4 Exemple : automatisation d'une perceuse



Cahier des charges :

1. Le foret tourne toujours sur lui-même en descente et en montée.
2. Les contacts sont à 1 lorsqu'ils sont enclenchés.
3. Les contacts h et b sont enclenchés par le passage de la came solidaire de la perceuse.
4. Le cycle démarre lorsque l'on appuie sur le bouton poussoir m et que le contact h est enclenché par la came.

2.5 Grafcet et organigramme



♦ REMARQUES :

1. **L'organigramme** commence par les conditions initiales et se poursuit par des tests et des actions jusqu'au moment où le système revient à l'état initial.

2. Le **Grafcet** est constitué des éléments suivants: les étapes, les transitions, les actions et les réceptivités.

3. Dans un **Grafcet** il y a toujours l'alternance étape - transition - étape - transition... alors que dans un organigramme il peut y avoir plusieurs tests successifs. Ainsi l'organigramme repose sur cette notion de test à partir duquel on décide tel ou tel enchaînement vers une action ou vers un autre test. Le Grafcet s'appuie sur l'alternance étape - transition avec tel ou tel enchaînement selon la valeur de la réceptivité associée à une transition.

2.6 LE GRAFCET C'EST QUOI?

♦ **DEFINITION**

Le Grafcet est un outil graphique de définition pour l'automatisme séquentiel, en tout ou rien/. C'est un langage universel; qui peut se câbler par séquenceur, être programmé sur automate ou sur ordinateur.

♦ **COMMENT EST NE LE GRAFCET**

en 1977 du travail d'un groupe de l'**AFCET** (**A**ssociation **F**rancaise pour la **C**ybernétique **E**conomique et **T**echnique

♦ **SIGNIFICATION :**

Au choix : **GR**aphe de l'**AFCET**

GRAphe **F**onctionnel de **C**ommande **E**tape **T**ransition

♦ **ACTUELLEMENT**

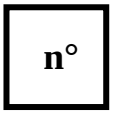
- Normalisé depuis **1982** en France et **1988** en tant que norme internationale par l'**ADEPA** (Agence Nationale pour le Développement de la Production Automatisé)

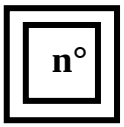
- Est enseigné du CAP au diplôme d'Ingénieur

Chapitre 3: LES ÉLÉMENTS DU GRAFCET

Un Grafcet est composé d'**ETAPES**, de **TRANSITIONS** et de **LIAISONS**

3.1 Normalisation

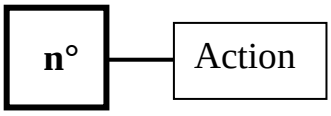
1. Les **étapes** sont représentées par des carrés. 

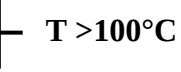
2. Les **étapes initiales** sont représentées par des carrés doubles 

3. Les **liaisons** orientées de haut en bas ne sont pas fléchées

4. Les **liaisons** orientées de bas en haut sont fléchées 

5. Les **transitions** sont représentées par des segments orthogonaux aux liaisons orientées 

6. Les **actions** s'écrivent à droite des étapes 

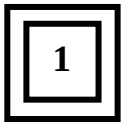
7. Les **réceptivités** s'écrivent à droite des transitions**.** 

3.2 les Eléments de base du Grafcet

♦ LES ETAPES :

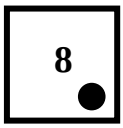
A un instant donné une étape peut être soit **active**, soit **inactive**. La situation d'un automatisme est défini par l'ensemble de toutes les étapes actives. Lors du déroulement de l'automatisme, les étapes sont actives les unes après les autres. A toute étape i, on associe une variable logique notée X_i telle que $X_i=1$ si l'étape est active et $X_i=0$ si l'étape est inactive.

Exemple :



Etape initiale N° 1 active

Etape N° 7 inactive

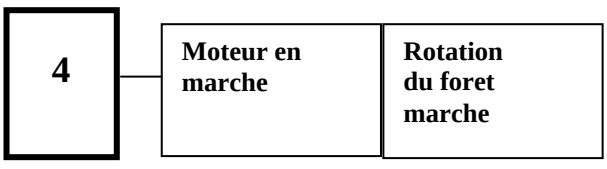


Etape N° 8 active

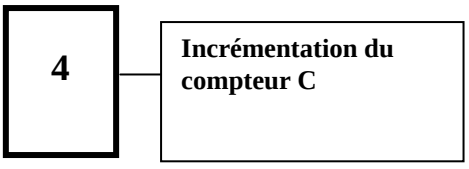
♦ LES ACTIONS :

A chaque étape peut être associé une ou plusieurs actions. Ces actions sont réalisées à chaque fois que l'on active l'étape à laquelle elles sont associées. Ces actions peuvent être **externes** (sortie de automatisme pour commander le procédé) ou **internes** (temporisation, comptage, calcul). Une étape peut n'avoir aucune action (attente d'un événement externe ou de la fin d'une temporisation.

- **Exemple :**



(Action externe)



(Action interne)

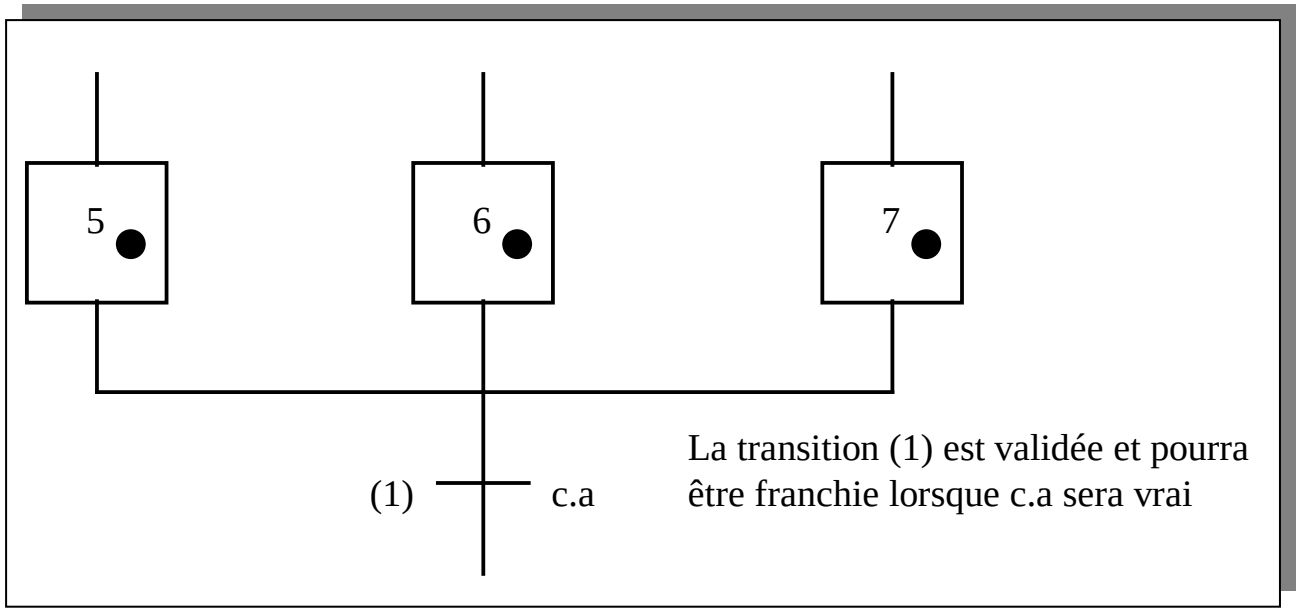
♦ LES TRANSITIONS

Elles expriment les possibilités d'évolution entre une ou plusieurs étapes.

Une transition peut être **validée** lorsque toutes les étapes immédiatement reliées à cette transition sont actives ou **non validée** dans le cas contraire.

Elle peut être **franchie** lorsqu'elle est validée et que la condition logique associée à cette transition est vraie.

- **Exemple** :

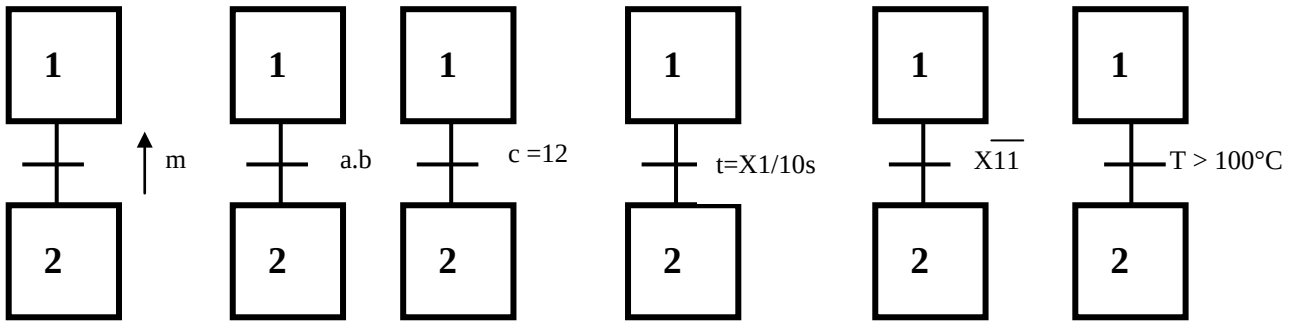


Mécanisme de franchissement d'une transition. Lorsqu'une transition est validée et que la réceptivité qui lui est associée est vrai la transition est franchie c'est à dire que les étapes précédant la transition sont désactivées et les étapes suivant la transition sont activées simultanément.

♦ **LES RECEPTIVITES** :

On associe à chaque transition une condition logique appelée **réceptivité** qui peut être soit vraie, soit fausse. Elle peut être fonction des variables externes (entrées, consignes affichées par l'opérateur) ou internes (compteurs, temporisations, étapes actives ou inactives

- **Exemple :**



• **Temporisation d'étape**.

Le lancement de la temporisation d'une étape est notée

$$\delta = t/Xi/q \text{ ou } Xi$$

est la variable logique associée à l'étape **i** et dont **la durée de la temporisation** est **q**.

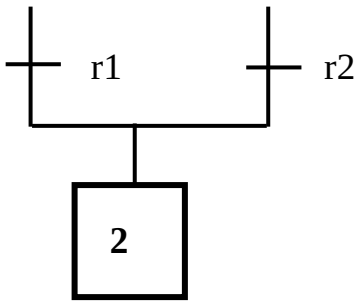
☐ 3.3 Autres éléments du Grafcet

♦ SEQUENCE UNIQUE

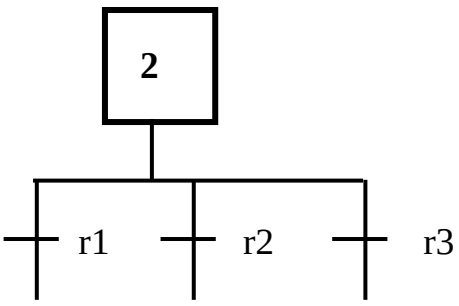
Elle est composée d'une suite d'étapes pouvant être activées les unes après les autres.
Chaque étape n'est suivie que par **une seule transition** et chaque transition **n'est validée que par une seule étape**. La séquence est dite **active** si au moins une étape est active. Elle est dite **inactive** si toutes les étapes sont inactives

♦ DIVERGENCE OU CONVERGENCE EN OU

Il s'agit d'un **aiguillage** ou d'une sélection de séquence selon certaines conditions données par les réceptivités associées aux transitions



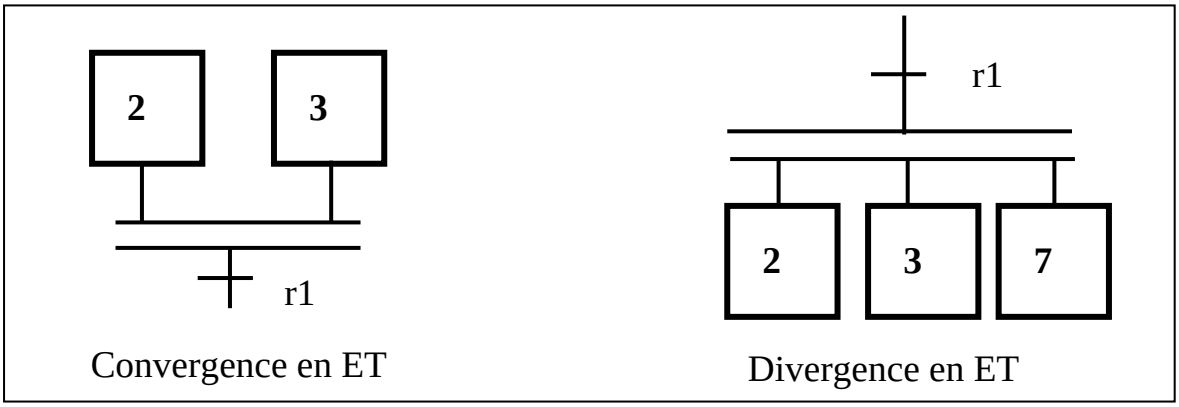
Convergence en OU
(Plusieurs transitions aboutissent à 1)



Divergence en OU
(Choix entre plusieurs transitions à 1

♦ **DIVERGENCE OU CONVERGENCE EN ET** :

Dit parallélisme **structural**. Le but est de permettre à l'automatisme **d'exécuter des séquences de façon simultanée**

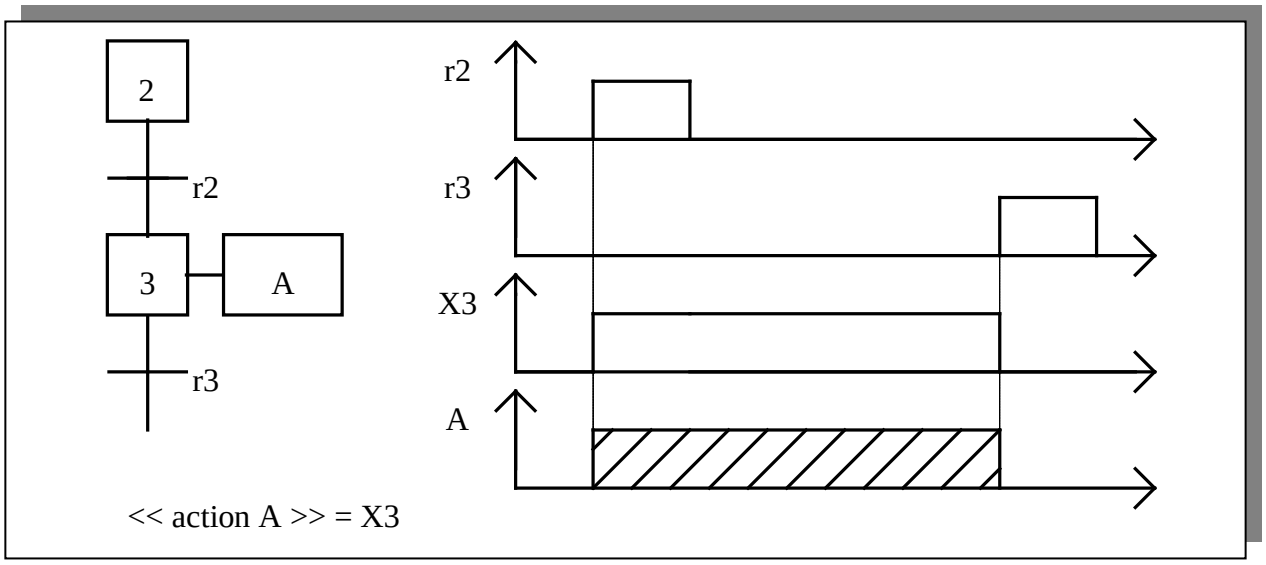


Remarque : Une divergence en OU commence et se termine toujours par une transition et une divergence en ET commence et se termine toujours par une étape.

☐LES DIFFERENTS TYPES D'ACTIONS :

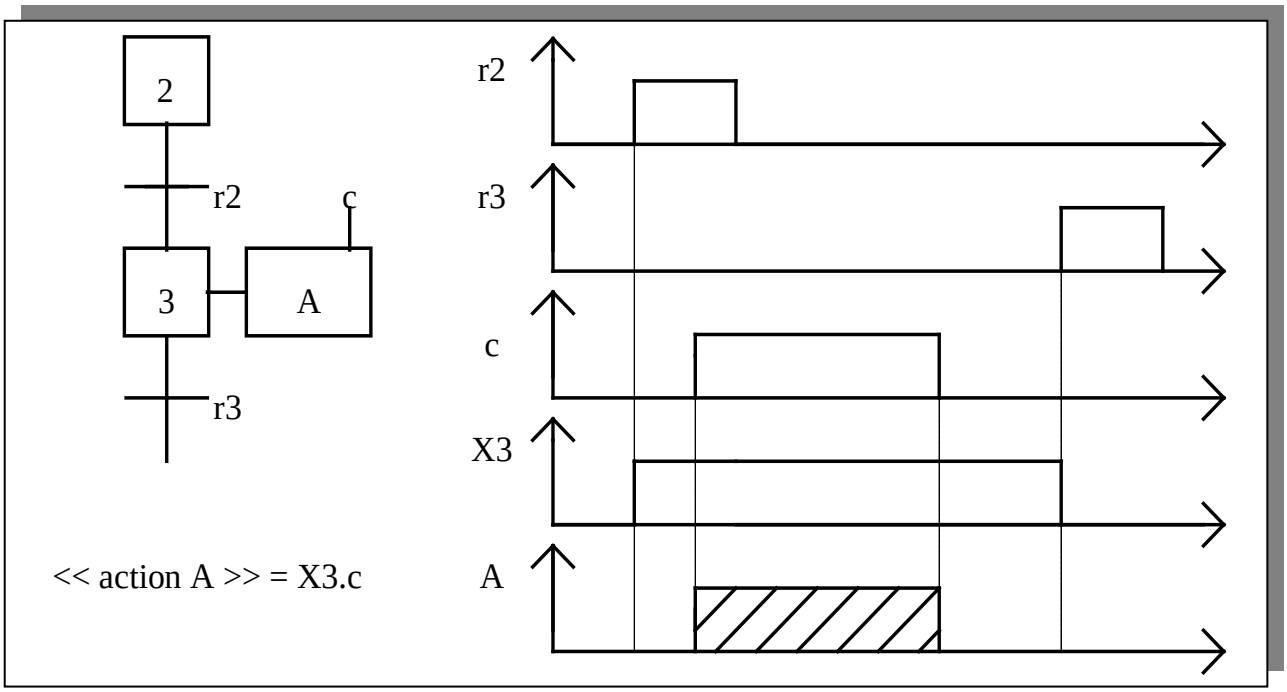
1. Action continue inconditionnelle :

L'ordre d'action est émis de façon continue tant que l'étape à laquelle il est associé est active.



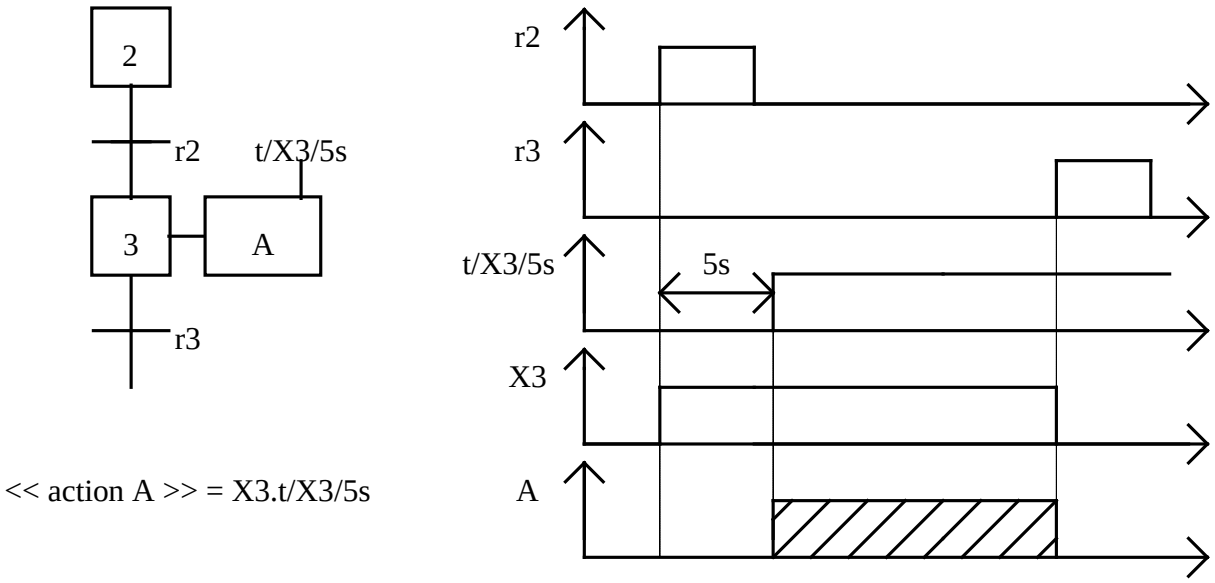
2. Action continue conditionnelle

C'est une action continue dont l'exécution est soumise à une condition logique.



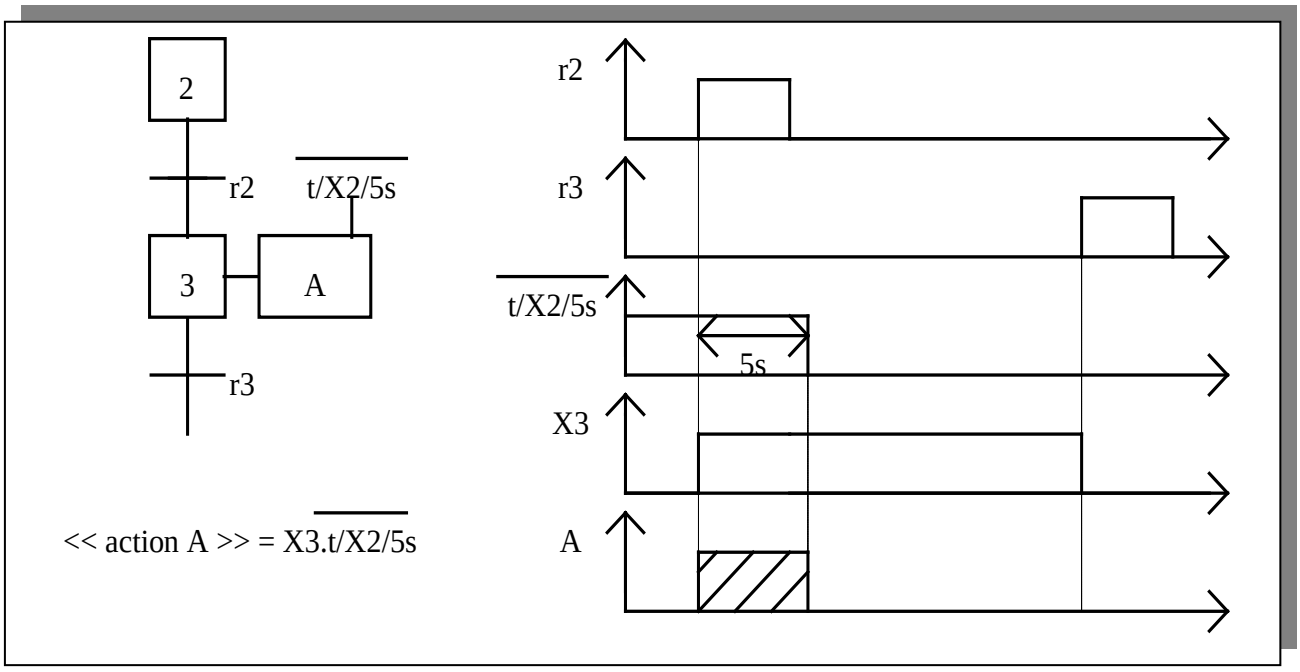
3. Action continue retardée :

Action continue conditionnelle dont la condition logique est une temporisation permettant de retarder l'action par rapport à l'activité de l'étape qui lui est associée.



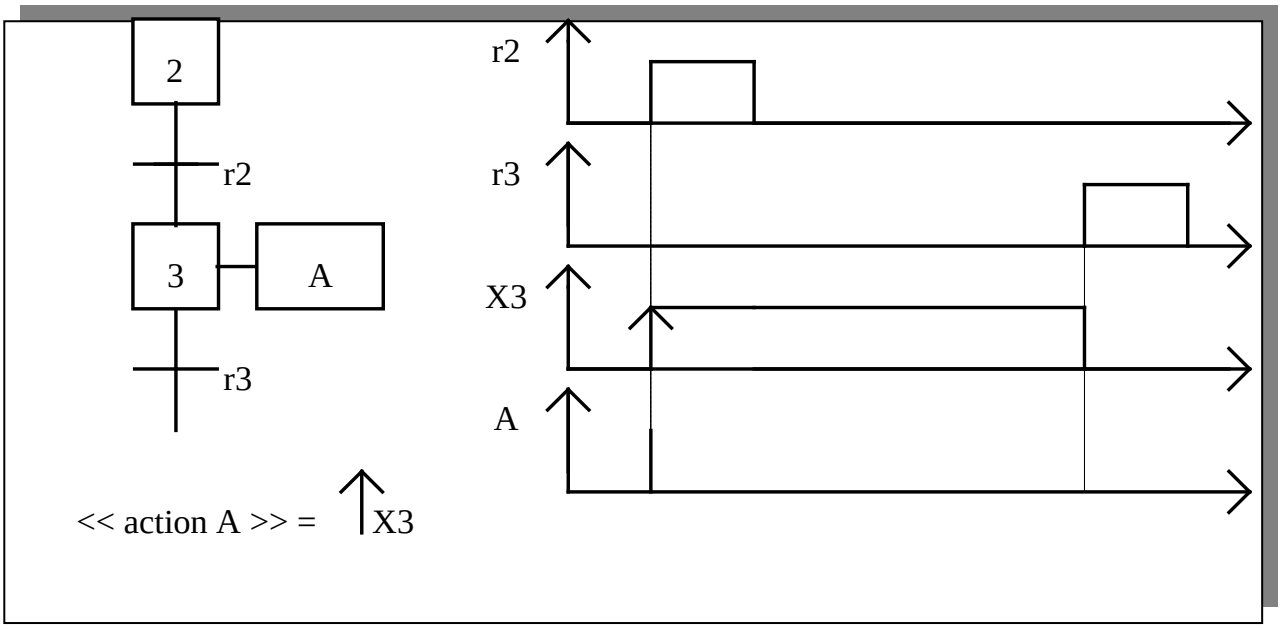
4. Action continue à durée limitée

Action continue conditionnelle dont l'ordre sera maintenu pendant un certain temps à partir de l'activation de l'étape qui lui est associée.



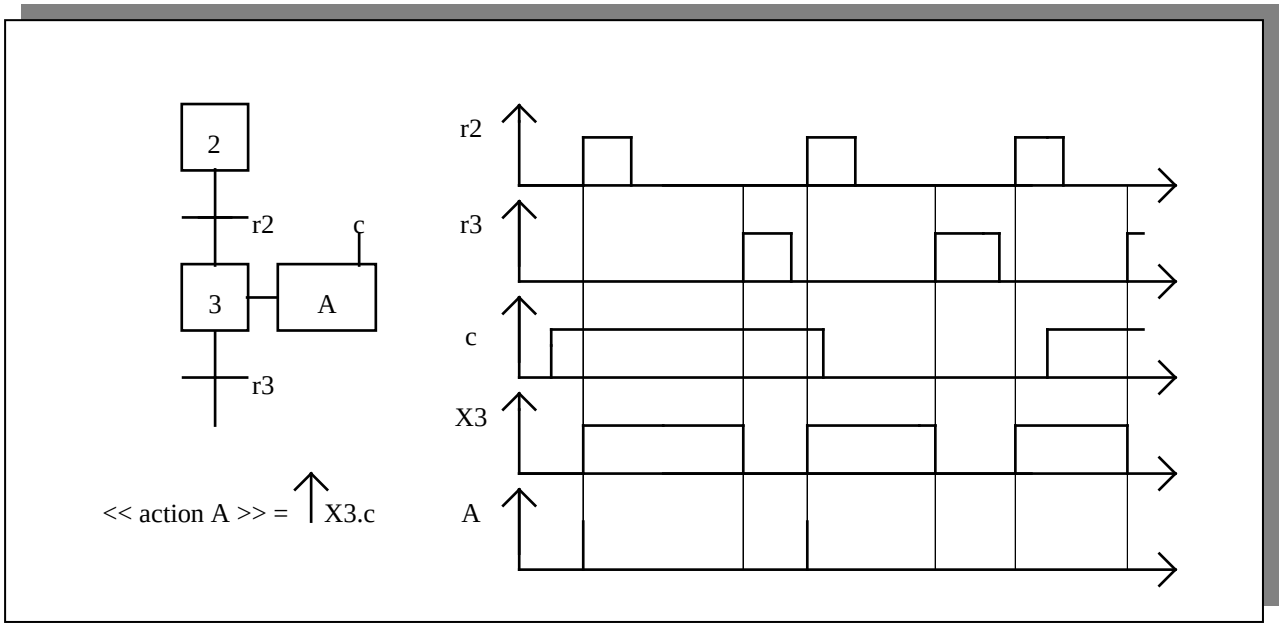
5. Action impulsionnelle inconditionnelle :

L''ordre d'action est fugitif et il est donné sur le front montant de l'activation de l'étape qui lui est associée.



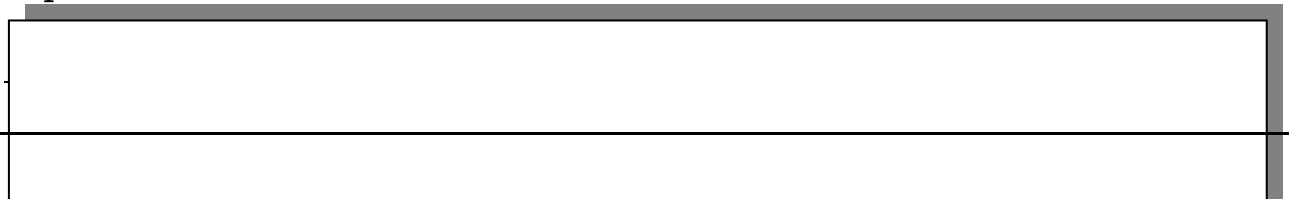
5. Action impulsionnelle conditionnelle :

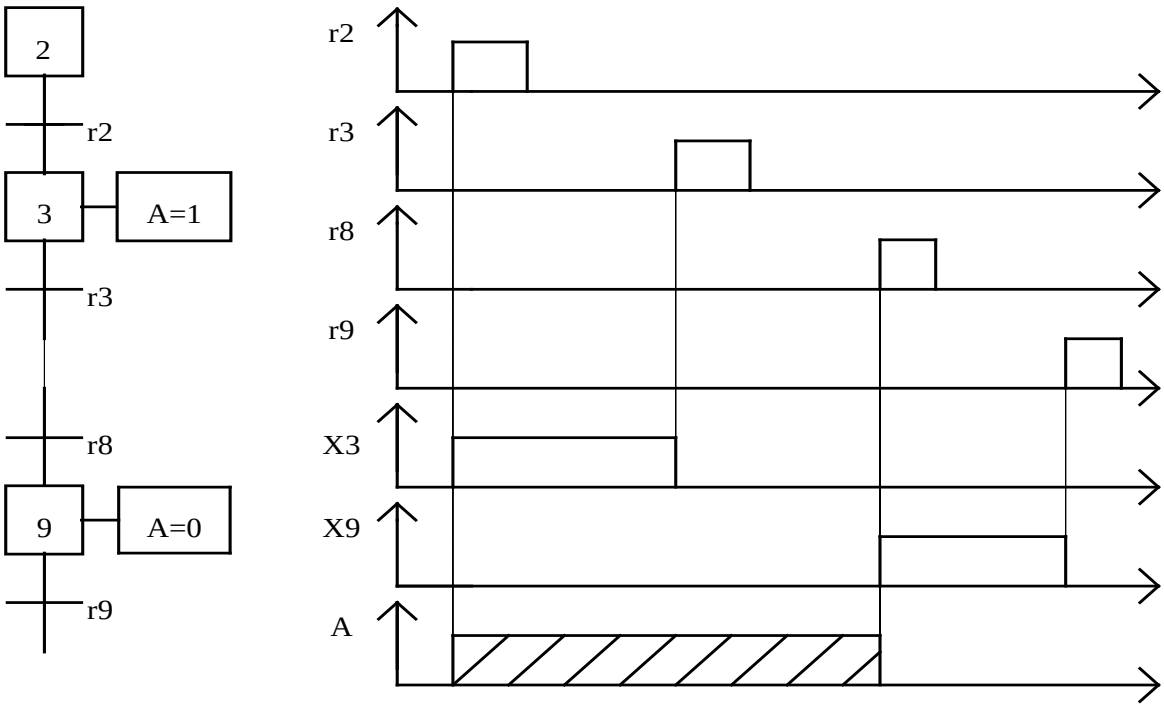
Action impulsionnelle dont l'ordre dépend d'une condition logique.



6. Action mémorisée :

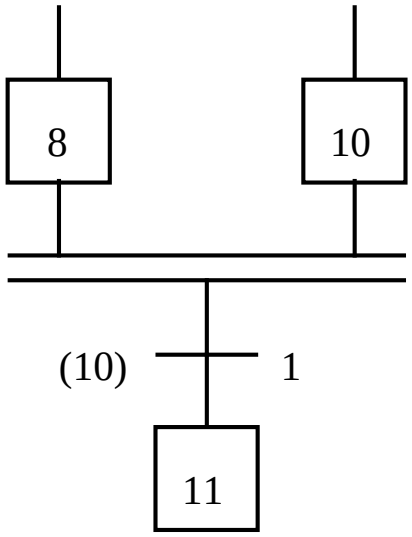
Action continue dont l'ordre d'exécution est mémorisé à une certaine étape et est maintenu tant que l'ordre d'arrêt n'est pas donné (il peut être donné à toute autre étape du Grafcet). On utilise généralement une variable booléenne qui est mise à 1 pour le début de l'action est à 0 pour son arrêt.





7._Réceptivité toujours vraie

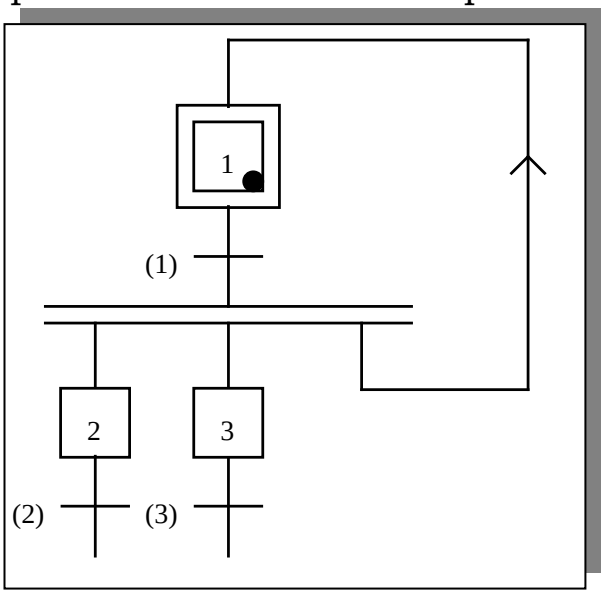
Est représentée par une **condition logique égale à 1**. Cette indication permet au concepteur de s'assurer qu'il n'a pas oublié de mentionner une réceptivité. Elle sert aussi de transition de synchronisation entre plusieurs séquences d'un Grafcet (dans le cadre des divergences et convergences en ET).



La transition (10) possède une réceptivité égale à 1. Ainsi le Grafcet se poursuit à l'étape 11 uniquement lorsque les deux étapes 8 et 10 sont actives car il y alors franchissement de la transition (10), désactivation des étapes 8 et 10 et activation de l'étape 11.

8.Transition toujours validée

lorsque l'étape précédant cette transition est toujours active. Ceci permet l'activation de séquences indépendantes

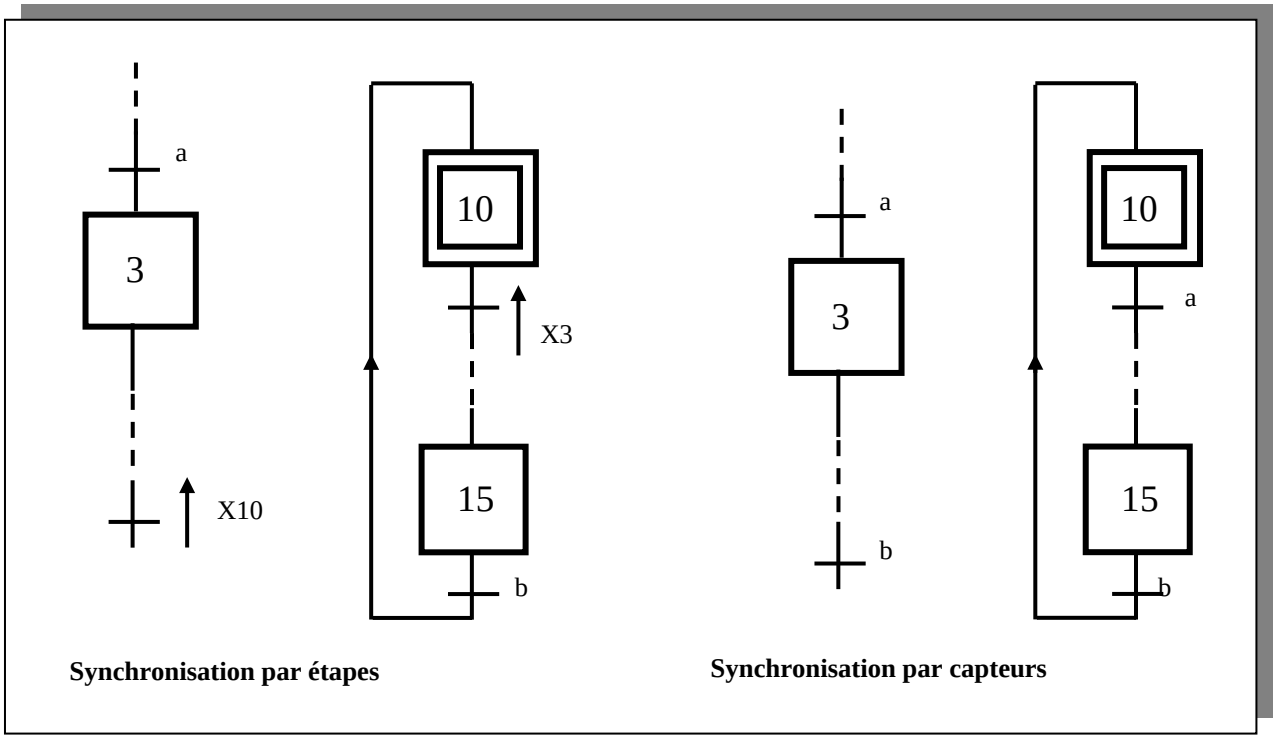


L'étape 1 étant toujours active, la transition (1) est toujours validée. A chaque fois que la réceptivité associée à la transition (1) est vraie, on active les deux étapes 2 et 3. Une divergence en OU à la suite de l'une de ces deux étapes permet alors la sélection d'une séquence en fonction de conditions associées aux réceptivités adéquates.

3.4 SYNCHRONISATION DES SOUS-GRAFCET

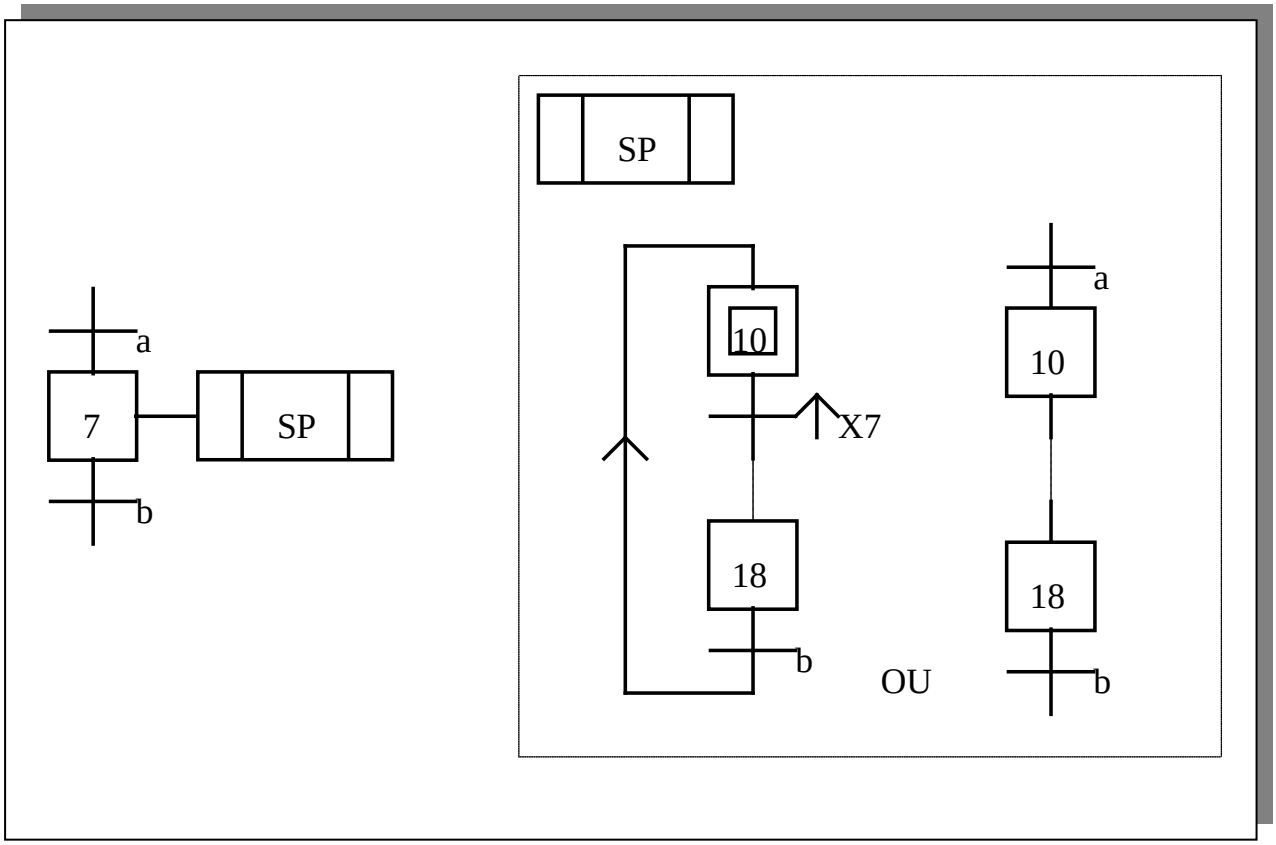
Un Grafcet possédant des séquences qui doivent être exécutées en parallèle peut être décomposé en plusieurs **sous-Grafcet**. Ces sous-Grafcet se construisent en respectant les mêmes règles de syntaxe et d'évolution que pour un Grafcet ordinaire. L'évolution de ces sous-Grafcet doit être synchronisée afin d'assurer un

fonctionnement correcte de l'automatisme. Cette synchronisation peut se faire par l'intermédiaire des variables d'étape $\mathbf{X_i}$ ou des **capteurs**.



3.5 LES SOUS-PROGRAMMES

Lorsqu'une même séquence est utilisée plusieurs fois, cette séquence peut être organisée sous une forme identique à celle d'un **sous programme** ou sous la forme d'un **sous-Grafcet synchronisé**.

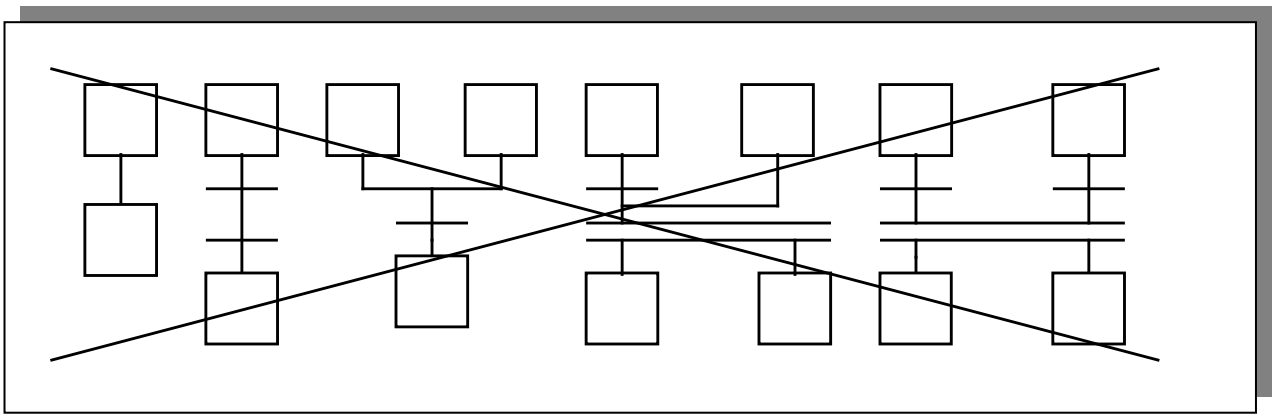


Chapitre 4 RÈGLES DU GRAFCET

♦4.1 Règles de syntaxe

Les règles de syntaxes se limitent aux choses suivantes: tout Grafcet doit commencer par une étape initiale, il faut respecter l'alternance étape - transition, une divergence en OU commence et se termine toujours par une transition et une divergence en ET commence et se termine toujours par une étape.

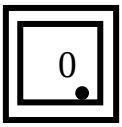
- **Exemple de choses interdites :**



♦ 4.2 Règles d'évolution

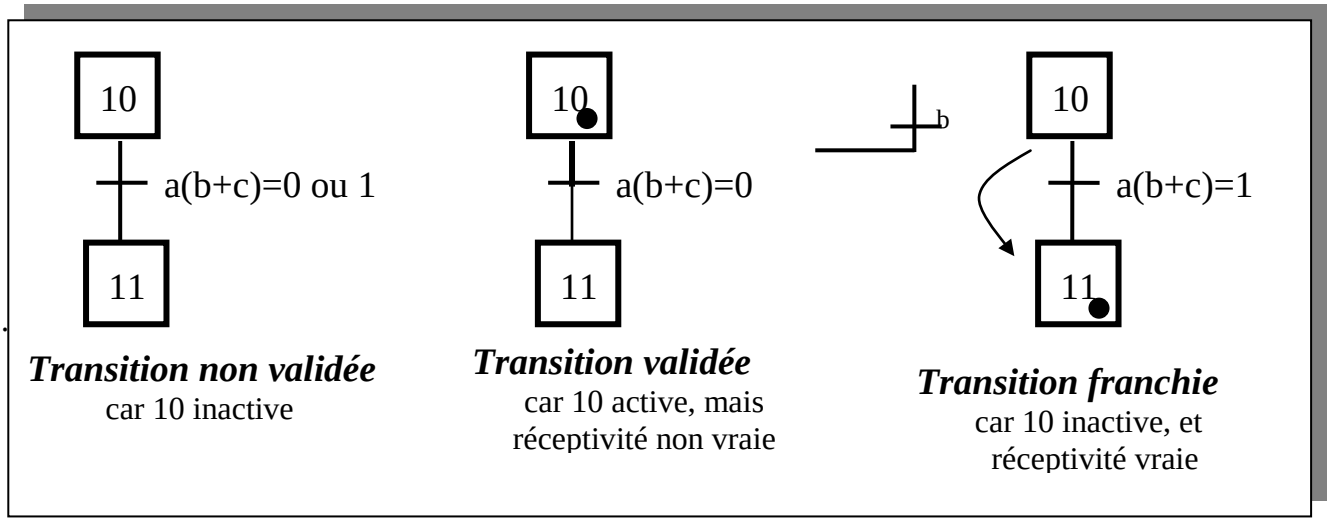
♦ Règle 1 : Situation initiale.

Le comportement initiale de la partie commande vis à vis de la partie opérative correspond aux étapes actives au début du fonctionnement de la partie commande. Ce sont des **étapes d'attente**



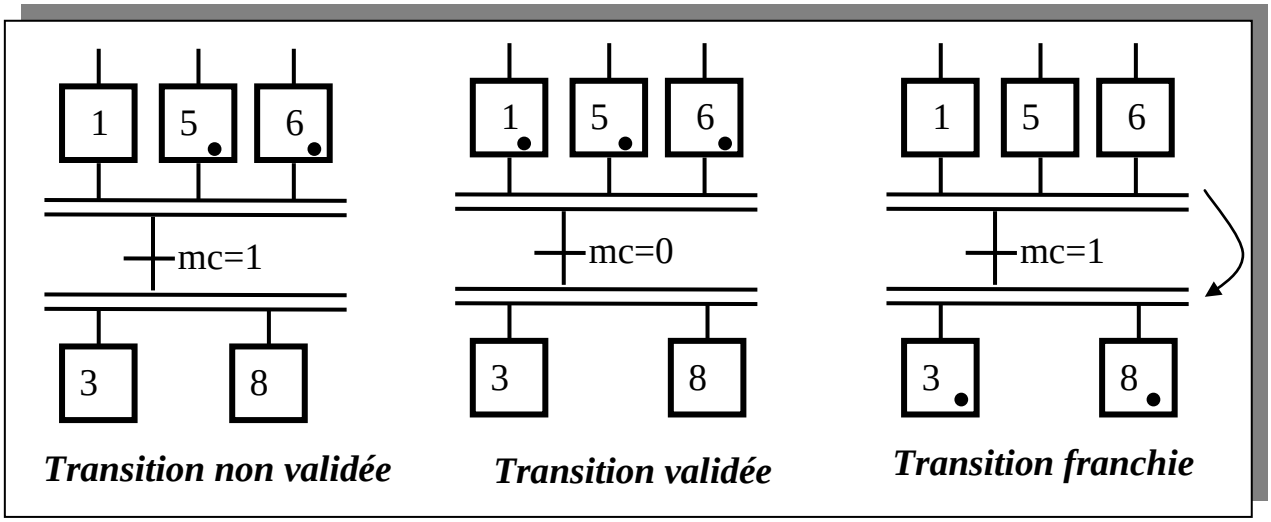
♦ Règle 2 : Franchissement d'une transition

Une transition entre étapes est dite **validée** si toutes les étapes d'entrée sont actives. Elle est **franchie** si elle est validée et si la réceptivité qui lui est associée est vraie.



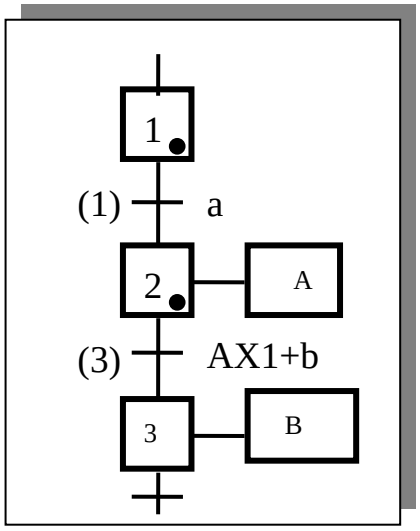
♦ Règle 3 : Évolution des étapes actives.

Le franchissement d'une transition entraîne **l'activation** de toutes les étapes qui suivent immédiatement cette transition et la **désactivation** de toutes les étapes qui précèdent immédiatement cette transition.



♦ Règle 4 :Évolutions simultanées.

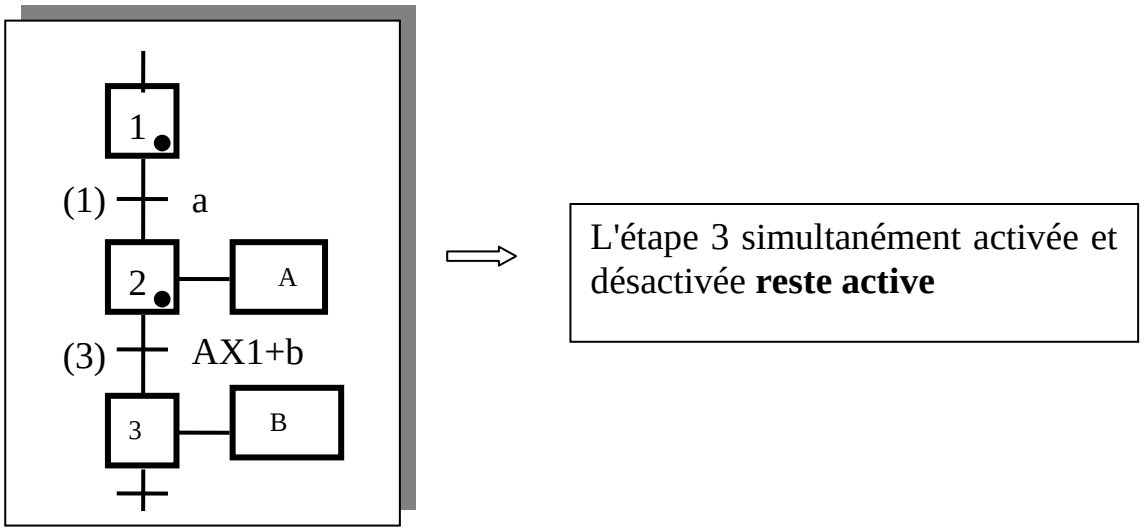
Plusieurs transitions **simultanément franchissables** sont **simultanément franchies.**



Les étapes 1 et 2 sont actives (X1=X2=1). les transitions (1) et (2) sont alors validées, lorsque a=1 (2) est aussi vraie **alors (2) et (3) simultanément franchissables** sont **simultanément franchis.**

♦ **Règle 5 : Activations et désactivation simultanées**.

Si au cours du fonctionnement de l'automatisme, une même étape doit être désactivée et activée simultanément, elle **reste active.**



4.3 SIMULTANEITE D'EVENEMENTS DANS UN GRAFCET

♦ **Définition : un événement externe** noté **Ei** est un front montant ou descendant d'une variable externe.

♦ **Postulat :** Le modèle Grafcet exclut formellement la simultanéité d'événements d'occurrence 2. C'est à dire que les événements externes sont indépendants (on ne peut jamais avoir deux fronts montants de variable simultanément).

♦ **Remarque: Structure d'une réceptivité**

Une réceptivité est une proposition logique pouvant comporter une ou deux parties: une partie dite "**active**" ou événement qui provoque le

franchissement de la transition validée. Une partie dite "**passive**" ou condition qui provoque le franchissement de la transition lorsqu'elle est vraie.

- **Réceptivité statique** : elle ne comporte qu'une partie passive. Elle est vraie quand elle est égale à 1 (exemple: a, a.b, $a+X_2$).

- **Réceptivité impulsionnelle** : Elle ne comporte qu'une partie active (exemple: ↑a, ↑(a+b))

- **Réceptivité dynamique** : elle comporte une partie passive et une partie active. Elle est vraie lorsque la partie impulsionnelle devient égale à 1 et que la partie statique est égale à 1. (exemple: ↑m.h).

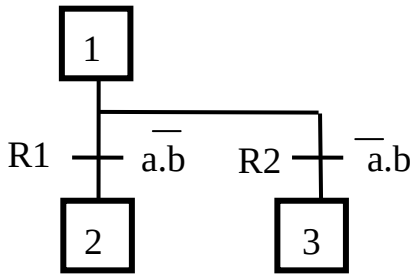
4.4 RECEPTIVITES ET REGLES D'EXCLUSION

♦ Règle d'exclusion divergente

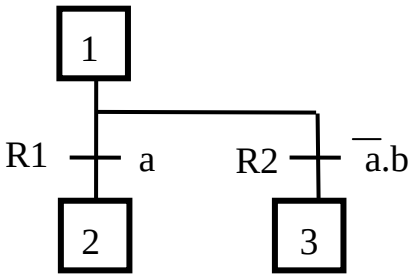
Deux réceptivités R_1 et R_2 s'excluent mutuellement si et seulement si $R_1.R_2 = 0$.

Ainsi pour rendre plusieurs séquences exclusives, il est nécessaire de s'assurer que toutes les réceptivités associées aux transitions initiales de ces séquences ne puissent être vraies en même temps.

R1 et R2 sont normalement exclusives car R1.R2 = 0.



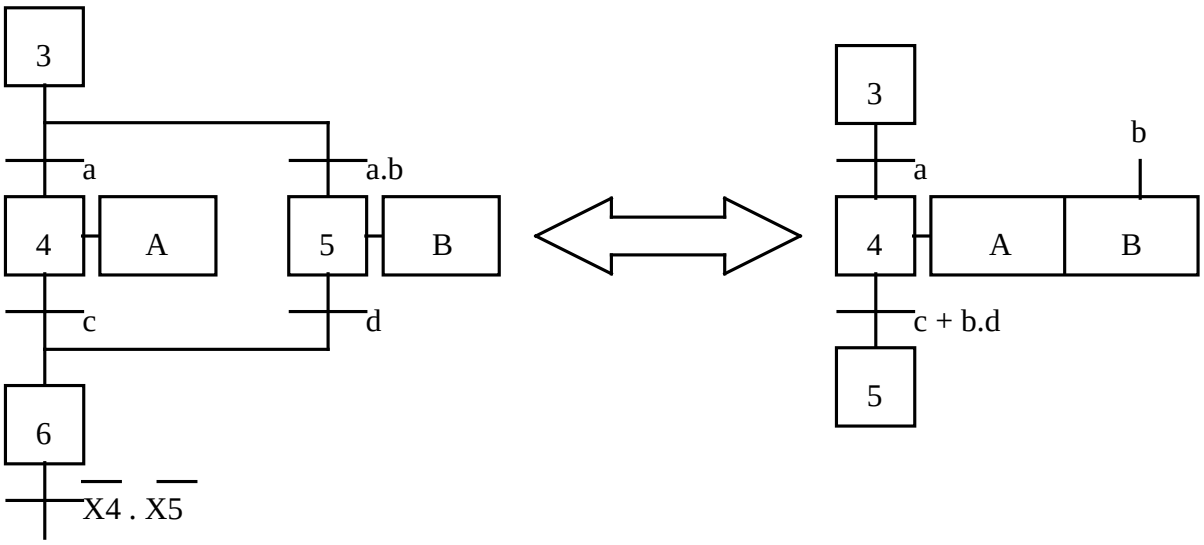
Si a=b=1 aucune des deux étapes 3 et 4 n'est activée



Si a=b=1 l'étape 3 est activée prioritairement

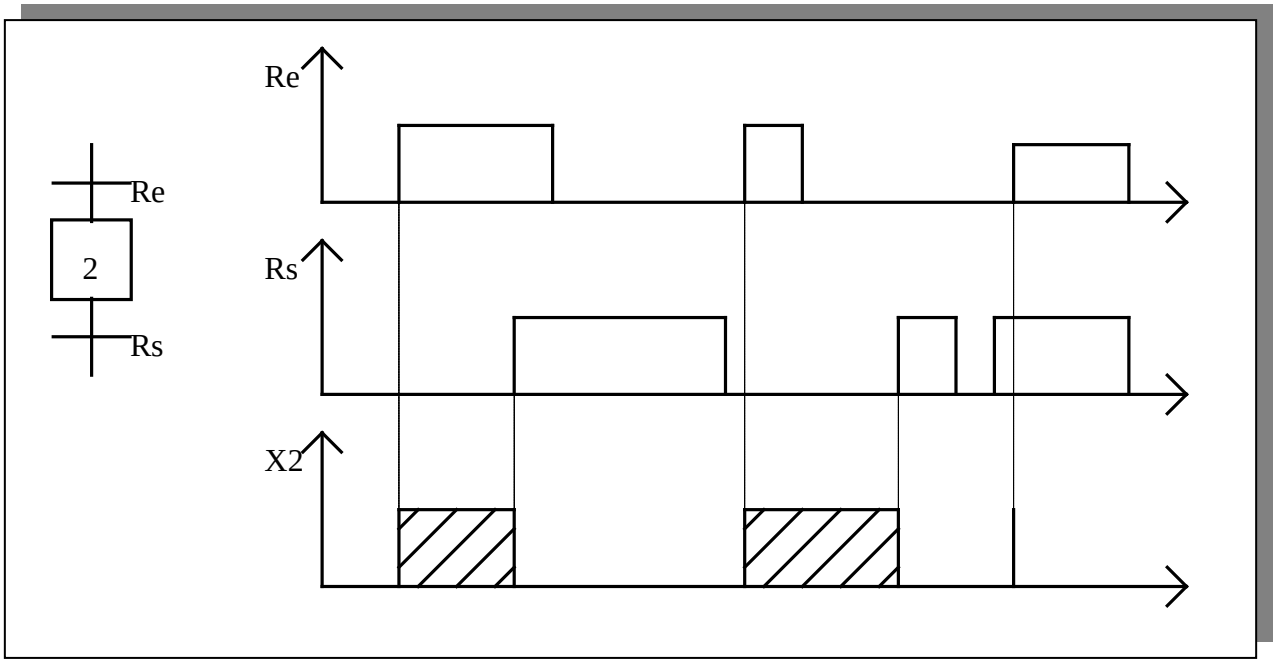
♦ Le parallélisme interprété.

Il est possible de faire évoluer le Grafcet sur plusieurs séquences simultanées sans que ces séquences ne soient commandées par une transition unique. C'est le cas où les réceptivités associées aux transitions validées ne sont pas exclusives et conduisent à activer plusieurs étapes à la fois. On appelle cela le **parallélisme interprété** et il doit être éviter ou utiliser avec prudence car la plus grande difficulté réside dans la spécification correcte de la façon dont il se termine.



♦ Règle d'exclusion transitive :

Elle ne s'applique que dans le cas de réceptivités statique. Pour qu'une étape ne soit pas fugitive il faut que $R_e.R_S = 0$ au moment de l'activation de l'étape, avec R_e: réceptivité entrante et R_S: réceptivité sortante. Une étape fugitive peut être caractéristique d'un mauvais fonctionnement de l'automatisme.



☐4.5 ÉLABORATION D'UN GRAFCET

Un Grafcet est toujours conçu, consciemment ou non d'après une ou plusieurs idées directrices. Il faut en prendre conscience dés le départ et les noter, car sinon on risque d'en changer au cours de l'élaboration ce qui peut rendre le Grafcet illogique et inexploitable. Une élaboration méthodique s'effectue en 5 phases:

1. **Dénombrement et dénomination des étapes:** à chaque étape correspond une "idée" qui doit donner son nom à l'étape considérée.

2. **Repérage des transitions:** connaissant les étapes qu'il se propose d'étudier, le concepteur dresse la liste des transitions possibles compte tenu du cahier des charges. Ce après quoi il trace la structure du Grafcet.

3. **Actions:** Se référant aux dénominations des étapes, le concepteur indique les actions, ou les séquences d'actions qui leur sont associées. Les séquences d'actions sont traitées en sous-programmes.

4. **Réceptivités:** Elles se déterminent en fonction du cahier des charges.

5. **Règles d'exclusion:** L'application de ces règles permet de détecter un certain nombre d'anomalies et d'erreurs flagrantes qui auraient pu être commises lors de la détermination des réceptivités.

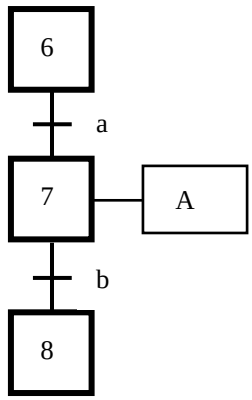
5 MATÉRIALISATION DES GRAFCET

5.1 Matérialisation d'un Grafcet par un séquenceur câblé

Un Grafcet est constitué d'un certain nombre d'étapes. A chaque étape ***i***, on associe une variable booléenne X_i représentant son activité (**X_i=1: l'étape i est active, X_i=0: l'étape i est inactive).** Il est alors possible de matérialiser **X_i par la variable de sortie Qi** d'une bascule (RS, JK ou D). La synthèse de l'automate consiste à calculer les entrées (R_i, S_i, J_i, K_i ou D_i) de ces bascules en fonction des variables Xi et des réceptivités associées aux transitions du Grafcet.

Il reste à résoudre le problème du marquage initial: les bascules correspondantes aux étapes initiales doivent être mises à 1 et les autres à 0. On utilise pour cela une entrée d'initialisation I qui permet de forcer les bascules dans la configuration initiale.

♦ **Exemple :**



Considérons l'étape 6. Si nous utilisons des bascules RS, la sortie Q_7 correspondant à X_7 est mise à 1 quand la transition précédante est franchie. Autrement dit il faut que X_6=1 et que a=1 (réceptivité à 1) donc nous obtenons S_7= X_6.a Cette bascule est remise à zéro lorsque l'étape 8 est activée (X_8=1) ou lorsque le Grafcet est forcé à son état initial (I=1). De cette façon on en déduit que R_7=X_8+I. L'action A est continue donc maintenu tant que X_7=1, ainsi A=X_7. Il suffit en suite de reconduire cette étude pour toutes les étapes du Grafcet et de tracer le logigramme correspondant.

5.2 L'automate programmable

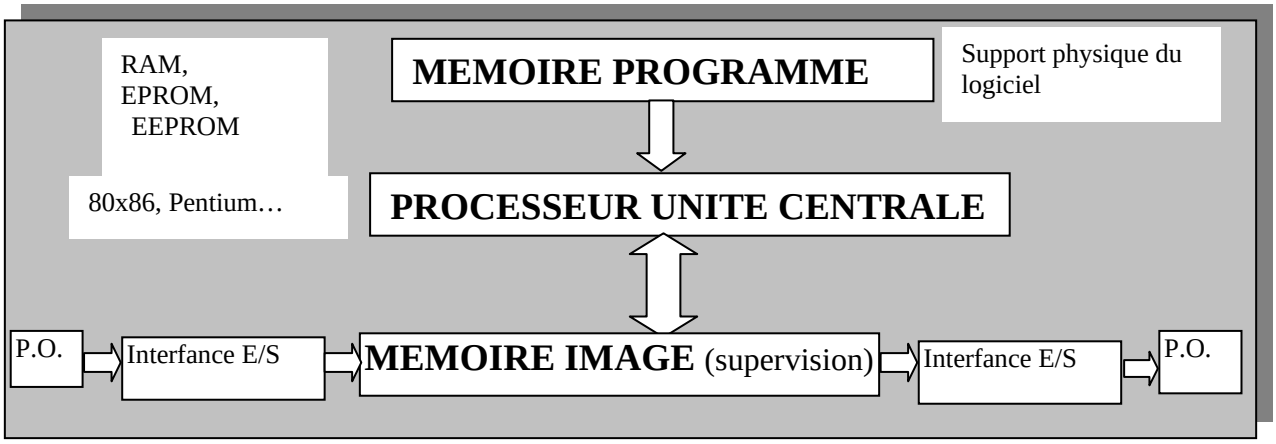
5.2.1 Définition: Un automate est une structure informatique industrielle travaillant en temps réel, pilotant un partie opérative et pouvant être programmée par un automaticien à l'aide d'un **langage adapté.**

Le but est de piloter une partie opérative avec un calculateur convenablement programmé. Du point de vue matériel il y a plusieurs solutions :

- **Mini ordinateur ou calculateur**: (PDP, Solar, VAX...) avec des capteurs d'entrées/sorties industriels.
- **Microprocesseur + mémoire + interface d'E/S** + autres interfaces: **une telle configuration est appelée A.P.I. (Automates Programmables Industriels)**. Les A.P.I. sont

conçus pour résister à l'environnement industriels (protection contre les parasites, les surtensions, les défauts d'isolement). Ils sont munis de batteries (protection contre les coupures) et de chiens de garde (pour le contrôle des cycles machine). L'architecture des A.P.I. est la suivante :

5.2.2 Architecture d'un API



5.2.3 Types d'Entrées/Sorties (E/S):

1. **TOR (tout ou rien)** : 0-5 volts (TTL), 0-24 volts, 0-48 volts, 0- 220 volts
2. **Analogiques :** pour mesurer des pressions, températures..

   0-10 Volts (boucle de tension) 4-20 milliampères (boucle de courant)
3. **numériques** : valeur binaire sur N bits: code à barre

5.2.4 Principaux fabricants d'automates :

- ♦ Télémécanique (N°1 Français)
- ♦ Siemens (N°1 Mondial)
- ♦ Allen Bradley (N°2 Mondial)

♦ Hitachi , Yukogawa…

5.3 LANGAGES LITERAL

Le langage littéral est composé d'une suite d'instructions qui comportent:

♦ **Une étiquette** facultative : c'est un mnémonique à 3 lettres permettant de désigner une instruction par sa fonction plutôt que par son numéro.

♦ **Un numéro de ligne** : il s'agit en fait d'une adresse décimal permettant de repérer une instruction.

♦ **Une instruction** : elle est en général constitué d'un opérateur et d'un opérande.

Exemple: ACT 15 D=X3
Etiquette Adresse décimale Instruction : opérateur, opérande

♦ Différents éléments du langage littéral

♦ **Les expressions booléennes** (<e.b>). Elles sont composées uniquement de variables et des opérateurs logiques ET, OU et NON. Elles peuvent aussi être constituées de variables temporisées : (<e.b>,ζ) : expression booléenne temporisée à l'appel.

♦ **Les affectations** Elles sont de quatre types :

1. Simples directes: **X = <e.b>**

2. Simples inverses: $\overline{X}$ **= <e.b>**

3 Mémorisées directes: **X ≅ <e.b>** (si <e.b> = 1 alors X = 1)

4. Mémorisées inverses: $\overline{X}$ **≅ <e.b>** (si <e.b> = 1 alors X = 0)

♦ **Les branchements**

Ils sont de deux types :
1. Branchements inconditionnels (saut): **ST <adresse>**
2. Branchements conditionnels:
   si <e.b1> = <e.b2> alors ST <adresse>
   si <e.b1> ≠ <e.b2> alors ST <adresse>

♦ **L'adressage** peut être de trois types:

1. Adressage symbolique: **ST <étiquette>**

2. Adressage indexé: **ST + <incrément>**

3. Adressage absolu (direct): **ST <N° d'instruction>**

5.4 MATERIALISATION D'UN GRAFCET SUR AUTOMATE

Pour matérialiser un Grafcet sur un automate nous allons procéder par gestion autours des transitions. Le programme contient successivement :

- La programmation des échanges (mnémonique ECH) : cette gestion est effectuée par l'automate.
- **L'animation du Grafcet** qui est décomposée en trois sous-ensembles :

1. **Le calcul des franchissabilité (mnémonique FRA) :** la franchissabilité d'une transition est une variable logique notée $\mathbf{F_i}$ qui vaut 1 si et seulement si cette transition peut être franchie. Autrement dit il faut que les étapes immédiatement précédantes soient actives et que la réceptivité associée à cette transition soit vraie. Ainsi $\mathbf{Fi=X_{i-1}.C_i}$.

3. **Le franchissement des transitions (mnémonique FTR) :** pour chacune des transitions on évalue sa franchissabilité (Fi) et lorsqu'elle est vraies on désactive les étapes immédiatement

précédantes et on active les étapes immédiatement suivantes si Fi=0 l'état du Grafcet reste inchangé. Ainsi $\overline{X}_i \cong F_i$ et $X_{i+1} \cong F_i$.

3. <u>Le calcul des variables de retard</u> (mnémonique RET) : dans le cas où le Grafcet contient des réceptivités dynamiques on utilise des variables de retard notées **M*** afin de modéliser les fronts d'une variable. Ainsi on a: $\uparrow M = M.\overline{M}^*$. Cette valeur **M*** est en fait la valeur de M au cycle d'automate précédant :

M	**0**	**0**	**0**	**1**	**1**	**1**
M*	**-**	**0**	**0**	**0**	**1**	**1**
$\overline{M}^*$	**-**	**1**	**1**	**1**	**0**	**0**
M. $\overline{M}^*$	**0**	**0**	**0**	**1**	**0**	**0**

- **<u>L'initialisation</u> (mnémonique INI) :** Le but de cette partie est d'activer les étapes initiales et de désactiver toutes les autres. On utilise pour cela une variable booléenne supplémentaire notée I de la façon suivante :

<u>Étape initiale :</u> $X_i \cong I$

<u>Étape non initiale :</u> $\overline{X}_i \cong I$

- **<u>Les actions</u> (mnémonique ACT) :** Les actions sont calculées dans cette dernière partie en fonction de l'activité des étapes et des conditions qui leur sont associées :

<u>Action simple :</u> $A = X_i$ action inconditionnelle

$A = X_i.c$ action conditionnelle

<u>Action mémorisée :</u> $A \cong X_i$ action inconditionnelle mémorisée à l'appel

$A \cong X_i.c$ action conditionnelle mémorisée à l'appel

$\overline{A} \cong X_i$ action inconditionnelle mémorisée à la retombée

$\overline{A} \cong X_i.c$ action conditionnelle mémorisée à la retombée

- **Structure d'un programme littéral :**
  - **ECH** : Gestion des échanges.
  - **FRA** : Calcul des franchissabilités.
  - **FTR** : Calcul des franchissements.
  - **RET** : Calcul des variables de retard.
  - **INI** : Initialisation du Grafcet.
  - **ACT** : Calcul des actions.