Le Grafcet – G7

grafcet fonctionnel

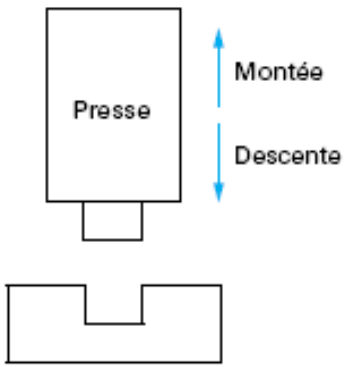
vs

grafcet technologique

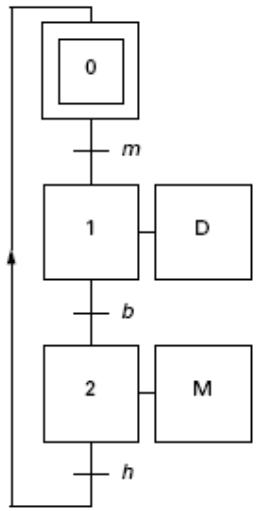
Grafcet fonctionnel / technologique

- **grafcet fonctionnel** : prise en compte de la partie fonctionnelle, en faisant abstraction de toute réalisation technologique
- **grafcet technologique** : en s'appuyant sur le grafcet fonctionnel, intègre les contraintes technologiques et opérationnelles.

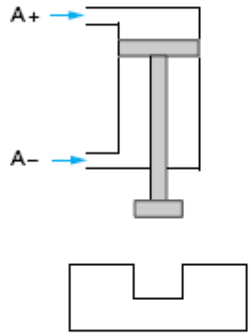
principe



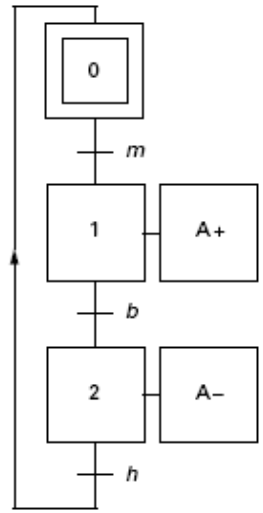
grafcet fonctionnel



technologie pneumatique



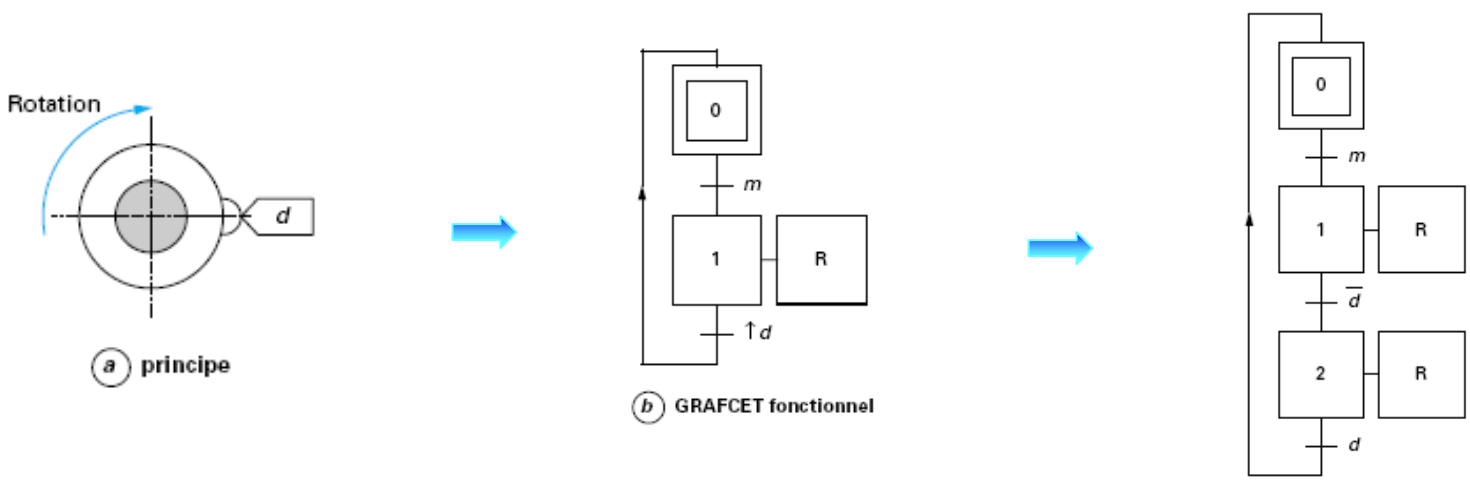
grafcet technologique



Grafcet fonctionnel / technologique

- Pour garantir l'indépendance du grafcet fonctionnel, faire attention à certains cas particuliers.

- **Gestion des fronts** : permet de tester l'apparition / disparition d'evt plutôt que leur présence. C'est le cas lorsqu'une information est déjà présente dans l'état initial.
- Exemple : commande d'un moteur : lorsque l'opérateur commande la rotation, l'information du capteur *p* de position est déjà vraie => tester l'apparition de *p* et non sa présence.
- Rmq : en fait le test d'un front montant se traduit ds le grafcet technologique à l'aide d'une étape supplémentaire.



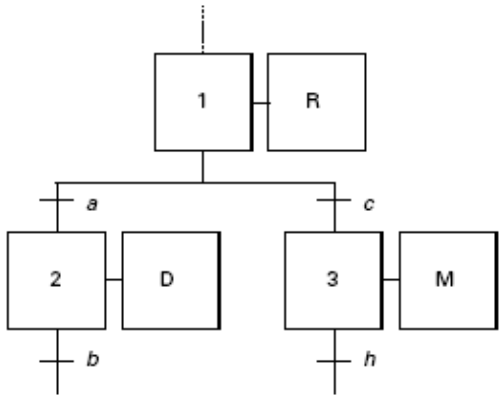
(a) principe

(b) GRAFCET fonctionnel

(c) GRAFCET de gestion de front

Grafcet fonctionnel / technologique

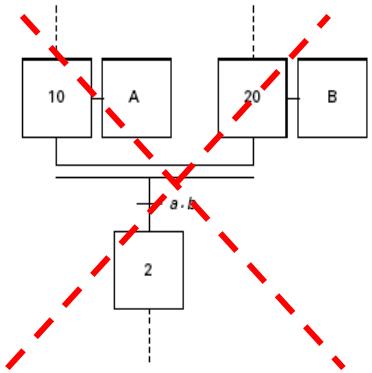
- **<u>Exclusivité au niveau des divergences</u>** : l'exclusivité de *a* et *c* peut être :
  - liée au procédé : par ex. capteurs "opposés" (objet à gauche / objet à droite);
  - incompatibilité temporelle : a priori jamais en même temps;
  - exclusion logique : structurellement, cf. schéma.

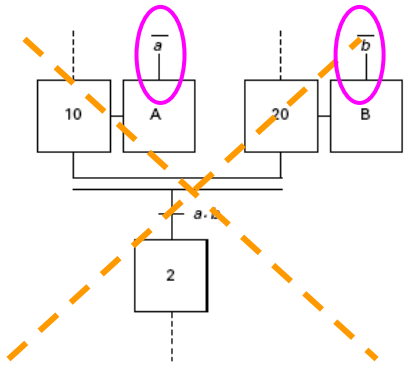


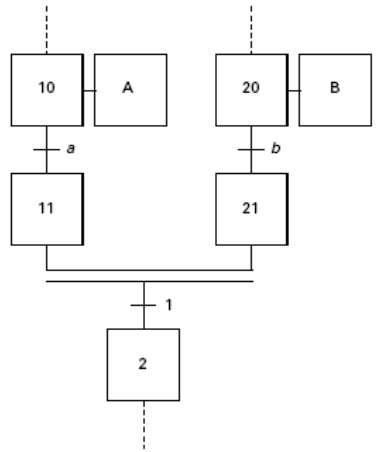
- <u>Mais pour garantir l'indépendance, il vaut mieux expliciter l'exclusivité de façon structurelle</u>

Grafcet fonctionnel / technologique

- **Gestion des simultanéités en fin de convergence**
- **Hypothèse sur les capteurs**







Pb : si une action finie avant l'autre, elle ne peut pas s'arrêter

1 solution, mais avec hypothèse : ***capteurs à contact maintenus***

solution la meilleure.

Le Grafcet – G7

Dialogue entre grafcets

Le Grafcet – G7

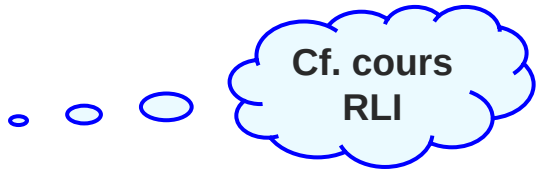
Dialogue entre grafcets :

Communication dans l'entreprise

- **CF COURS RLI**

Communications

- Evolution des architectures d'automatismes : introduction de communications de façon hiérarchique
- En entreprise : hiérarchie des communications = pyramide CIM (Computer Integrated Manufacturing)



- **Avantages :**
  - **Réduction du câblage**
  - **Réduction des coûts (câblage, interface PO/PC (interface capteurs))**
  - **Modularité**

- Inconvénients :
  - Gestion des délais, pertes, contraintes temporelles, répartition des données
  - ⇒ *problématique des **réseaux locaux industriels (RLI)***

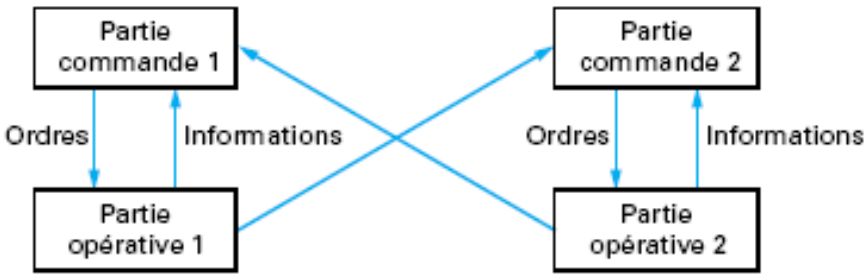
Le Grafcet – G7

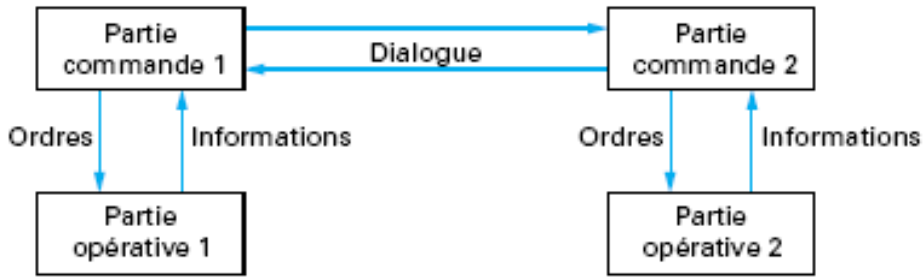
Dialogue entre grafcets :

Division technologique

Réalisation technologique

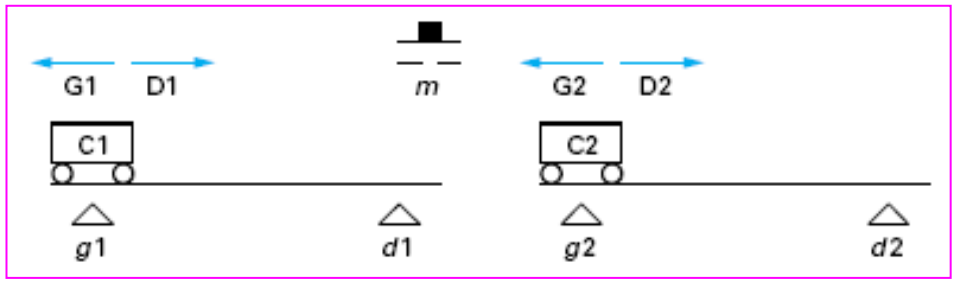
- Il est donc souvent nécessaire ou judicieux de **diviser la partie commande et/ou la partie opérative** :
  - Application complexe divisée en sous parties de moindre complexité;
  - Synchronisation et dialogue entre sites réparties géographiquement;
  - Intégration du concept de CIM avec nécessité d'optimiser les communications entre niveaux.

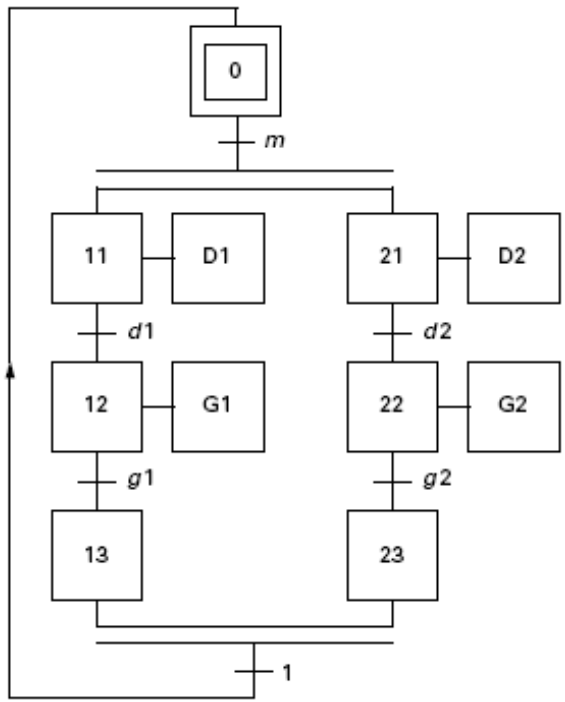


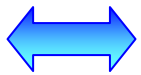


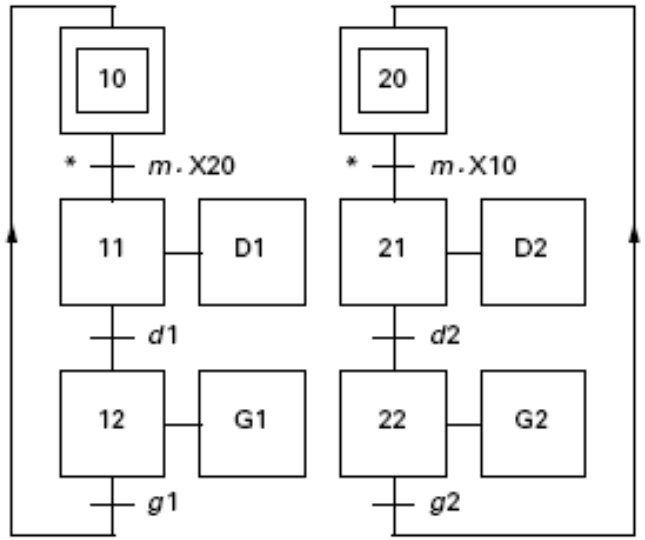
Réalisation technologique

- Exemple : 2 chariots







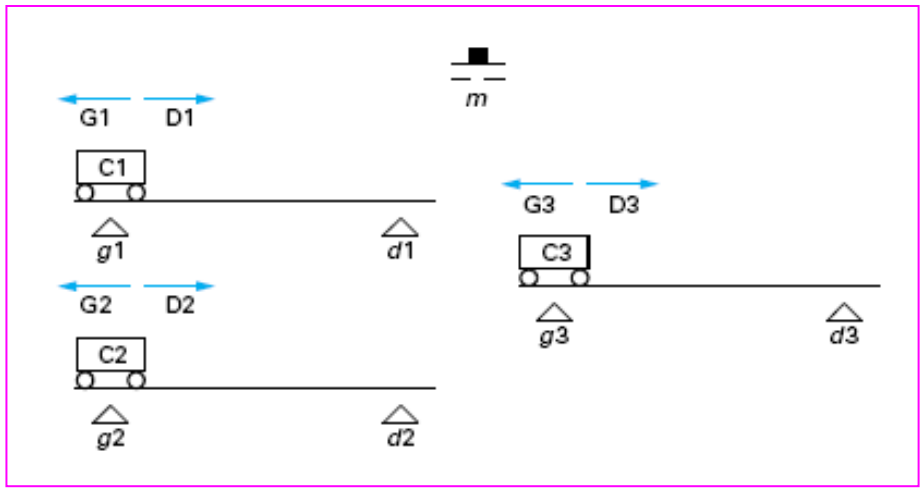


Le Grafcet – G7

Hiérarchisation de la partie commande

Hiérarchisation

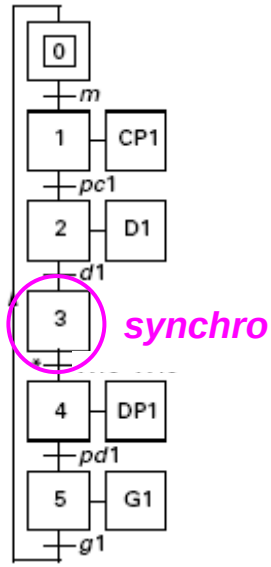
- Exemple : 3 chariots.
  - Les chariots 1 et 2 se chargent (*CPi*) à gauche et se déchargent (*Dpi*) à droite dans le chariot 3;
  - chariot 1 en 1er
  - Le chariot 3 se décharge (*Dp3*) à droite.



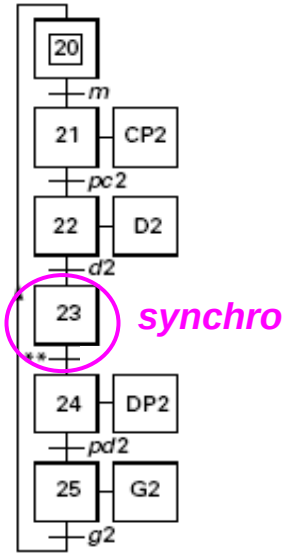
Hiérarchisation

- Découpage de la PC :
  - Gestion des chariots (3 grafcets, un par chariot)
  - Gestion des synchronisations, mémorisations, ressources.

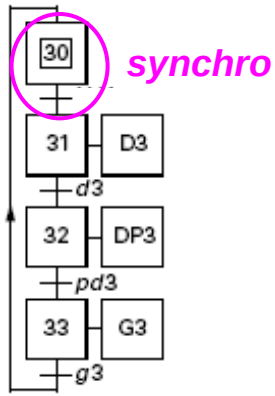
chariot 1



chariot 2



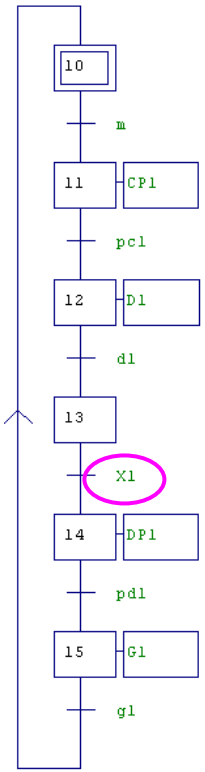
chariot 3



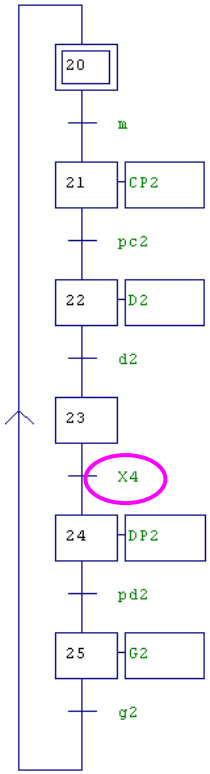
Hiérarchisation

- Synchronisation et gestion de la ressource chariot 3 :

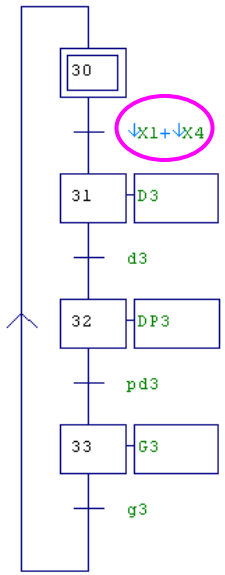
chariot 1



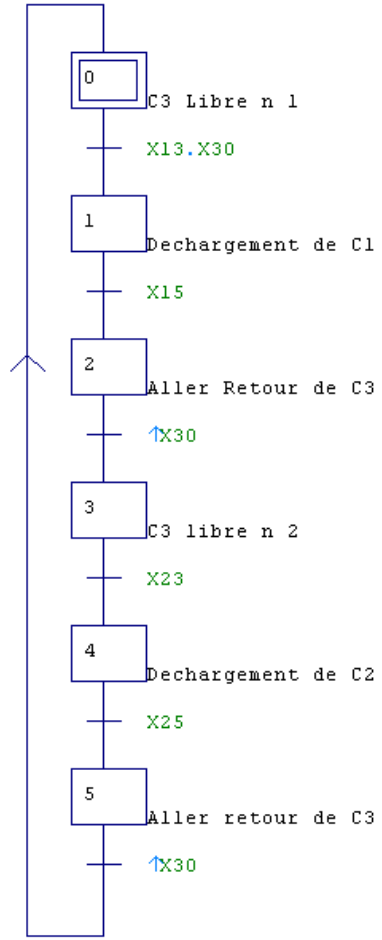
chariot 2



chariot 3

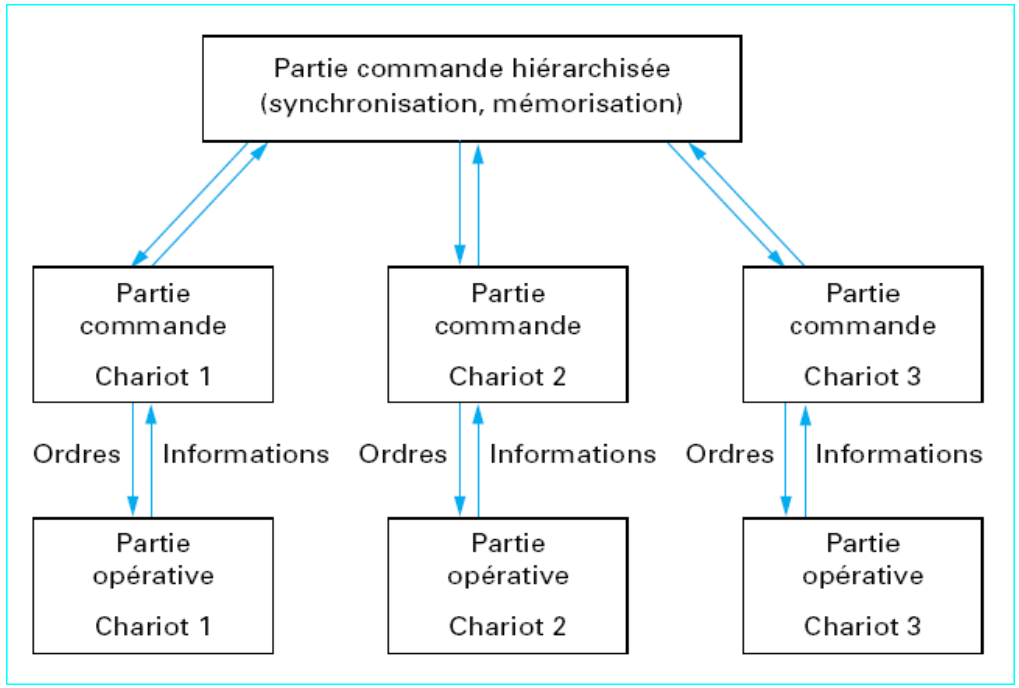


grafcet de synchronisation



Hiérarchisation

- **Commande hiérarchisée :**



Le Grafcet – G7

Implantation

Mise en oeuvre

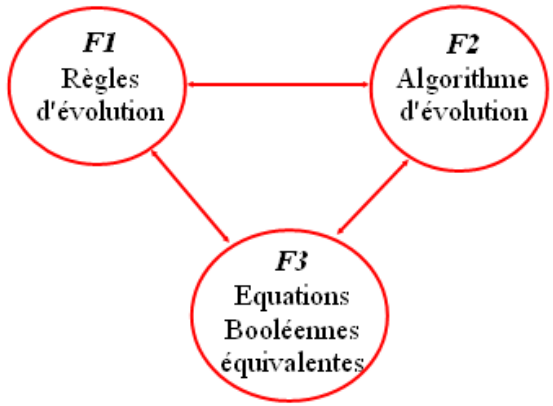
- La norme définit les symboles et les règles nécessaires à la représentation graphique de ce langage, ainsi que l'interprétation qui en est faite.

- *Les techniques de mise en œuvre (passage d'une spécification GRAFCET à une réalisation câblée et (ou) programmée) ne font pas partie du domaine d'application de cette norme.*

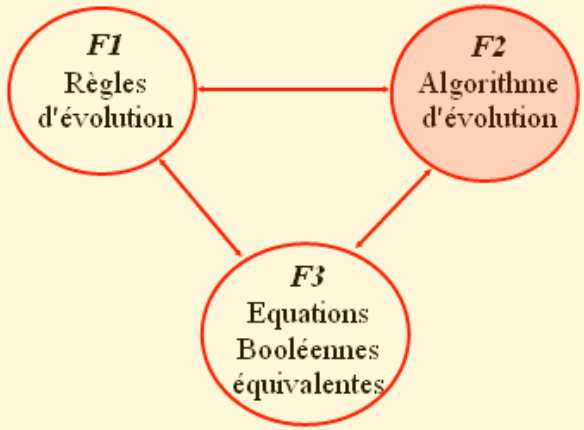
- Rmq : ds le cas des systèmes de commande intégrant un automate programmable, la norme CEI 61131-3 (1993) définie un ensemble de langages de programmation destinés aux automates programmables.

Implantation du grafcet

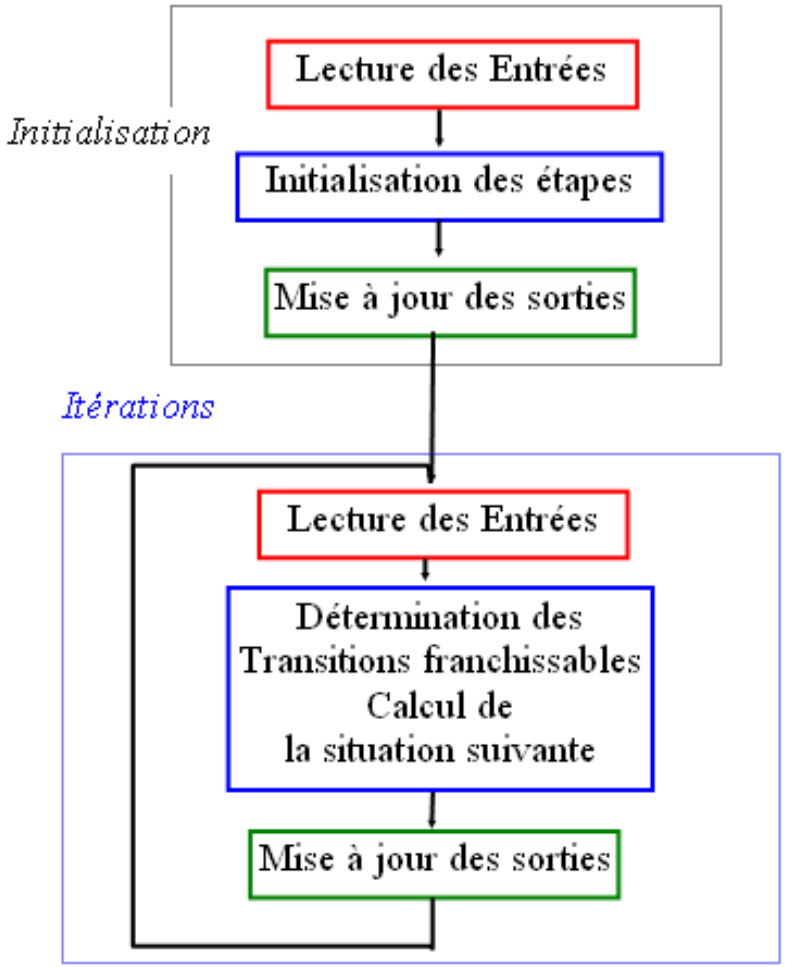
- Description du comportement du grafcet = **règles d'évolution** => modèle comportementale.
- Pour l'implantation, on a besoin d'une formalisation supplémentaire :
  - Soit **algorithme d'évolution**
  - Soit **équation booléennes équivalentes (équations logiques)**
- Ces 3 représentations sont équivalentes (même comportement du système vis-à-vis des E/S)



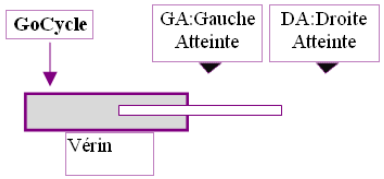
Le Grafcet – G7



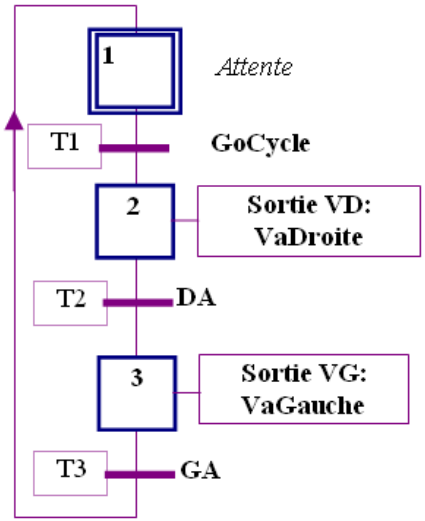
Algorithme d'évolution

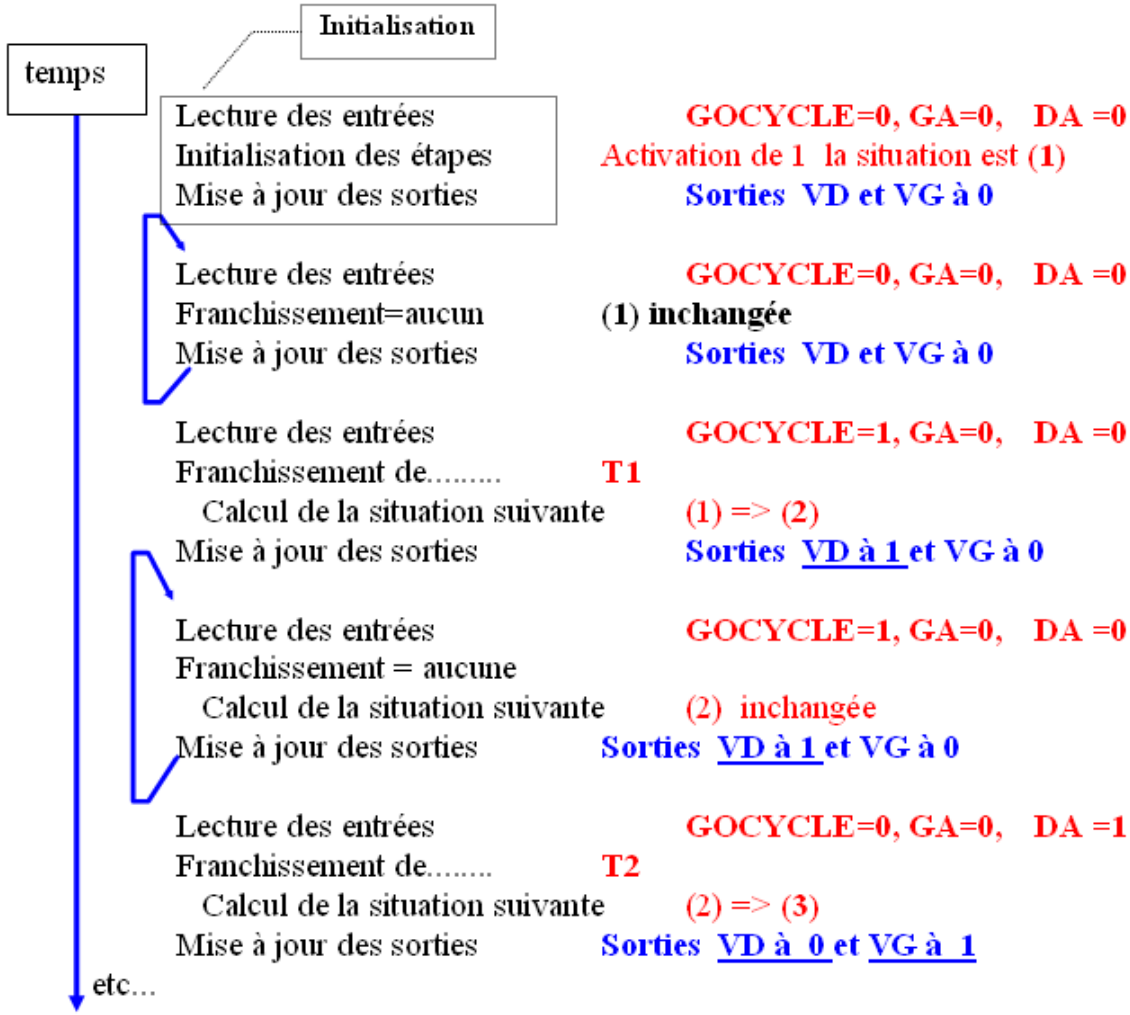


Algorithme d'évolution



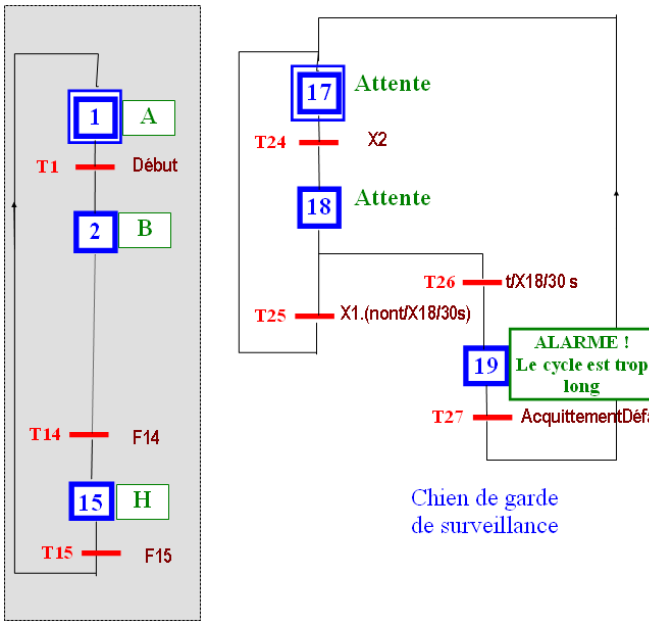
- Exemple :





Algorithme d'évolution

- Exemple :



Déroulement dans le temps

-Lecture des entrées -Initialisation des étapes -Mise à jour des sorties	**Début =0** Activation de 1 et 17 => **(1,17)** **Sortie A à 1**
-Lecture des entrées -Franchissement = aucun -Mise à jour des sorties	Début =0 **(1,17) inchangée** **Sortie A à 1**
-Lecture des entrées -Franchissement de ..... -Calcul de la situation suivante -Mise à jour des sorties	**Début =1** **T1** (1,17) => **(2,17)** **Sortie A à 0, Sortie de B à 1**
-Lecture des entrées -Franchissement de..... -Calcul de la situation suivante -Mise à jour des sorties	Début =1 **T24 car X2=1** (2,17) => **(2,18)** **Sortie de B à 1**
-Lecture des entrées -Franchissement de..... -Calcul de la situation suivante -Mise à jour des sorties	Début =1 rien (timer t < 30s) **(2,18) inchangée** **Sortie de B à 1**
-Lecture des entrées -Franchissement de..... -Calcul de la situation suivante -Mise à jour des sorties	**F14=1** **T14 et T26 (car F14 et timer=30s)** (2,18) => **(15,19)** **Sortie de H à 1, Sortie Alarme à 1**

etc...

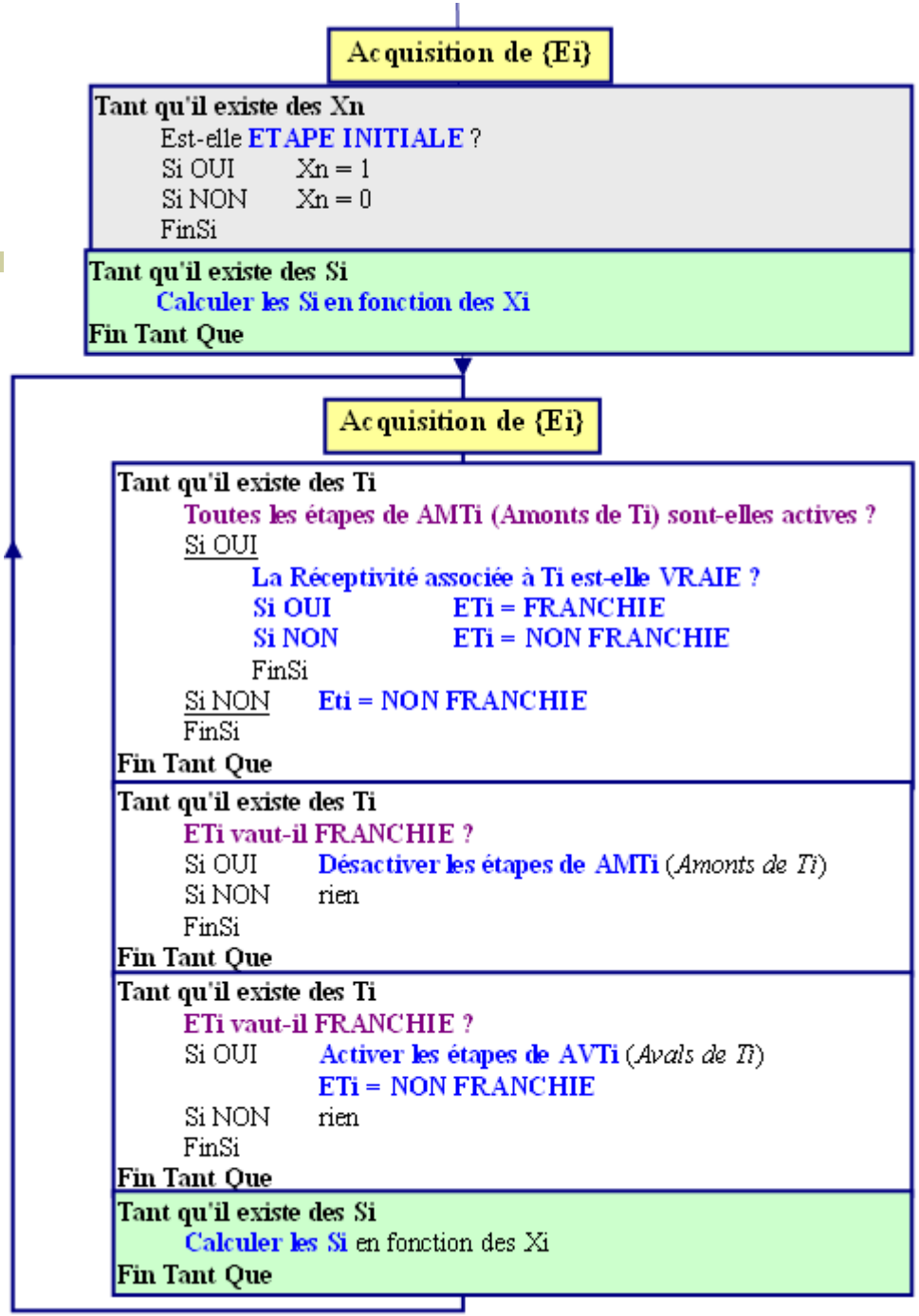
Algorithme d'évolution

- **IMPLANTATION INFORMATIQUE** : plusieurs façons selon des critères d'optimisation (taille mémoire, tps d'exécution..)

Exemple simple :

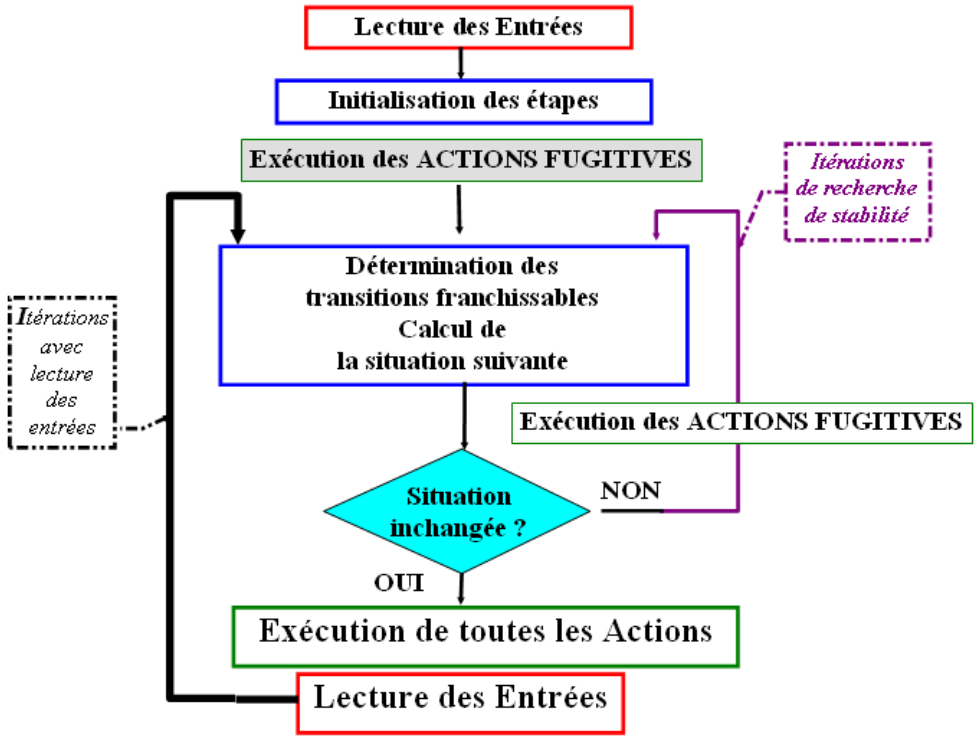
Données	*Paramètre. ou Propriété*	*Paramètre. ou Propriété*	*Paramètre. ou Propriété*	*Paramètre. ou Propriété*	*Paramètre. ou Propriété*
Entrée	**Nom**	**Valeur**	**N° Connex.**	**1 = vert 0 =rouge**	
Sortie	**Nom**	**Valeur**	**N° Connex.**	**1 = vert 0 =rouge**	
Etapes	**Nom**	**Etat**	**Initiale ?**	**Liste Trans. Amont**	**Liste Trans. Aval**
Transition	**Nom**	**Franchissable ?**	**Réceptivité**	**Liste Etapes Amont**	**Liste Etapes Aval**
Réceptivité	**Nom**	**Valeur**	**Fonction**		

- **Algorithme** :


Acquisition de {Ei}
Tant qu'il existe des Xn
Est-elle ETAPE INITIALE ?
Si OUI Xn = 1
Si NON Xn = 0
FinSi
Tant qu'il existe des Si
Calculer les Si en fonction des Xi
Fin Tant Que
Acquisition de {Ei}
Tant qu'il existe des Ti
Toutes les étapes de AMTi (Amonts de Ti) sont-elles actives ?
Si OUI
La Réceptivité associée à Ti est-elle VRAIE ?
Si OUI ETi = FRANCHIE
Si NON ETi = NON FRANCHIE
FinSi
Si NON Eti = NON FRANCHIE
FinSi
Fin Tant Que
Tant qu'il existe des Ti
ETi vaut-il FRANCHIE ?
Si OUI Désactiver les étapes de AMTi (Amonts de Ti)
Si NON rien
FinSi
Fin Tant Que
Tant qu'il existe des Ti
ETi vaut-il FRANCHIE ?
Si OUI Activer les étapes de AVTi (Avals de Ti)
ETi = NON FRANCHIE
Si NON rien
FinSi
Fin Tant Que
Tant qu'il existe des Si
Calculer les Si en fonction des Xi
Fin Tant Que

Algorithme d'évolution

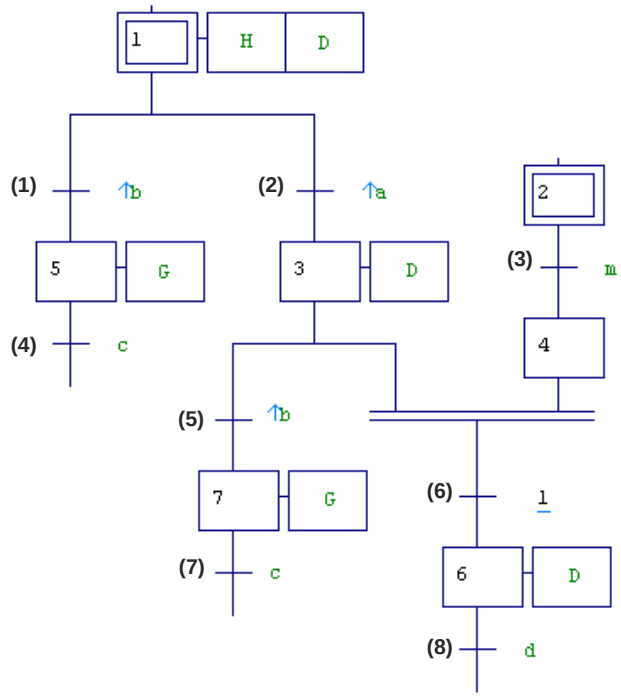
- **Variantes** :
  - Algorithme sans recherche de stabilité : le précédent; (nom : SRS)
  - Algorithme avec recherche de stabilité (nom : ARS) : de nouvelles entrées ne sont lues que lorsque le grafcet a atteint un état stable.



Algorithme d'évolution

- Intérêt de la recherche de stabilité

- Exemple : déplacement d'un mobile
- Fonctionnement :
  - Déplacements initiaux : *H* et *D*
  - Si atteinte de *b* avant *a* : le mobile repart à gauche jusqu'à *c*
  - Si atteinte de *a* avant *b*, et le bouton poussoir *m* enclenché : continu d'aller à droite jusqu'à *d* (ne monte plus)
  - Si atteinte de *a* avant *b*, et *m* non enclenché : va à droite jusqu'à *b*, puis revient à gauche jusqu'à *c*.



Algorithme d'évolution

- Mise en œuvre SANS stabilité :
  1) Lecture des entrées
  2) Evolution de la situation (franchissement d'1 ou plusieurs transitions simultanées)
  3) Exécutions des actions
- Exemple de scénario d'entrée : capteur m, puis a, puis b rapidement après a
- Comportement : (situation initiale {1,2})
  - Lecture de m
  - Evolution {1,2} →{1,4}
  - Action : H,D
  - Lecture de ↑a
  - Evolution {1,4} →{3,4}
  - Action : D
  - Lecture de ↑ b
  - Transitions franchissables : n°5 ET n°6

  ⇒ simultanément franchies, **pas très correct a priori** (branches censées être exclusives)
  - **Actions : G et D => confirmation du problème !!**

Rmq : si b arrive + tard, la transition n°6 sera franchie avant la n°5 => comportement non déterministe, ce qui n'est pas dans l'esprit du Grafcet

Algorithme d'évolution

- Une solution : assurer **explicitement** l'exclusivité des branches d'une structure de choix, ou éviter les situations instables (difficile, et grafcet souvent limité).

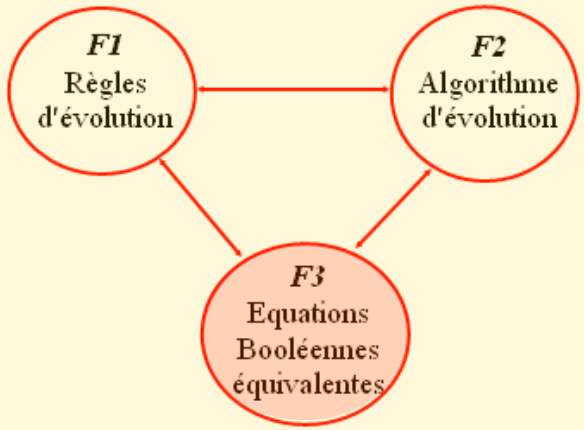
- Autre solution : **Mise en œuvre AVEC stabilité**
  1) Lecture des entrées
  2) Evolution de la situation jusqu'à atteinte d'une situation stable
  3) Exécutions des actions

- Sauf cas particulier, **la recherche de stabilité est indispensable à la mise en œuvre correcte d'un grafcet**.

Algorithme d'évolution

- **Optimisations** :
  - Diminution du nb de transitions à explorer : au lieu d'explorer toutes les transitions, on sélectionne les transitions aval des étapes actives.
  - Diminution du nb de transitions à explorer : au lieu d'explorer toutes les transitions, on ne sélectionne que les transitions associées aux entrées qui ont changées.
  - Rmq : ces optimisations sont + ou – efficaces suivants l'implémentation mémoire des données (listes, chaînées ou doublement chaînées, tableaux, etc..)

Le Grafcet – G7



Équations équivalentes

Equation logique d'une étape

- Principe : on considère une étape Xp et son environnement.
- Une étape p est active (Xp vaut 1) si :
  - elle est activée par l'amont
  - elle valait déjà 1 et n'est pas désactivée par l'aval

$$Xp(t_{n+1}) = ConditionActivation + Xp(t_n) \times (\overline{ConditionDesactivation})$$

Équations équivalentes

Equation logique d'une étape

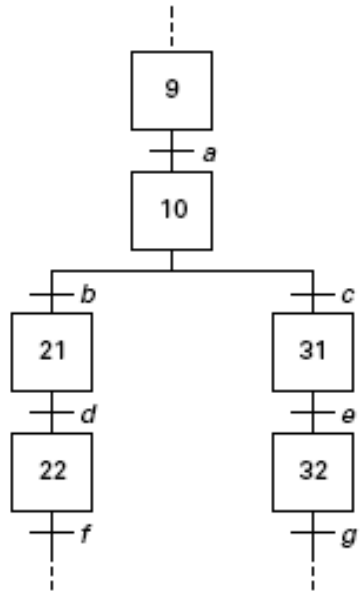
- **Initialisation** : activation des étapes initiales, désactivation des autres :
  - Init(Xp) = 1 à t_0 si Xp est une étape initiale, 0 sinon.

$$Xp(t_{n+1}) = Init(Xp) + ConditionActivation + Xp(t_n) \times (\overline{ConditionDesactivation})$$

Équations équivalentes

- Exemple : **début de choix de séquences**

Étape	CAX*i*	CDX*i*
10	X9 · *a*	***b+c***
21	X10 · *b*	***d***
31	X10 · *c*	***e***

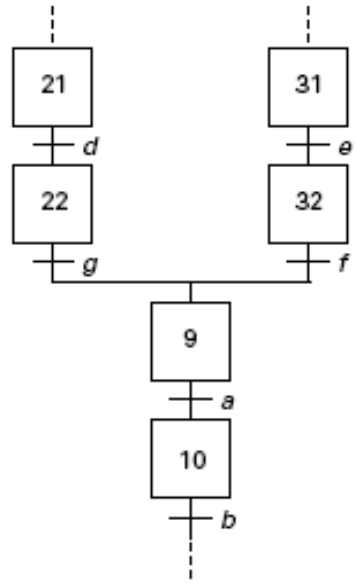


$$X10 = X9.a + X10.(\overline{b + c})$$
$$X21 = X10.b + X21.\overline{d}$$
$$X31 = X10.c + X31.\overline{e}$$

Équations équivalentes

- Exemple : **fin de choix de séquences**

Étape	CAX*i*	CDX*i*
9	$X22 \cdot g + X32 \cdot f$	***a***
22	$X21 \cdot d$	***g***
32	$X31 \cdot e$	***f***

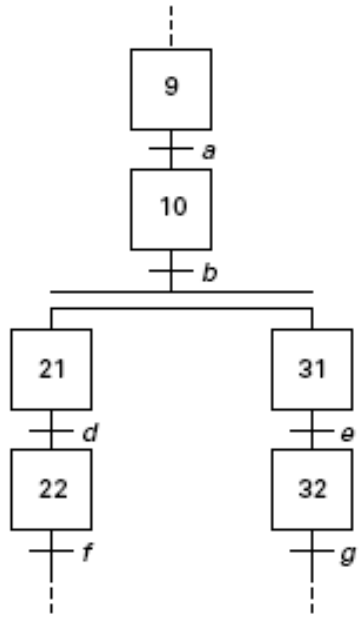


$$X9 = (X22.g + X32.f) + X9.\overline{a}$$
$$X22 = X21.d + X22.\overline{g}$$
$$X32 = X31.e + X32.\overline{f}$$

Équations équivalentes

- Exemple : **début de séquences parallèles**

Étape	CAX*i*	CDX*i*
10	X9 · *a*	***b***
21	X10 · *b*	***d***
31	X10 · *b*	***e***

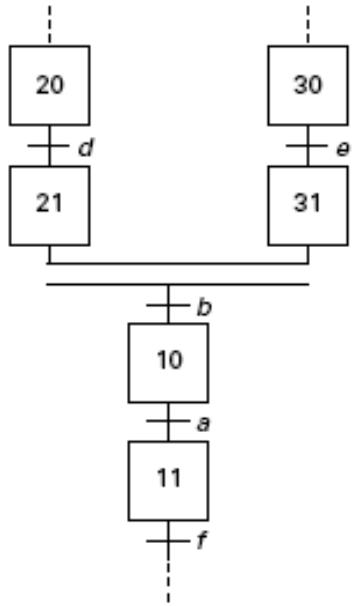


$$X10 = X9.a + X10.\overline{b}$$
$$X21 = X10.b + X21.\overline{d}$$
$$X31 = X10.c + X21.\overline{e}$$

Équations équivalentes

- Exemple : **fin de séquences parallèles**

Étape	CAX*i*	CDX*i*
10	$X21 \cdot X31 \cdot b$	***a***
21	$X20 \cdot d$	**X31.*b***
31	**X30.*e***	**X21.*b***



$$X10 = X21.X31.b + X10.\overline{a}$$
$$X21 = X20.d + X21.\overline{X31.b}$$
$$X31 = X30.e + X31.\overline{X21.b}$$

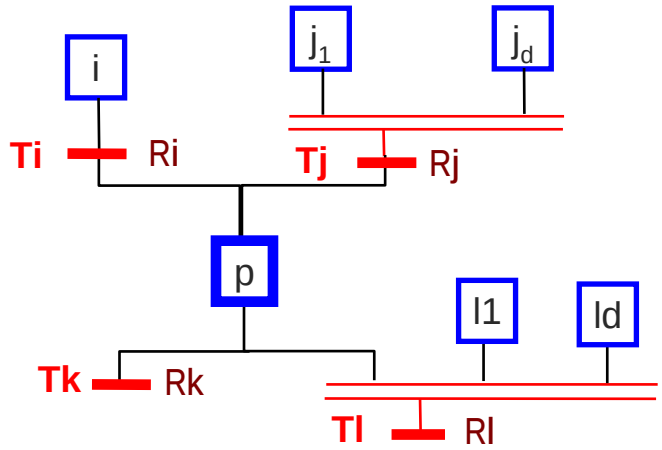
Équations équivalentes

- Conditions d'activation (CAXp) : *pour Xp à l'instant n+1*

$$\sum_{\forall i,j \text{ amont de } p} \left\{ \sum_{i} (X_i . R_i) + \sum_{j} \left[\prod_{ju} X_{j1} .. X_{jd} . R_j \right) \right\}$$

instant n

instant n+1



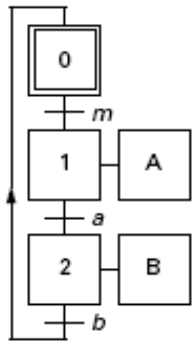
- Conditions de désactivation (CDXp) :

$$\sum_{\forall k,l \text{ aval de } p} \left\{ \sum_{k} (R_k) + \sum_{l} \left[\prod_{lv} X_{l1} .. X_{ld} . R_l \right) \right\}$$

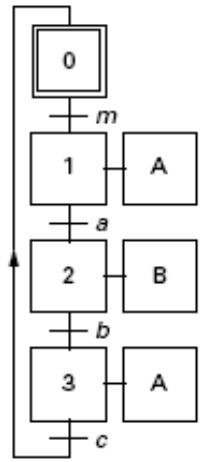
Équations équivalentes

Gestion des actions

- **Action continue** : son activité dépend de l'activité des étapes auxquelles elle est associée.
- Exemples :



- A = X1
- B = X2

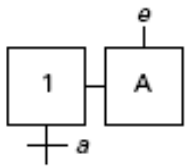


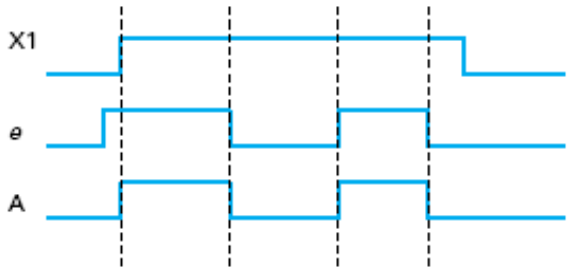
- A = X1 + X3
- B = X2

Équations équivalentes

Gestion des actions

- **Action conditionelle** : son activité dépend de l'activité des étapes auxquelles elle est associée ET des conditions.
- Exemple :





A = X1 . e

Équations équivalentes

- Gestion des arrêts d'urgence :
  - AU doux : stop des actions

$$A = Xp \,.\, \overline{AUd}$$

Signaux d'arrêt d'urgence

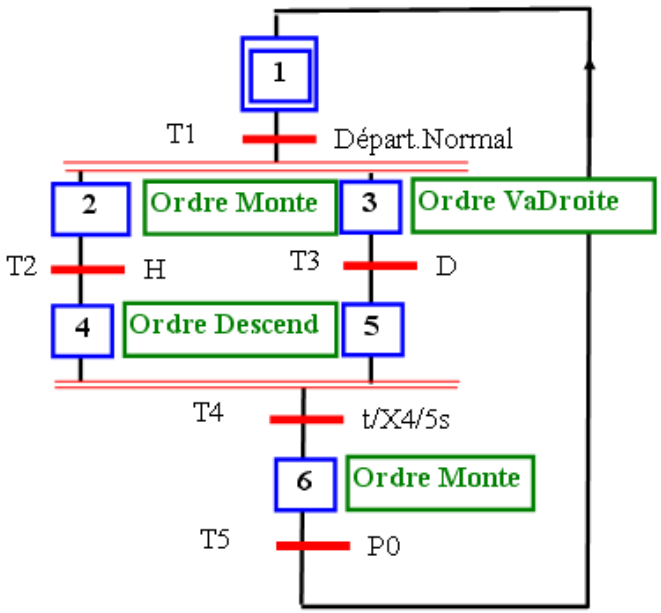
  - AU Dur : désactivation des étapes

$$Xp(t_{n+1}) = (Init(Xp) + CAXp + Xp(t_n) \times (\overline{CDXp})) \,.\, \overline{AUD}$$

Équations équivalentes

- Exemple 1 :

le grafcet



Les équations équivalentes

- $X_i(0) = 0 \ \forall i$
- **Init** = 1 à t=1

$$X_1(t_{n+1}) = (X6(t_n).P0(t_{n+1})) + Init + X1(t_n) \,.\, non(Depart(t_{n+1}).Normal(t_{n+1}))$$

$$X_2(t_{n+1}) = (X1(t_n).Départ(t_{n+1}).Normal(t_{n+1})) + X2(t_n) \,.\, non(H(t_{n+1}))$$

$$X_3(t_{n+1}) = (X1(t_n).Départ(t_{n+1}).Normal(t_{n+1})) + X3(t_n) \,.\, non(D(t_{n+1}))$$

$$X_4(t_{n+1}) = (X2(t_n).H(t_{n+1})) + X4(t_n) \,.\, non(t/X4/5s \,.\, X5(t_n))$$

$$X_5(t_{n+1}) = (X3(t_n).D(t_{n+1})) + X5(t_n) \,.\, non(t/X4/5s.\ X4(t_n))$$

$$X_6(t_{n+1}) = (X4(t_n).\ X5(t_n).\ t/X4/5s) + X6(t_n) \,.\, non(P0(t_{n+1}))$$

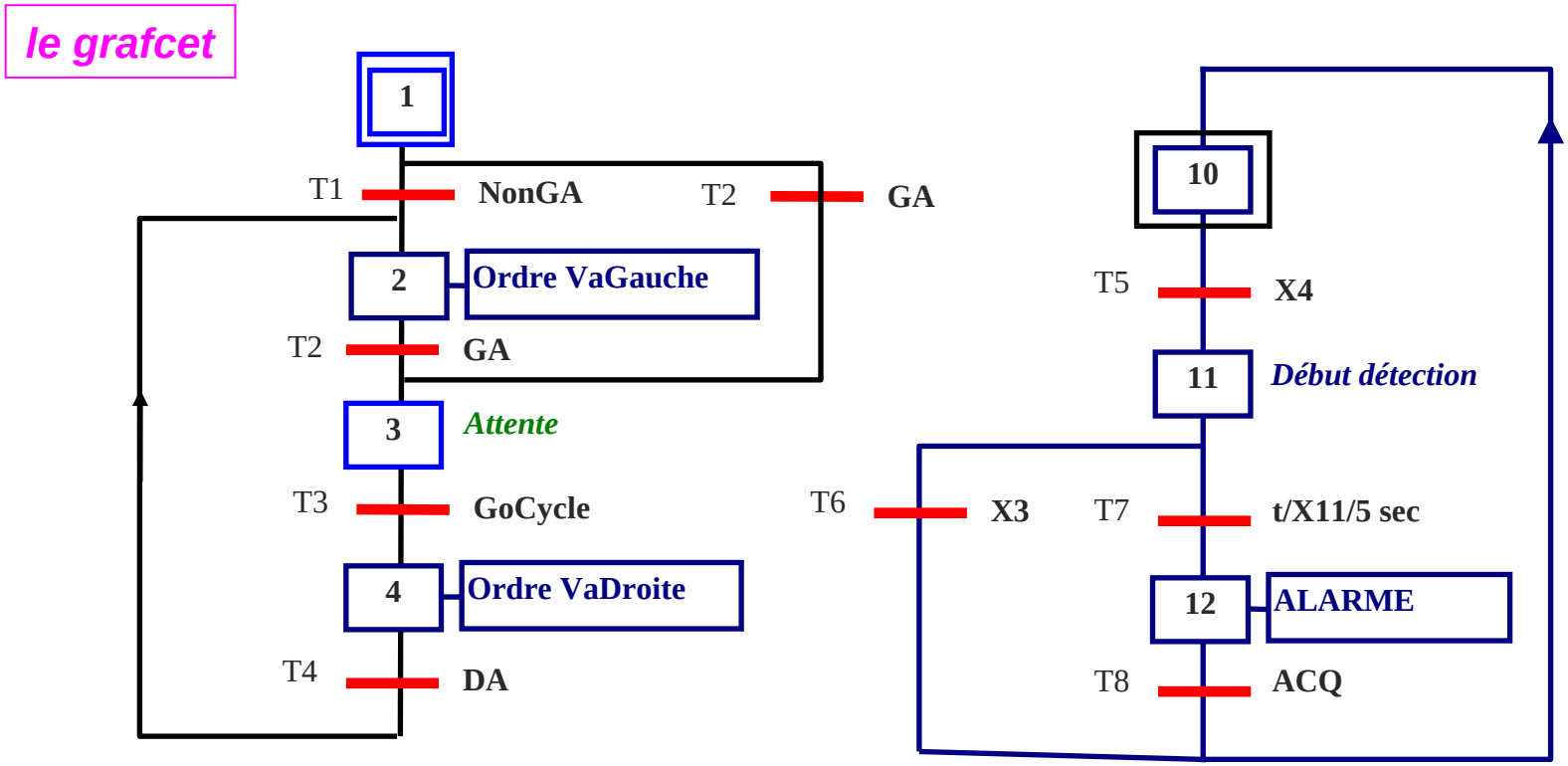
$$Monte = X_2(t_{n+1}) + X_6(t_{n+1})$$

$$VaDroite = X_3(t_{n+1})$$

$$Descend = X_4(t_{n+1})$$

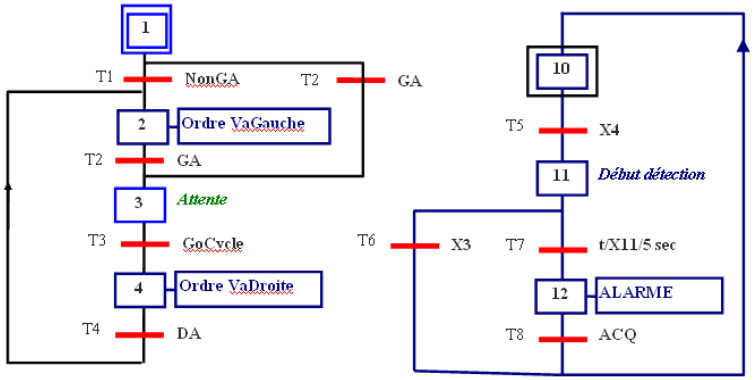
Équations équivalentes

- Exemple 2 : commande du cycle d'1 vérin avec initialisation et détect° de dépassement du tps



Équations équivalentes

le grafcet



Les équations équivalentes

$$X1 = Init + X1.\overline{(\overline{GA} + GA)}$$

$$X1 = Init + X1.\bar{1} = Init + X1.0$$

$$X1 = Init$$

$$X2 = X1.\overline{GA} + X4.DA + X2.\overline{GA}$$

$$X3 = X2.GA + X1.GA + X3.\overline{GoCycle}$$

$$X4 = X3.\overline{GoCycle} + X4.\overline{DA}$$

$$X10 = Init + X12.ACQ + X11.X3 + X10.\overline{X4}$$

$$X11 = X10.X4 + X11.(\overline{X3 + (t / X11 / 5\,sec)})$$

$$X12 = X11.(t / X11 / 5\,sec) + X12.\overline{ACQ}$$

Les équations de sorties

$$VaGauche = X2$$

$$VaDroite = X4$$

$$ALARME = X12$$

Conclusion

- Ces principes montrent comment passer d'un Grafcet à une réalisation algorithmique ou à un circuit.

- Ils s'appliquent quelque soit la technologie:
  - circuits logiques
  - programmation logique sur API
  - programmation classique sur PC industriels
  - etc...

Le Grafcet – G7

Quelle technologie ?

Mise en œuvre

- Le Grafcet n'impose aucune solution : le choix ne dépend que de critères économiques ou des conditions d'utilisation.

- Exemples :
  - Si besoin de modification, si bcp d'E/S ou si implémentation personnalisée : API, programmé directement en G7 par une console de programmation (et / ou réseau).
  - Si : langage simple (ET, OU, mémoires)
  - Si fonctionnement prédéfini et figé, et simple : câblage électronique (portes et bascules).
  - Si environnement peu compatible avec l'électronique : câblage non électronique (réalisation pneumatique ou électrique)
  - Si fonctionnement complexe avec besoin de rapidité ou de calcul numérique : carte µ-contrôleur ou même µ-processeur.

Mise en œuvre

- Approches de mise en œuvre :
  - Approche **câblée asynchrone**
    bascules RS
  - Approche **câblée synchrone**
    diagramme d'états, séquenceur à bascules D
  - Approche **câblée programmée**
    mémoire et bascules D
  - Approche **programmée**
    µcontrôleur et processeur booléen
  - **Automate Programmable**

+ flexible

Le Grafcet – G7

Programmation par logique câblée

Réalisation technologique

- Programmation en logique câblée :
  - Logique combinatoire
  - Logique séquentielle asynchrone
    - Bascule RS
  - Logique séquentielle synchrone
  - Séquenceur électrique
  - Séquenceur pneumatique

Le Grafcet – G7

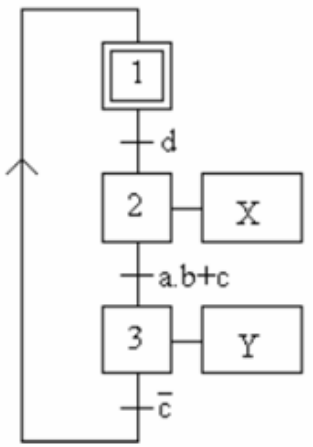
Exemple :

Câblage par bascules RS

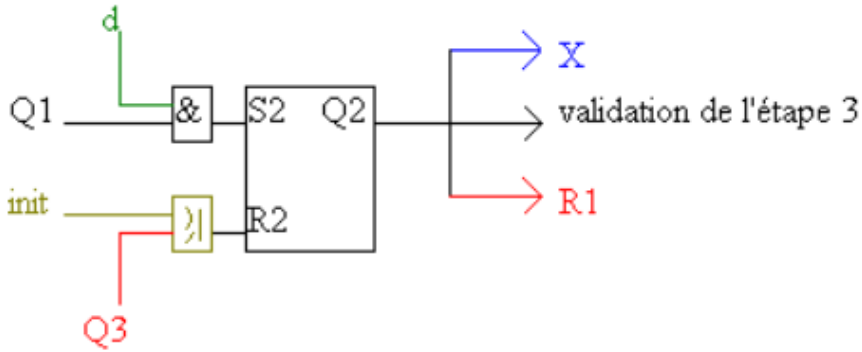
Câblage asynchrone

- Mise en œuvre par câblage asynchrone : cas simple d'un grafcet linéaire.
  - Une **bascule RS** par étape
  - Une étape s'active si son étape amont est active + réceptivité vraie
  - Une étape se désactive qd la suivante est active

Etape i :
- ***Si : Set***
- ***Ri : Reset***
- ***Qi : Sortie***

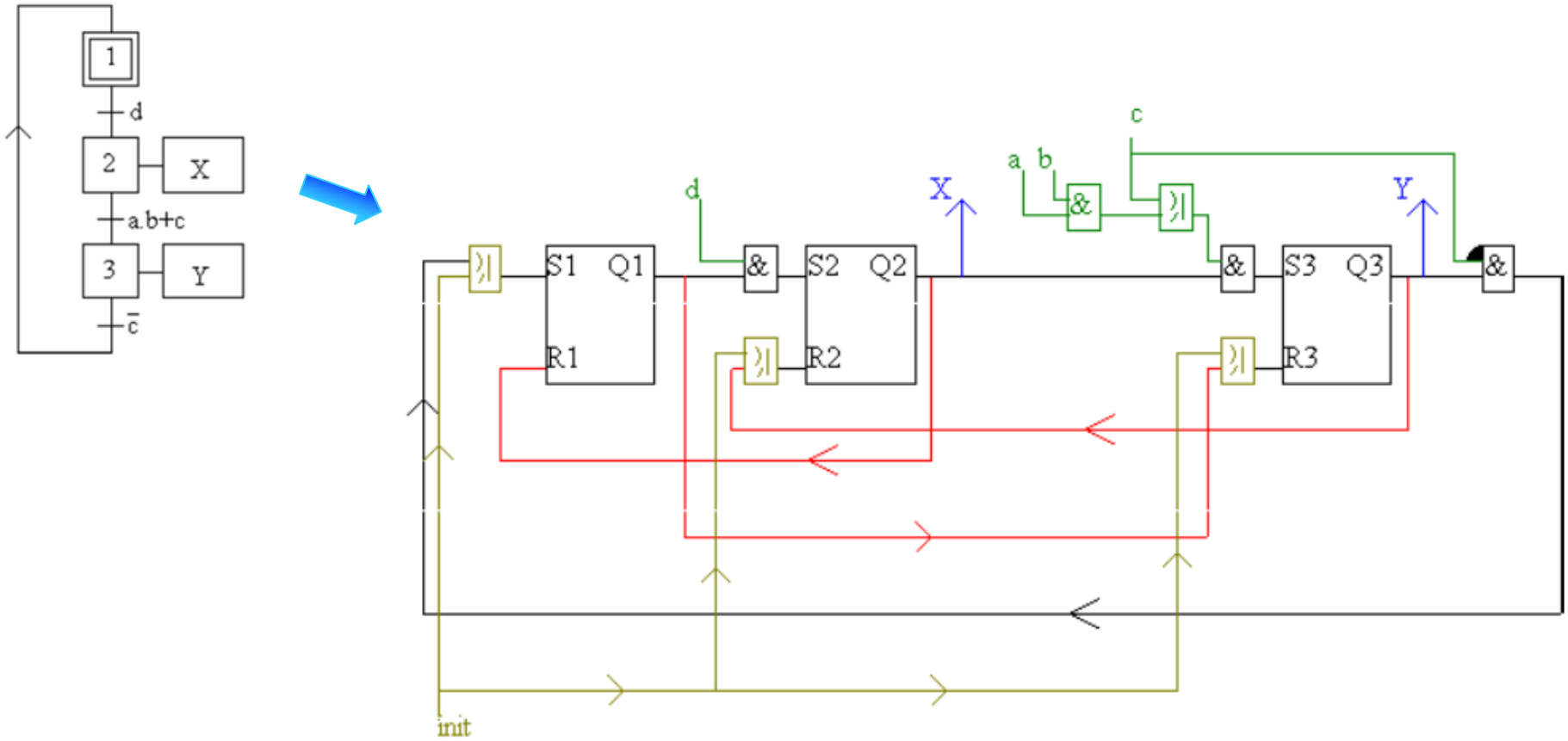


- Etape 2 :
  - Activation : S2 = Q1.d
  - Sortie X : X=Q2
  - Désactivation : R2=Q3



Câblage asynchrone

- Schéma du câblage du système complet :



Câblage asynchrone

- Principale difficulté : le **routage**, surtout pour un circuit imprimé (croisements de pistes impossibles). D'autant plus que chaque composant doit être **alimenté** (alimentations non représentées ici). Mais il existe désormais de bons logiciels de routage.

- La mise en œuvre d'un Grafcet par câblage n'est pas très compliquée, par contre la **modification est difficile** (souvent, nouveau câblage si modif du G7). De même, la recherche d'erreurs après coup étant difficile => test du câblage dès sa réalisation.

- Autre problème de cette méthode :
  - une étape active désactive son étape amont en permanence, tant qu'elle reste active (au lieu de ne le faire qu'au moment de la transition).
  - ⇒ L'étape amont ne peut alors pas être activée par un autre signal (ex : bouton "init")

- Solution : mémoriser par une bascule intermédiaire l'état des transitions.
  - trop complexe !
  - reste des difficultés liées aux tps de réponse ≠ des bascules

⇒ **câblage synchrone (bascule D), …**

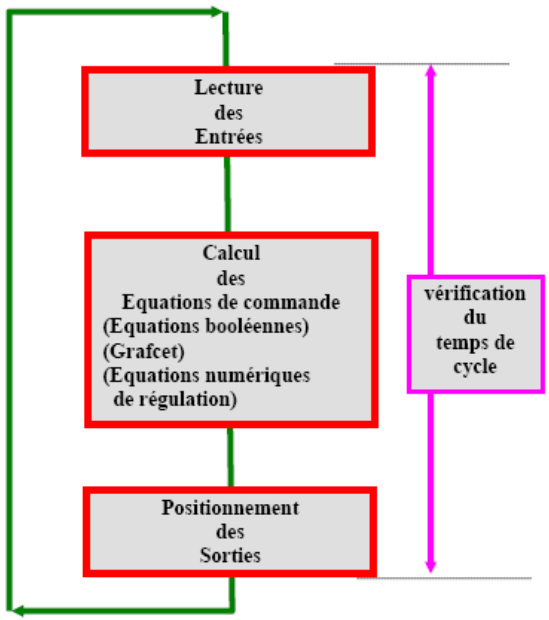
Le Grafcet – G7

Automate Programmable Industriel (API)

Automate Programmable Industriel

- Caractéristiques essentielles : adapté pour le contrôle de procédé
  - Un ordinateur durci : ambiance industrielle
  - Une connectique adaptée : connexion rapide
  - Un système d'exploitation adapté : fonctionnement préformé
  - Programme et exploitation adaptés : formalisme proche de la définition des équations de commande.

- Cycle de base :

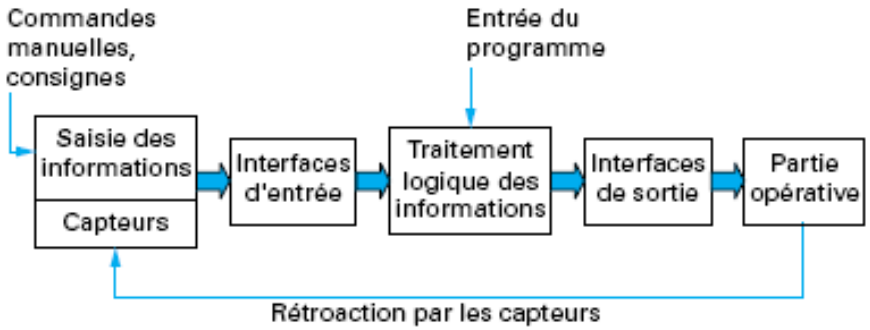


E/S

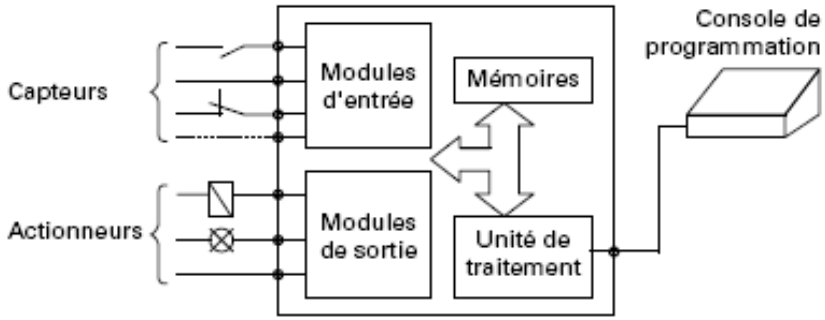
modularité

Automate Programmable Industriel

- Structure des systèmes automatiques :



- Principe de fonctionnement d'un API :



Automate Programmable Industriel

- *si machine = µ-processeur*
  ⇒ spécifique au µ-processeur
- *si machine = machine logique spécialisée*
  ⇒ proche du langage booléen

Langages des API

le LANGAGE MACHINE

le plus rudimentaire

```
In   A   1002
Sto  A,  5000
LdA  A   4000
Out  A   1002
```

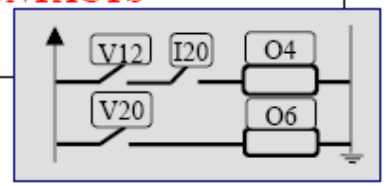
LANGAGE INTERPRETEUR BOOLEEN

assez simple, mais un peu archaïque

```
OU   In12
ET   M20
LdA  A
Out  A
```

LANGAGE A CONTACTS

Trés utilisé



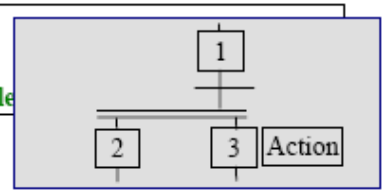
EXPRESSIONS BOOLEENNES

Assez utilisé

```
V12 = I34 + I25
V3  = I10 . I24 . I45
010 =  (I12.V12)+(I25)
```

LE GRAFCET

Le plus évolué pour la partie séquentielle



Automate Programmable Industriel

Programmation en langage booléen

- Fonctions combinatoires élémentaires :
  - ET logique : entre la variable et le résultat précédent,
  - OU logique : entre la variable et le résultat précédent,
  - LIRE : l'état de la variable indiquée,
  - RANGER : le résultat dans la variable indiquée,
  - NON : inverser l'état de la variable sélectionnée

Ex : langage STEP5 sur une console PG605 pour les automates de types SIMATIC S5 Siemens

- Pour chaque étape *i* :
  - Écrire sa condition d'activation *CAXi*
  - Écrire sa condition de désactivation *CDXi*
  - Écrire son action associée
  - Utilisation d'une variable interne associée *M0,i*

Programmation d'une étape

U « CAX*i* »	**(U : fonction ET)**
S M0, *i*	(mise à 1 : SET)
U « CDX*i* »	
R M0, *i*	(mise à 0 : RESET)

Automate Programmable Industriel

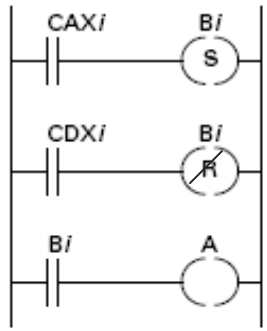
Programmation en langage à contacts (*ladder*)

Projet P5+P7

Ex : langage PL7-2 sur un terminal TSX-T407 pour les automates Télémécanique TSX 17-20/27/47-J/47-10/20

- Type d'éléments :
  - Entrées (contacts) : contact ouvert ou fermé
  - Sorties (bobines) : set et reset
  - Fonctions : AND, OR, XOR, NOT (NAND, NOR)

- Programmation d'une étape *i* avec action *A* :
  - Écrire sa condition d'activation *CAXi*
  - Écrire sa condition de désactivation *CDXi*
  - remplacement de chaque étape Xi par une variable interne de l'automate notée Bi ;
  - câblage de la CAXi sur le SET de Bi et de la CDXi sur le RESET de Bi;
  - câblage de l'action A associée à Xisur la variable interne Bi.



Automate Programmable Industriel

Programmation en expressions booléennes

- Langage proche de l'écriture directe et non du schéma.
- **Ligne** :

  Ln° : NomVariable = ExpressionBooleenne

- **Instruction de saut**
  - exemple : L20 : (E3+E5) = L35
    - ⇒ si (E3+E5) est vraie, le programme va ligne L35 (sinon L21)
  - utilisation :
    - Initialisation
    - Mode Panne ou Normal
    - Calcul en fonction d'une valeur de variable

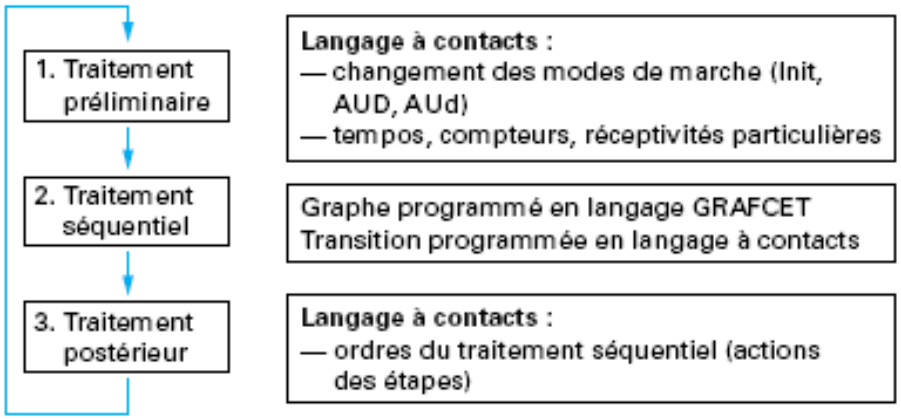
Automate Programmable Industriel

Programmation "directe" en GRAFCET

Projet P5+P7

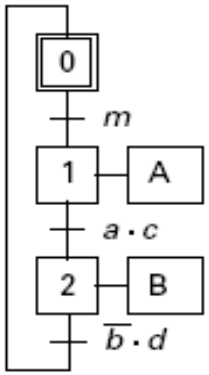
Ex : automate Télémécanique TSX 47 avec le langage GRAFCET PL7-2 sur un terminal TSX-T407

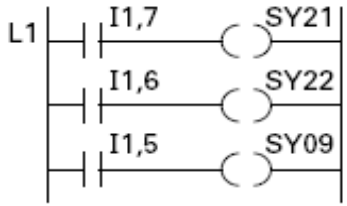
- Programmation structurée en 3 parties :



Automate Programmable Industriel

Programmation "directe" en GRAFCET
Exemple



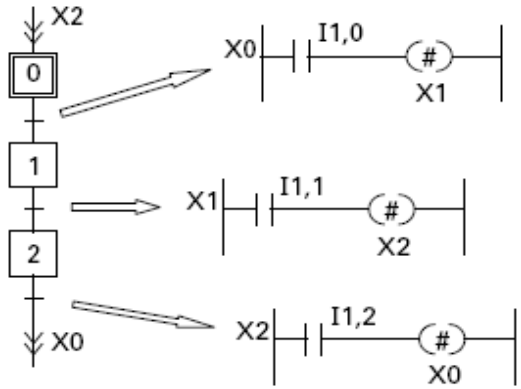


(a) traitement préliminaire

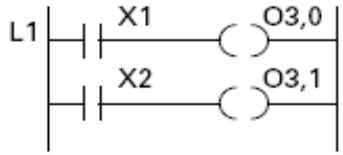
m = I1,0

a = I2,0 b = I2,1
c = I2,2 d = I2,3

A = O3,1 B = O3,2



(b) traitement séquentiel



(c) traitement postérieur

Bits systèmes de l'automate :
SY21=I1,7 : init du G7
SY22=I1,6 : RAZ du G7 (AUD)
SY09=I1,5 : mise à 0 des Out (Aud)

Automate Programmable Industriel

- **Programmation des API : aléas**
  - Il existe une différence entre l'exécution du système d'équations séquentielles théoriques et l'exécution d'une suite d'équations sur un API.
  - Dans un API, l'état final atteint peut dépendre de l'ordre des équations / instructions / ladder.
  - Pour éviter cela : si modification d'une variable pendant un cycle, ne pas faire les calculs restant avec la nouvelle valeur mais avec celle de début du cycle (utiliser une variable temp).

- Exemple :

```
a  = b+c        d = a
d = a      ≠    a  = b+c
```

```
temp = a
a  = b+c        ⇔    d = a
d = temp             a  = b+c
```

Automate Programmable Industriel

Programmation en langage évolué

- Le programme est également structuré de façon séquentielle (comme pour l'API) afin de tenir compte du traitement séquentiel des informations du système automatique.
- Implémentation de **l'algorithme d'évolution.**
- Ex : Pascal, C.