Chapitre 2 : Fondements théorique des SED :
THEORIE DES AUTOMATES A ETATS FINIS :

En théorie des langages : systèmes de transitions	**En théorie des automates : diagrammes d'états**
Reconnaissance d'expressions régulières - un état correspond à une étape dans la reconnaissance d'une expression - une transition correspond à un pas de plus dans la reconnaissance d'une expression - une étiquette correspond à un symbole du langage de référence	Modélisation de SED - un état correspond à une étape dans la vie d'un système - une transition correspond à un changement dans la vie d'un système - une étiquette correspond à un événement survenant dans la vie d'un système

I. Généralité sur les langages et les automates :

Base de la théorie des langages et des automates :

- un **alphabet Σ** est un ensemble fini non vide de symboles
- un **mot w** défini sur Σ est une séquence finie de symboles
- un **langage L** défini sur Σ est un sous-ensemble de Σ* (Σ* : l'ensemble de tous les mots que l'on peut construire sur Σ).

exemples d'alphabet :
- {0,1}
- {a,b}
- {A,...,Z}

exemples de mots :
- 101100 est un mot sur {0,1}
- abba est un mot sur {a,b}
- BONJOUR est un mot sur {A,...,Z}

exemples de langages :
- {000,001,010,011,100,101,110,111}
- {ba,baba,babababa,...}
- {R,UR,UUR,UUUR,...}

Longueur d'un mot :

- la longueur d'un mot w, notée |w|, est le nombre de symboles qui le composent
- Soit l'alphabet Σ, on note Σ^n l'ensemble des mots de longueur n
- Soit Σ* l'ensemble de tous les mots que l'on peut construire sur Σ :

$$\Sigma^* = \bigcup_{\{i \geq 0\}} \Sigma^i$$

Opérations sur les langages :

Soient L, L_1, L_2 définis sur Σ,
- union : $(L_1 \cup L_2) = \{w \in \Sigma^*, w \in L_1 \text{ ou } w \in L_2\}$
- intersection $(L_1 \cap L_2) = \{w \in \Sigma^*, w \in L_1 \text{ et } w \in L_2\}$
- concaténation (produit) : $L_1.L_2 = \{w \in \Sigma^*, \exists(w_1 \in L_1), \exists(w_2 \in L_2) \text{ et } w = w_1.w_2\}$

- fermeture itérative : $L^* = L^0 \cup L^1 \cup ... = \cup_{\{i \geq 0\}} L^i$
  $= \{w \in \Sigma^*; \exists n \geq 0, \exists (w_1, w_2, ... w_n), w = w_1.w_2...w_n\}$
- associativité
- distributivité
- autres (ε : mot vide, $\emptyset$: langage vide)

Un langage régulier sur un alphabet Σ est soit
- $\emptyset$ un langage vide
  $\{\varepsilon\}$ (langage réduit au mot vide)
- $\{x\}$ pour n'importe quel symbole $x \in \Sigma$
- un langage quelconque (sous ensemble de Σ^*) obtenu par des opérations d'union, de concaténation et de fermeture itérative

Les expressions régulières servent à représenter les langages réguliers :
- $\emptyset$, ε et tout symbole $x \in \Sigma$ sont des expressions régulières sur Σ
- si α et β sont des expressions régulières sur Σ alors $\alpha + \beta$, $\alpha . \beta$, $(\alpha)^*$ et $(\beta)^*$ sont des expressions régulières sur Σ
- héritent des propriétés des langages

Expressions régulières :

- Formalisme de description des langages mais par énumération exhaustive de ses composants
- Énumération irréalisable pour des ensembles infinis!
- Diagrammes ETAT/TRANSITION : formalisme plus puissant
  - automates à états finis
  - automates étendus : SDL, Statecharts,...

II. Automates à états Finis (AeF) :

- Machine abstraite :
  - qui sait reconnaître l'appartenance ou non d'un mot à un langage donné;
  - qui sait décrire les évolutions possibles d'un système

Composée de :

- un ensemble fini d'états dont un initialisation
- une relation de transition : décrit les règles de passage d'un état à un autre

Définition formelle d'un AeF

Un AeF est défini par un quintuplet (Σ, Q, q_0, F et δ) tel que :
- Σ : alphabet
  Q : ensemble fini d'états
- $q_0 \in Q$: l'état initial
- $F \subseteq Q$: ensemble des états finaux (états de satisfaction ou états marqués)
- $\delta : Q \times \Sigma \rightarrow Q$: relation de transition (exprime le passage entre deux états)

Représentation d'un AeF :

On associe à un AeF (Σ, Q, q_0, F et δ) un graphe orienté et étiqueté G :

- les sommets S sont exactement les états de Q
- à chaque triplet (q, a, q') de δ, on associe un arc (q, q') étiqueté par a
- on représente différemment l'état initial et les états finaux

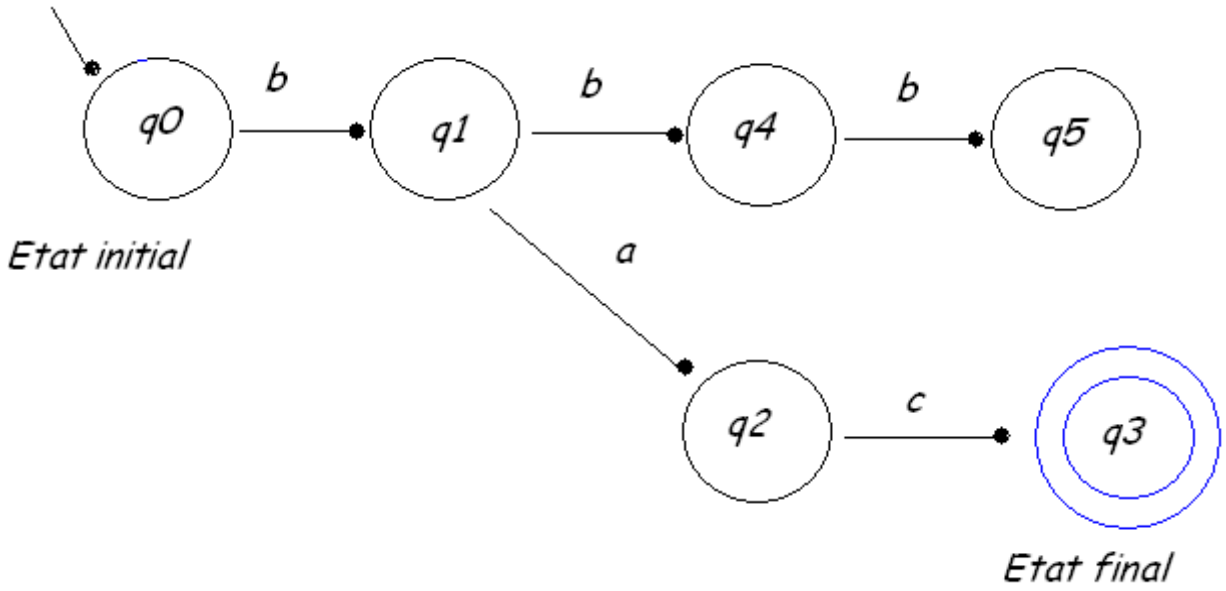
Langage d'un AeF :

- l'ensemble des mots engendrés par un AeF A forme le langage (régulier) de l'AeF A, noté L(A)
- l'ensemble des mots engendrés par un AeF A pour atteindre un ou plusieurs états finaux forme le langage reconnu (ou marqué) de l'AeF A, noté $L_m(A)$
- un AeF engendre un mot m s'il existe un chemin étiqueté par le mot m :
  - qui mène de l'état initial à un état quelconque (L(A))
  - qui suit des arcs étiquetés
  - La concaténation des étiquettes des arcs donne le mot m

États d'un AeF :

- un état $q_j \in Q$ est accessible s'il existe un mot $m \in \Sigma^*$ tel que $q_j = \delta(q_0, m)$
- un état $q_i \in Q$ est co-accessible s'il existe un mot $m \in \Sigma^*$ tel que $q_j = \delta(q_i, m)$ et $q_j \in F$
- un AeF est dit non-bloquant lorsque tout état accessible est co-accessible

Représentation de l'automate A :



Description formelle :

A=(Σ, Q, q_0, F et δ)
Σ={a,b,c}
Q={ q_0, q1 , q2, q3, q4, q5 }
q_0 est l'état initial
F={q3}
δ={(q_0 ,b,q_1),(q_1,a,q_2),(q_2 ,c,q_3),(q_1,b,q_4),(q_4 ,b,q_5)}

Langages :

$L(A) = \varepsilon + b + bb + bbb + ba + bac$
$L_m(A) = bac$

Etats accessibles : q0, q1, q2, q3, q4, q5
Etats co-accessibles : q0, q1, q2, q3

Aef Déterministe/AeF Non Déterministe

Aef est déterministe si δ est une fonction plutôt qu'une relation de transition

- AeFND : δ est une relation de transition → à partir d'un état et pour un symbole donné
  - plusieurs états sont atteignables
  - le choix entre plusieurs chemins est équiprobable
- AeFG : δ est une fonction de transition → à partir d'un état et pour un symbole donnée, il y a 1 unique état atteignable

AeF complet

Un AeF est complet si et seulement si et seulement si δ est une relation de transition totale et non partielle sur Q x Σ :

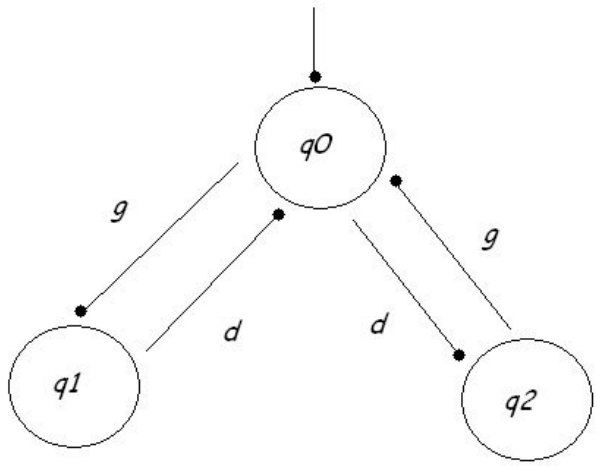
- de chaque état et pour un symbole donné, il y a au moins un état d'arrivée.

Un AeFD est complet si de chaque état et pour un symbole donné, il y a exactement un état d'arrivée.

Table de transition : Elle permet de décrire la relation de transition δ d'un AeF

Q \ Σ	a_1	...	a_j	...	a_n
q_1					
...					
q_i			$\delta(q_i, a_j)$		
q_n					

Exemple:



- δ:
  - δ(q0,d) = q2
  - δ(q0,d) = q1
  - δ(q1,d) = q0
  - δ(q2,d) = q0

δ	d	g
q0	q2	q1
q1	q0	-
q2	-	q0

- δ*:
  - δ*(q0, g.d) = q0
  - δ*(q1, d.d.g) = q0

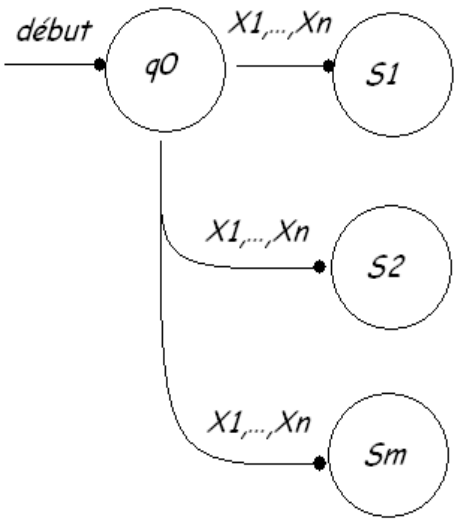
Déterminisation des AeF : (Élimination du non déterminisme)

Théorème : à tout AeFND correspond un AeFD équivalent et réciproquement.

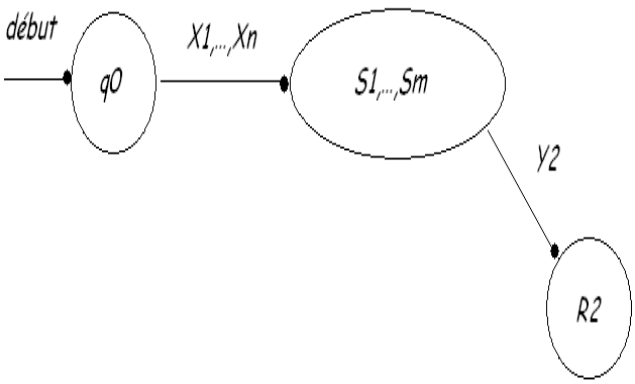
Définition : Deux AeF A_1 et A_2 sont équivalents si et seulement si ils acceptent le même langage $L_m(A_1) = L_m(A_2)$

Idée de la déterminisation

AeFND



1. Nouvel état copie de tous les états accessibles $S_1, S_2, \ldots S_m$.



2. Etats accessibles de $S_1, \ldots, S_m$:
   états accessibles de S_1, de S_2, ... et de S_m.

Algorithme de déterminisation

A partir d'un automate A = (Σ, Q, q_0, F et δ) non déterministe, on construit un nouvel automate A' déterministe

- de même alphabet Σ
- nouveaux états Q' = ensembles d'anciens états de Q (copies d'états)
- état initial q_0' = état q_0
- états finaux F' = états contenant un ancien état final de Q
- fonction de transitions δ ' = ensemble des états accessibles depuis chaque copie d'états.

Exemple : AeFND A = (Σ, Q, q_0, F et δ)

Σ = {0,1}
Q = {q_0 ,q_1}
F = {q_1}

Déterminisation de A
AeFD A' = (Σ, Q', q_0', F' et δ ')

δ	0	1
---> q_0	q_0 , q_1	q_0
* q_1	-	-

Donne :

δ	0	1
---> q_0	q_0q_1	q_0
* q_0q_1	q_0q_1	q_0

$q_0 = q_0$' et Q' = {q_0,q_0q_1}

III. Extensions des AeF

Plusieurs extensions ont été proposées :

extensions transitions → gardes
extensions transitions → actions
extensions transitions → fonctions de sorties
extensions états → actions
extensions états → fonctions de sorties
extensions états → état-automates

Deux générations d'extensions

1. *automates de Moore et Mealy*
2. *automates structurés*

1. Automates de Moore et Mealy

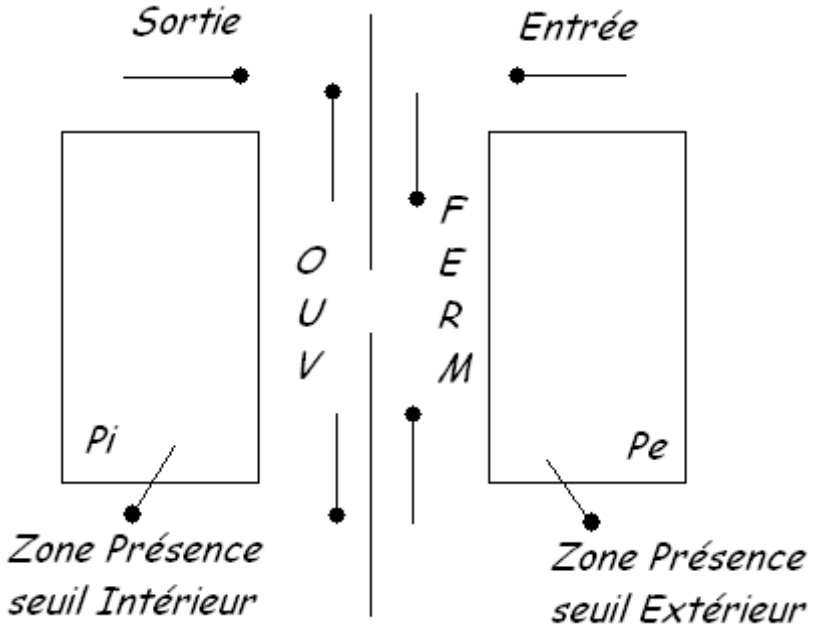
- Distribution des entrées / sorties (un alphabet d'entrée Σ et un de sortie Ω)
- Ajout de fonction de sortie ou d'action (w_1 ou w_2)
  - Moore : sortie associée à un état ($w_1 : Q \rightarrow \Omega$)
  - Mealy : sortie associée à une transition ($w_2 : Q \times \Sigma \rightarrow \Omega$)

Caractéristiques des AeF de Moore / Mealy :

- un état dénote l'état complet du système (non hiérarchique)
- le système est dans un seul état à la fois (pas de simultanéité de fonctionnement)

Exemple : commande d'une porte coulissante :

Commande ouverture / fermeture d'une porte coulissante (type grands magasins) en fonction de deux informations détectant la présence de personnes aux seuils intérieur et extérieur de la porte (capteurs optiques ou tapis sensitifs)



a.

E	S	États
Pi	OUV	Porte ouverte
Pe	FERM	Porte fermée

b.

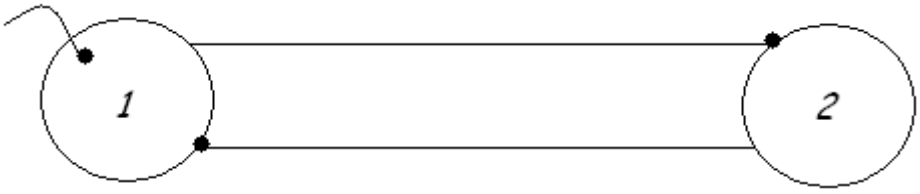
Une personne seule entre : ↑Pe ; ↓Pe ; ↑Pi ; ↓Pi ; FERM
Une personne seule sort : ↑Pi ; ↓Pi ; ↑Pe ; ↓Pe ; FERM

c.

Condition de sécurité.
Fermeture : ↓Pi/Pe/Pi = ↓Pi/Pe
↓Pe/Pe/Pi = ↓Pe/Pi

Représentation :

Porte Ouverte ↑Pe + ↑Pi / OUV Porte Fermée



↓Pi/Pe + ↓Pe/Pi / FERM

2. Automates structurés :

plusieurs notations utilisées :

- SDL (Specification of description Language),
- State chart,
- UML chart, ...

Diagrammes états / transitions généralisés.
Caractéristiques principales :

- Actions associées aux transitions et / ou aux états
- Conditions associées aux transitions (garde)
- Concurrence / synchronisation (plusieurs états possibles)
- États hiérarchiques (state chart)

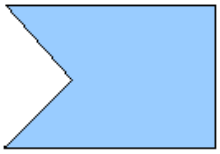
SDL – Notation

Etat

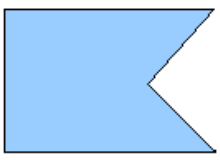
Transition :

Message :

Evenement / Trigger :

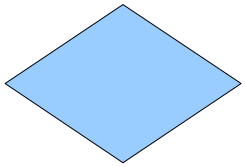


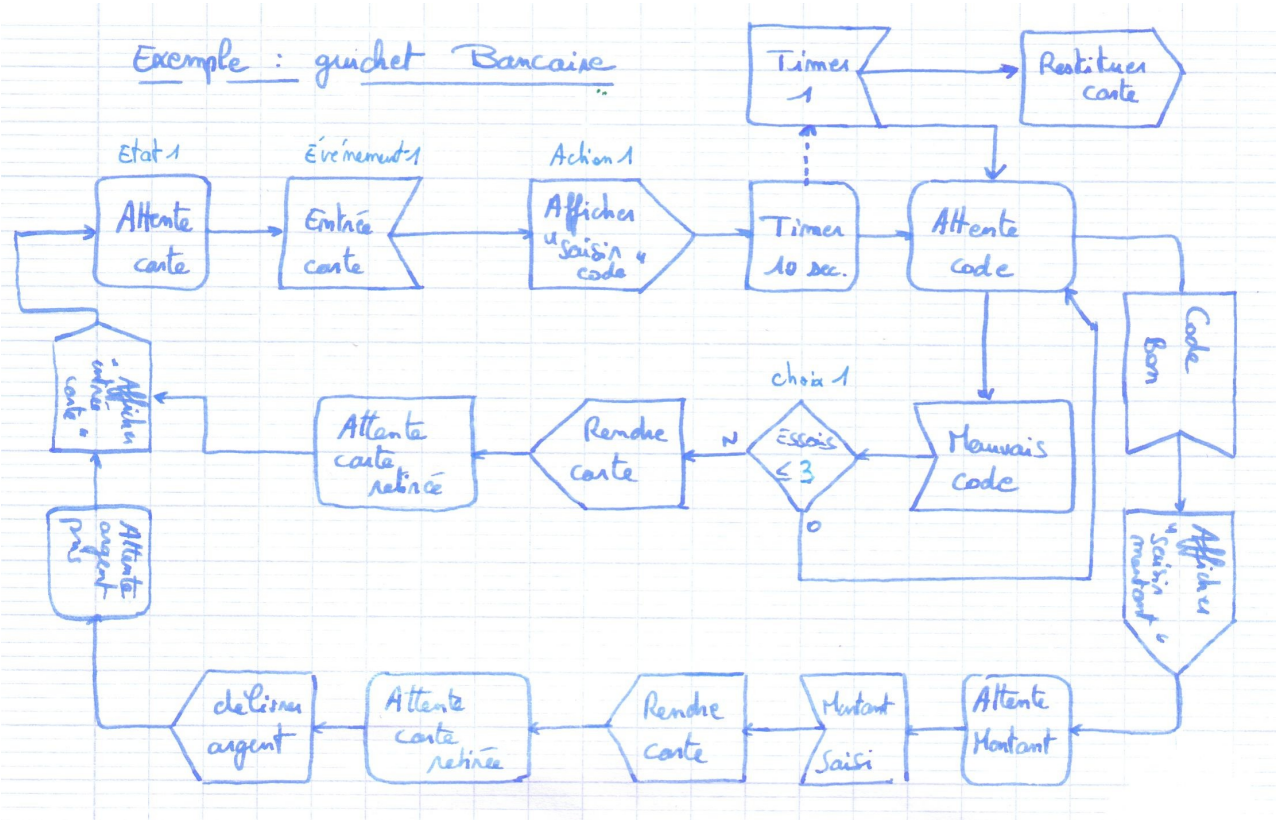
ou



Réponse / Action (action sans sortie) :

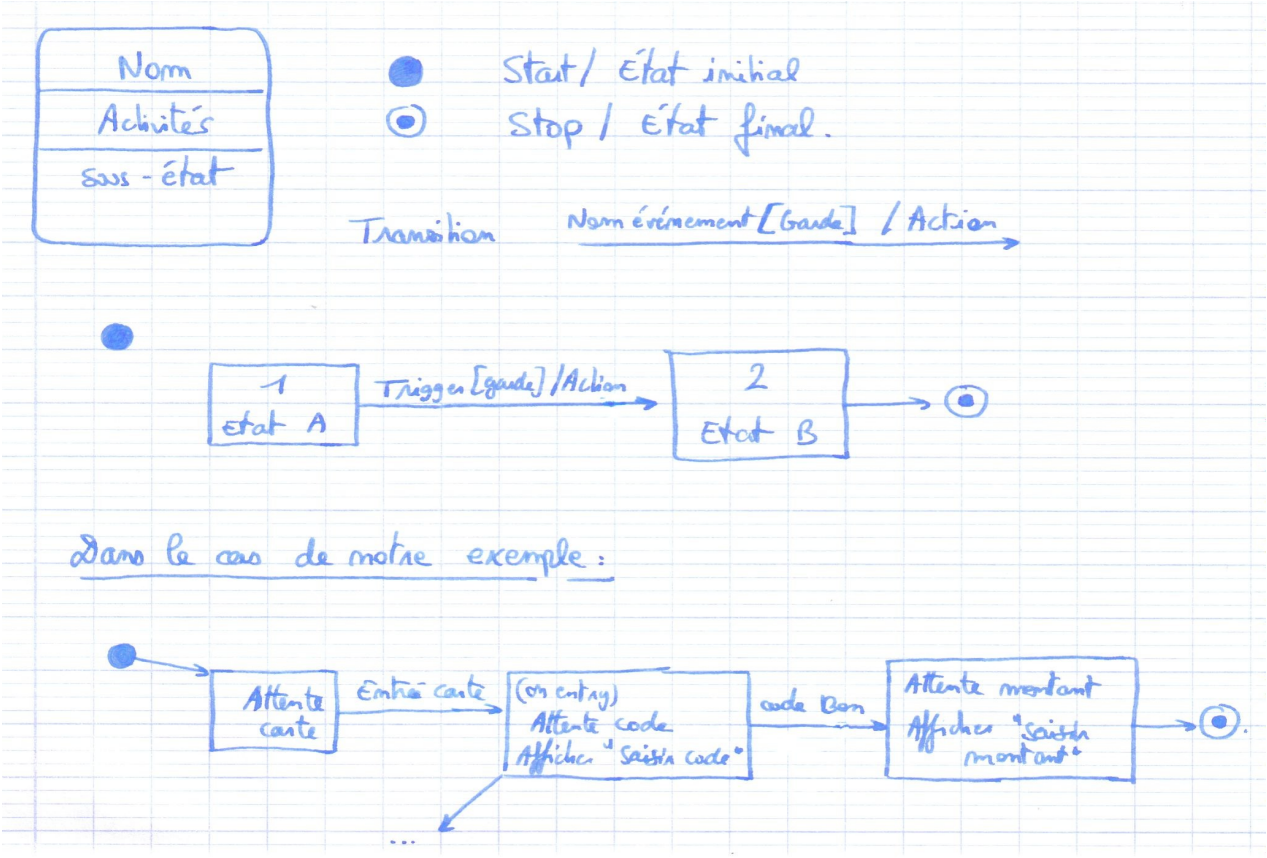
Choix / Décision :





State chart – Notation

État-Automate (décomposable) : permet une hiérarchisation

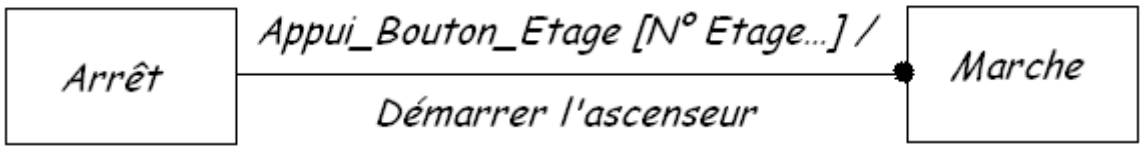


Activités : actions associées aux états.

- Do / Activité : l'action s'exécute tant que l'AeF est dans cet état ;
- On entry / Activité : l'action s'exécute à l'entrée de l'état ;
- On exit / Activité : l'action s'exécute à la sortie de l'état.

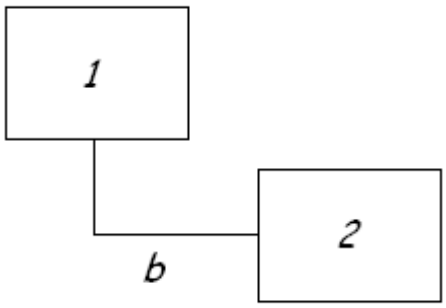
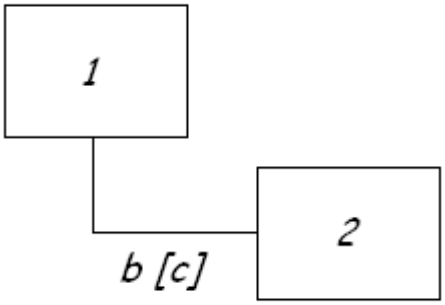
Nom Événement [garde] / action (associés aux transitions) :

- **Nom événement (param1 : type , param2 : type) :** événement qui peut déclencher le franchissement de la transition
  Exemple : appui_Bouton_Étage (N° Étage)
- **[Garde] :** expression booléenne (préconditions devant être vérifiées pour le franchissement)
  Exemple : [N° Étage <> Étage courant]
- **Action :** action associée au franchissement de la transition
  Exemple : démarrer l'ascenseur



State Chart – Notion de garde

Une transition est associée à un événement survenant dans la vie du système sans condition automatique.
La garde introduit des conditions à vérifier.

	
Dans l'état 1, si l'événement b se produit, le système passe dans l'état 2 sans aucune autre condition.	Dans l'état 1, si b se produit, ET si la condition c est vérifiée, le système passe dans l'état 2, si la condition c n'est pas vérifiée, le système reste dans l'état 1.

State chart – Hiérarchisation d'états

Dans le cas d'un comportement de systèmes complexes, les AeF sur un seul niveau deviennent rapidement illisibles.
Les state charts intègrent la notion d'état-automates (automates dans lesquels les états sont eux-même des automates)

- super-états : états principaux
- sous-états : décomposition des super-états.

La notion d'état-automate permet une approche structurée par une décomposition hiérarchique de l'étude d'un système.