UNIVERSITÉ DE SHERBROOKE

TM00
Techniques de modélisation

Références

Bray 10-14

TM00.1 - Présentation

Catégorisation des méthodes

Externe

- intrants + extrants
- décrire la correspondance
- boîte opaque (« noire »)

Interne

- service ::= fonction*
- décrire la décomposition
- boîte translucide (« blanche »)

TM00.1 - Présentation

Catégorisation des méthodes externes

- Comportementales
  - Fonctionnelles
    - Tables de décision (TD)
    - Cas d'utilisation (CU)
    - Descriptions textuelles (TEX)
  - À états
    - Automates (AEF, Mealy, Moore, SDL, SC)
    - Grammaires (GC, GA)
    - Réseaux de Petri (RP)
- Représentatoires (*representational*)
  - Statiques
    - Maquettes (MAQ)
  - Dynamiques
    - Prototypes (PRO)

TM00.1 - Présentation

Catégorisation des méthodes internes

- Données
  - Diagrammes entités-relations (DER)
  - Diagrammes entités-relations étendus (DERE)
- Processus
  - Uniques
    - Pseudo-code (PC)
    - Ordinogrammes et diagrammes d'activités (ORD)
  - Séquentiels
    - Diagrammes de structures (DS)
  - Concurrents
    - Diagrammes de flux de données (DFD)
    - Diagrammes d'états de Jackson (JSD)
    - Processus séquentiels communicants (CSP)
- Données x Processus
  - Traces d'entité (ELH – *entity life history*)
  - Modèles orientés objets (UML)

TM00.1 - Présentation
Sélection des techniques

- complète
  - doit permettre d'exprimer tout ce qu'il y à exprimer
- univoque
  - ne permet qu'une interprétation
- la plus simple
  - minimisation des risques d'erreurs

TM00.2 – Tables de décision

Références

- Bray section 12.4 - pages 259 à 266

TM00.2 – Tables de décision
Spécification du comportement

- Définir les interfaces externes sans limiter les choix de conception
- Technique de la "boîte noire"

TM00.2 – Tables de décision

Étapes

1. Identifier les entrées et les sorties de chaque fonction
   (ex: avec DFD)
2. Définir les entrées valides
   (ex: antécédents sur les entrées)
3. Définir la sortie pour chaque entrée valide (relation entrée-sortie)
   (ex: pseudo-code, table de décision, automate, machine à états, conséquents)

TM00.2 – Tables de décision

Structure générale

Decision rules ->>	1	2	3	4	5	.	.	.	.
Condition 1	Y	Y	Y	Y	N	N	.	.	.
Condition 2	Y	Y	.	.	.	.	.		
. . . .	.	.	.	.	.				
Action 2	O	N	O	O	.	.	.	.	
Action 2	O	N	N	N	.	.	.	.	.
. . . .	.	.	.	.	.				

TM00.2 – Tables de décision

Variante

Cond A				
Cond B				
. . . .				
Actions ->	**Action 1**	**Action 2**	**. . .**	**. . .**

Afficher les actions sur une seule ligne

TM00.2 – Tables de décision

Contenu

- Condition
  - fondée sur les paramètres d'entrée de la fonction
  - identifie
    - valeurs, ou
    - intervalles de valeurs, ou
    - condition sur les valeurs
- Action
  - sorties de la fonction

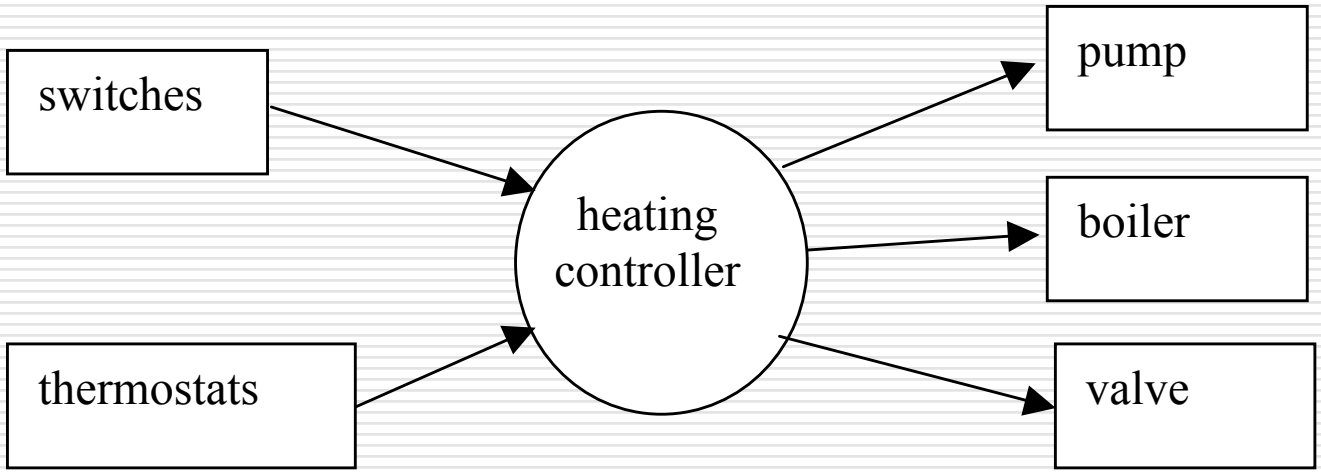
TM00.2 – Tables de décision

Nombre de règles de décision

- $n_1 * n_2 * \ldots$
  - où n_i représente le nombre de valeurs pour le paramètre i de la fonction
- peut donc être très grand
- compresser la table si la sortie est indépendante de certains paramètres d'entrée dans certains cas

TM00.2 – Tables de décision

Exemple 1 - Contexte



TM00.2 – Tables de décision

Exemple 1 - Texte de la spécification

"If hot water only is switched on, then the 3-way valve is set to the hot water position (A), the boiler and the pump are both turned on only when the tank temperature is below the tank thermostat setting. If heating only is switched on, then the valve is set to the heating position (B), the pump runs continuously and the boiler is turned on only when the room temperature is below the room thermostat setting. If the hot water and heating are both switched on, and both of the thermostat settings exceed the ambient temperature, then the boiler will be on, the pump will be running and the valve will be in the central position (C). If neither thermostat setting exceeds the ambient temperature, the boiler is turned off but the pump remains on and the valve is set to position B. If only the hot water tank thermostat setting exceeds its ambient temperature then boiler and pump are both on and the valve is set to position A. If only the room thermostat setting exceeds its ambient temperature then boiler and pump are both on but the valve is set to position B. If neither heating nor hot water are switched on then boiler and pump are turned off and the position of the 3-way valve is irrelevant."

TM00.2 – Tables de décision

Exemple 1 - Table de décision

heating	on								off							
hotwater	on				off				on				off			
tank thermostat	ov		un		ov		un		ov		un		ov		un	
room thermostat	ov	un	ov	un	ov	un	ov	un	ov	un	ov	un	ov	un	ov	un
boiler on		Y	Y	Y		Y		Y			Y	Y				
pump on	Y	Y	Y	Y	Y	Y	Y	Y			Y	Y				
3-way valve	B	B	A	C	B	B	B	B	A	A	A	A	-	-	-	-

ov(er) = ambient temperature over thermostat setting,
un(der) = ambient temperature under thermostat setting,
A = hot water position, B = heating, C = both.

TM00.2 – Tables de décision

Exemple 1 - Table de décision compressée

central heating	1	2	3	4	5	6	7	8	9
heating	on						off		
hotwater	on				off		on		off
tank thermostat	ov		un		-		ov	un	-
room thermostat	ov	un	ov	un	ov	un	-	-	-
boiler on		Y	Y	Y		Y		Y	
pump on	Y	Y	Y	Y	Y	Y		Y	
3 way valve	B	B	A	C	B	B	A	A	-

TM00.2 – Tables de décision

Règle du SINON

triangle	1	2	3	4	5	ELSE
A = B	Y	Y	N	N	N	
B = C	Y	N	Y	N	N	
A = C	-	-	-	Y	N	
A + B > C	-	Y	-	-	Y	
B + C > A	-	-	Y	-	-	
A + C > B	-	-	-	Y	Y	
Equilateral	Y					
Isosceles		Y	Y	Y		
Irregular					Y	
Not a triangle						Y

TM00.2 – Tables de décision

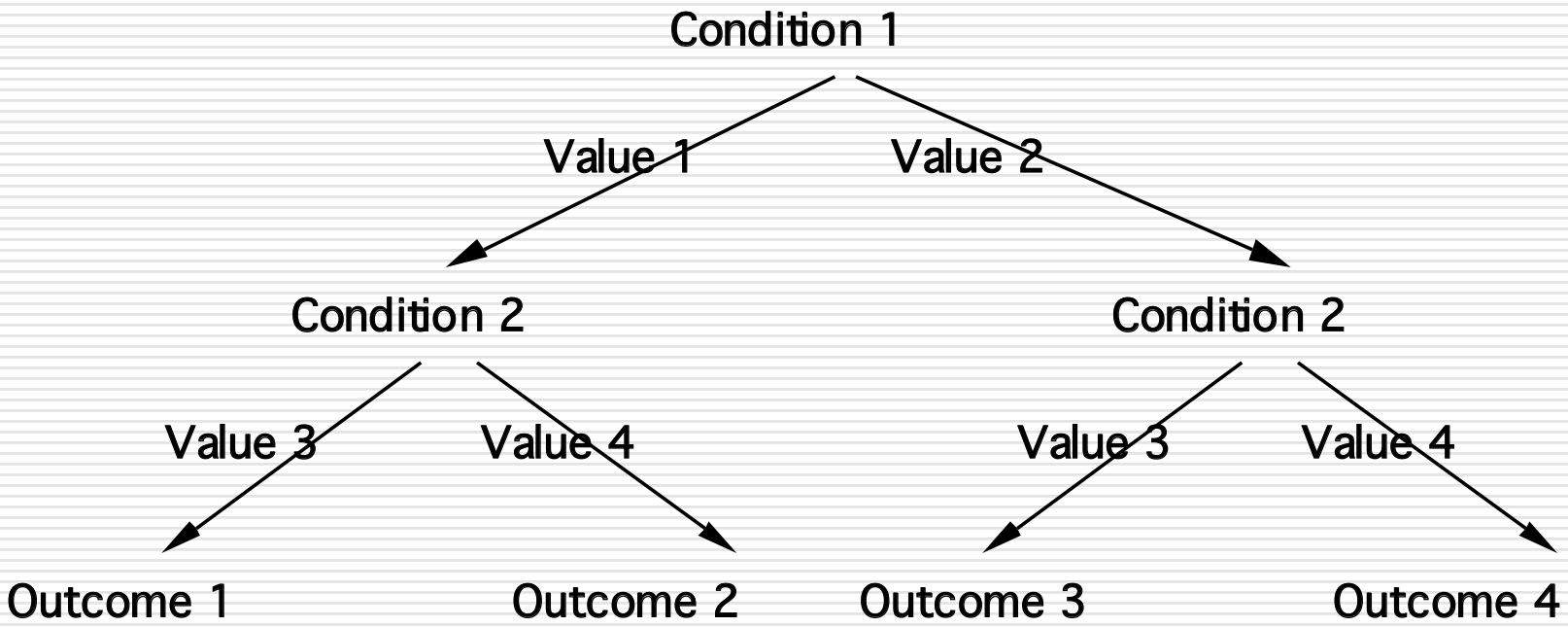
Tables liées

central heating (table 1)	1	2	3	4
heating	on		off	
hot water	on	off	on	off
	see table 2	see table 3	see table 4	pump and boiler off

table 2 (heat & h/w on)	1	2	3	4
tank thermostat	ov		un	
room thermostat	ov	un	ov	un
boiler on		Y	Y	Y
pump on	Y	Y	Y	Y
3-way valve	B	B	A	C

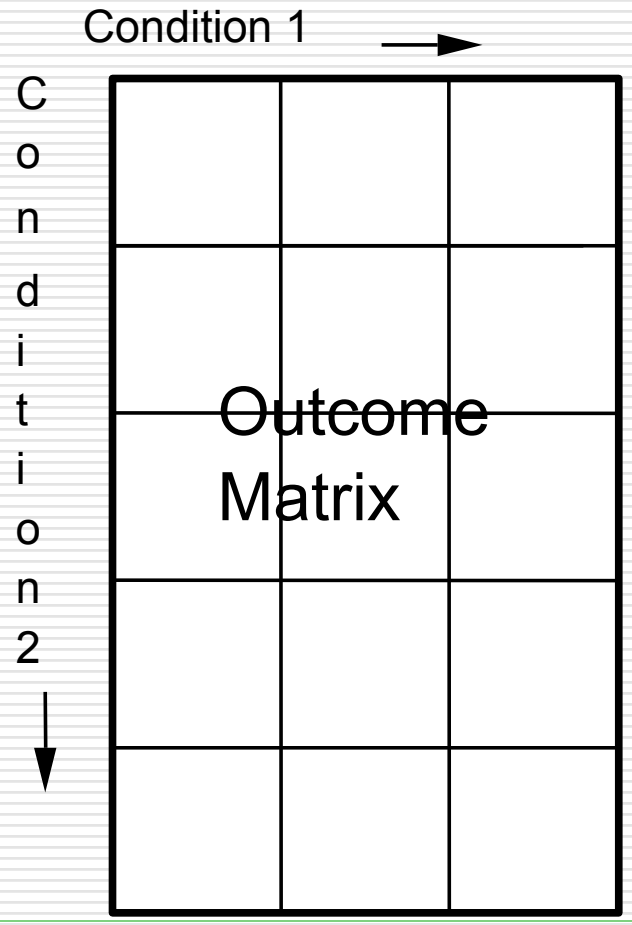
TM00.2 – Tables de décision

Arbre de décision



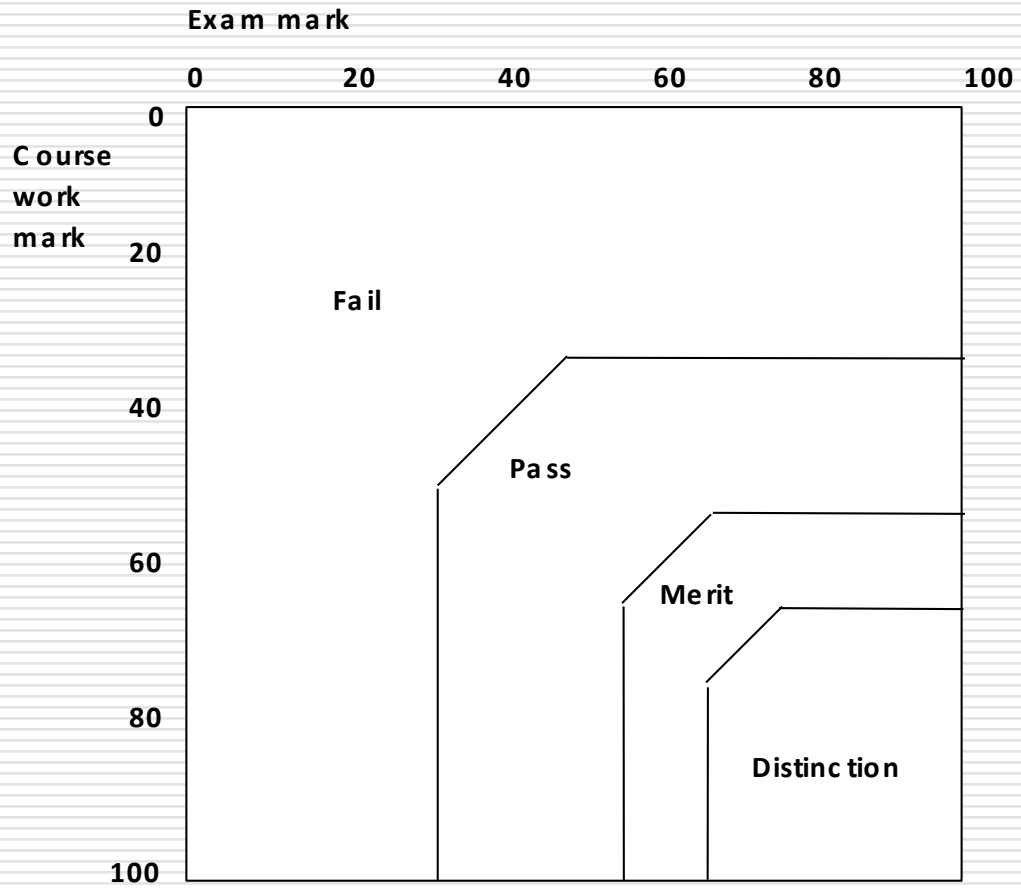
TM00.2 – Tables de décision

Matrice de décision



TM00.2 – Tables de décision

Matrice de décision – un exemple



TM00.2 – Tables de décision

Exercice 1

- Une personne peut recevoir une pension si elle est âgée de 65 ou plus, si elle a suffisamment contribuée et si elle n'occupe pas un emploi à temps plein.
- Si elle est âgée de 70 ans ou plus, la condition d'emploi ne s'applique pas.

TM00.2 – Tables de décision

Exercice 1 - Solution

- ... allez, un petit effort!

TM00.2 – Tables de décision

Exercice 2

- Un système de gestion des configurations doit gérer l'accès aux fichiers.
- Une demande d'accès peut-être soit "write" ou "read".
- L'accès courant au fichier est soit "write", "read" ou "aucun".
- L'état du fichier peut être "new" ou "release".
- On peut accorder à un utilisateur un accès en "read" ou "write" si le fichier n'a aucun accès courant et si le fichier n'est pas "release".
- S'il est "release", il peut être accédé seulement en "read".
- Si l'accès courant est "write", alors on peut accorder seulement un accès en "read".
- Le propriétaire du fichier doit être avisé si quelqu'un obtient un accès en "write" ou si on essaie d'obtenir un accès sur un fichier "release".

TM00.2 – Tables de décision

Exercice 2 - solution

current access	read				write				none			
requested access	read		write		read		write		read		write	
file type	n	r	n	r	n	r	n	r	n	r	n	r
allow access	y	y	n	n	y	n	n	n	y	y	y	n
notify DA		y		y			y	y			y	y

Trouvez les erreurs!

TM00.2 – Tables de décision

Avantages

- Définition de la relation causale entre
  - conditions sur les entrées
  - sortie produite
- Consultation facile
  - plus simple que le pseudo-code dans plusieurs cas
  - non ambigüe
- Exhaustivité

TM00.2 – Tables de décision

Inconvénients

- Usage limité
- Inexistence d'outils spécialisés
- Inadéquation des outils bureautiques (traitement de texte, chiffrier)

TM00.3 – Diagrammes états-transitions

Références

- Bray section 12.4 - pages 259 à 266
- Bray section 12.6 - pages 266 à 283
- Desharnais, J., Frappier, M., Mili, A.; *State Transition Diagrams, in :* Handbook on Architectures of Information Systems, P. Bernus, K. Mertins, G. Schmidt, ed., Springer-Verlag, 1998, ISBN 3-540-64453-TM00.

UNIVERSITÉ DE SHERBROOKE

TM00.3 – Diagrammes états-transitions

… et autres automates!

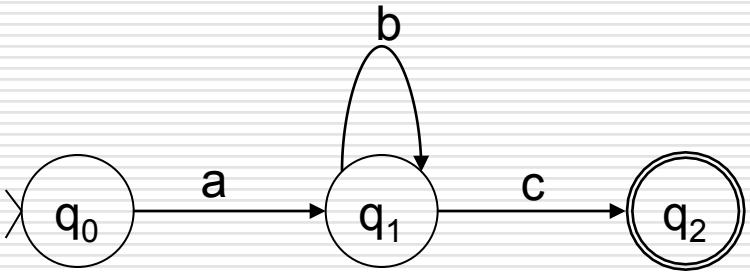
- Automates à états finis
  - déterministes
  - non déterministes
- Machines à états
  - Moore
  - Mealy
- Machine à états étendue
  - SDL
  - State chart
  - ROOM
- Diagrammes d'états
  - UML
  - …
- Diagramme de transitions
  - UML
  - …
- Notations
  - Yourdon
  - SDL
  - UML
  - …

TM00.3.1 – Automate déterministe

Définition

- Un automate d'états fini (Q, A, T, q0) où
  - Q : ensemble fini des états
  - A : alphabet, ensemble fini de symboles
  - T : fonction de transition, Q X A -> Q
  - q0 : état initial, élément de Q
- L'automate peut être muni d'un ensemble d'états dits finaux
  - F : ensemble des états finaux, sous-ensemble de Q
- L'automate est *déterministe* ssi
  - T une fonction

TM00.3.1 – Automate déterministe
Représentation graphique



Mais qu'arrive-t-il dans le cas T(q0,b) ?

Deux stratégies :
- le puits
- l'élimination

$Q = \{q_0, q_1, q_2\}$

$A = \{a, b, c\}$

$T = \{(q_0,a) \mapsto q_1, (q_1,b) \mapsto q_1, (q_1,c) \mapsto q_2\}$

$F = \{q_2\}$

TM00.3.1 – Automate déterministe

Modélisation

- Représentation de séquences d'évènements valides d'un système
- Un événement peut prendre plusieurs formes
  - une entrée,
  - une sortie,
  - un appel de méthode,
  - etc.
- Représentation des états d'un système
  - habituellement impossible de représenter tous les états du système
  - point de vue partiel
- Difficile de représenter le parallélisme

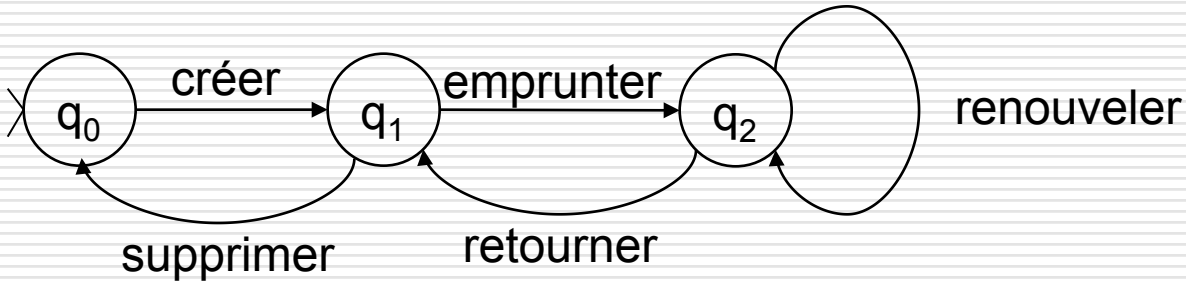
TM00.3.1 – Automate déterministe

Exercice 1

- Utilisez un automate pour spécifier le comportement d'une bibliothèque
- Prenez en compte seulement les prêts de livres aux membres
- Décrivez les limitations et les informations implicites

TM00.3.1 – Automate déterministe

Exercice 1 - solution



information implicite et limitations

- point de vue d'un livre seulement
- n'exprime pas les contraintes sur le membre
  - limite de prêt
  - n'indique pas si un membre peut emprunter plusieurs livres à la fois

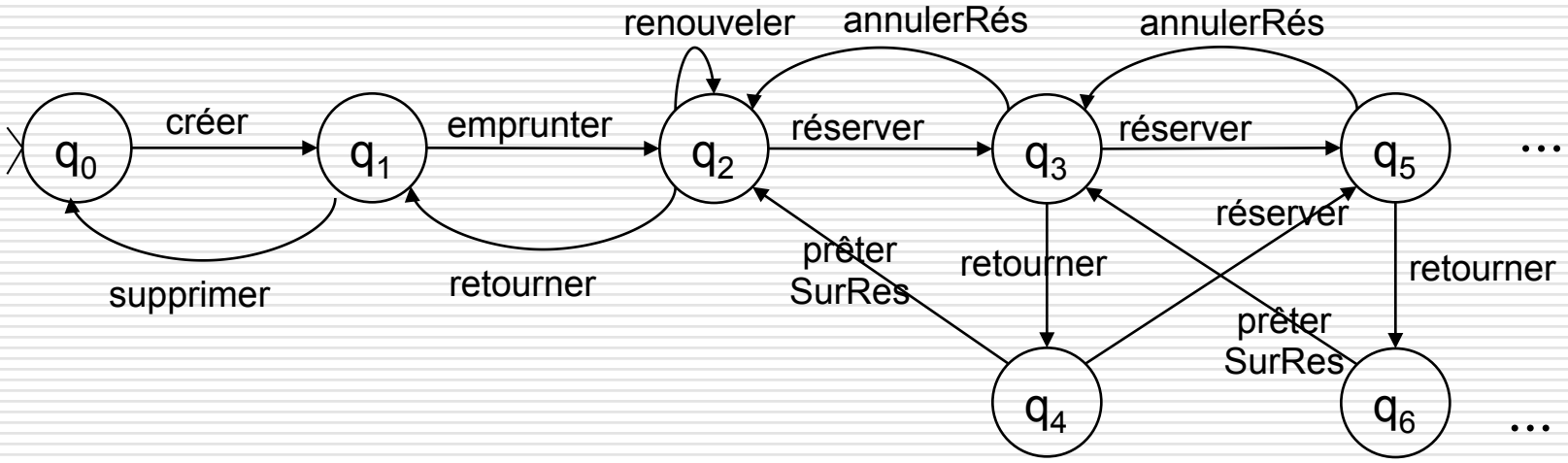
TM00.3.1 – Automate déterministe

Exercice 2

- Ajouter les réservations à l'exercice précédent
- Décrivez les limitations et les informations implicites

TM00.3.1 – Automate déterministe

Exercice 2 - solution



Limitations et informations implicites

- ne représente que deux réservations à la fois
- impossible de représenter toutes les réservations
- n'exprime pas les contraintes sur le membre :
  - l'emprunteur courant ne peut réserver son livre
  - réservations servies dans l'ordre d'arrivée
  - réservations annulées dans n'importe quel ordre

TM00.3.1 – Automate déterministe

Exercice 3

- Décomposez votre spécification en plusieurs diagrammes
- Exprimez les liens entre les diagrammes

TM00.3.1 – Automate déterministe

Exercice 4

- Écrivez le pseudo-code d'un programme qui traite les évènements d'une bibliothèque
  - soyez le plus précis possible ;
  - faites référence au modèle de données ;
  - donnez tous les antécédents et toutes les mises à jour de chaque évènement

TM00.3.1 – Automate déterministe

Exercice 4 (suite)

- traitez les évènements suivants :

```
creer <idLivre> <titre> <auteur> <dateAcquisition>
emprunter <idLivre> <idMembre> <dateEmprunt>
renouveler <idLivre> <dateRenouvellement>
retourner <idLivre> <dateRetour>
supprimer <idLivre>
inscrire <idMembre> <nom> <telephone> <limitePret>
désinscrire <idMembre>
reserver <idReservation> <idLivre> <idMembre> <dateReservation>
preterSurRes <idReservation> <dateEmprunt>
annulerRes <idReservation>
afficherPrets
```

TM00.3.1 – Automate déterministe

Exercice 5

- Décrivez les liens entre vos automates et votre pseudo-code pour la bibliothèque
  - indiquez à quel état du système correspond un état de l'automate
  - indiquez le lien entre les antécédents des évènements et le pseudo-code
  - indiquez les forces et les limitations de la notation d'automate

TM00.3.1 – Automate déterministe

Limitations des automates

- Les paramètres d'entrée et de sortie des évènements ne sont pas explicites
- Il est difficile de représenter entièrement l'espace du système
- Il est difficile de donner tous les antécédents
- Il est difficile d'associer des messages d'erreurs
- Un automate ne sait pas compter!

TM00.3.1 – Automate déterministe

Exercice 6

- Spécifiez à l'aide d'un automate le comportement d'un système téléphonique de base (POTS)
  - décrochez
  - composer un numéro
  - répondre à un appel
- Identifiez les entrées et les sorties

TM00.3.1 – Automate déterministe

Exercice 6 (suite)

- Faire
  - le modèle conceptuel de données du POTS
  - faire le diagramme de classe du POTS
  - faire les cas d'utilisation du POTS
- Quels liens existent-il entre ces diagrammes ?

TM00.3.1 – Automate déterministe

Exercice 7

- Définissez la notion d'équivalence de deux automates

TM00.3.2 – Machine à états

Machine à états finie

- Associer une sortie à toute entrée
  - à la réception d'une entrée, une sortie est produite
- Identifier de la nature d'un évènement (entrée ou sortie)
- Distinguer l'ensemble des entrées de celui des sorties
- Proposé par Mealy et Moore (séparément)
  - Mealy : sortie associée à une transition
  - Moore : sortie associée à un état

TM00.3.2 – Machine à états

Machine de Mealy

- Automate de Mealy (Q, A, B, T, S, q0) où
  - Q : ensemble fini des états
  - A : alphabet d'entrée, ensemble fini de symboles
  - B : alphabet de sortie, ensemble fini de symboles
  - T : fonction de transition, Q X A -> Q
  - S : fonction de sortie, Q x A -> B
  - q0 : état initial, élément de Q
- T et S sont des fonctions

TM00.3.2 – Machine à états

Machine de Mealy : convention

- Transition q1 --- a / b --> q2:
  - si l'entrée « a » est reçue alors que le système est dans l'état q1, la sortie « b » est produite et le nouvel état du système est q2
  - « a » est aussi appelé le déclencheur (*trigger*)
- $\lambda \in B$: sortie inintéressante, ou absente
- Si une entrée est reçue et qu'aucune transition n'est définie pour cette entrée, il ne se passe rien (l'entrée est ignorée)

TM00.3.2 – Machine à états

Machine de Moore

- Automate de Moore (Q, A, B, T, S, q0) où
  - Q : ensemble fini des états
  - A : alphabet d'entrée, ensemble fini de symboles
  - B : alphabet de sortie, ensemble fini de symboles
  - T : fonction de transition, Q X A -> Q
  - S : fonction de sortie, Q -> B
  - q0 : état initial, élément de Q
- T et S sont des fonctions

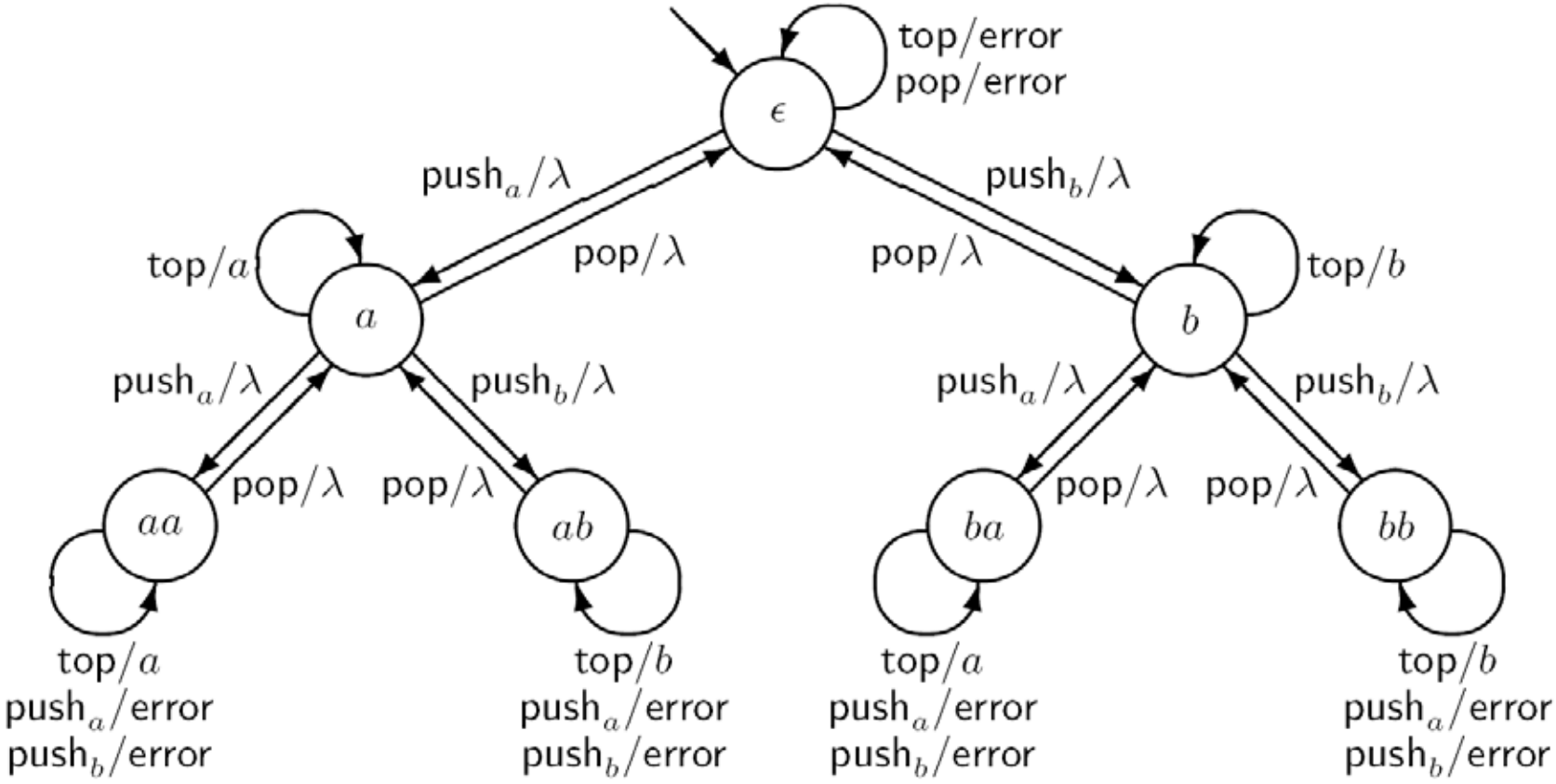
TM00.3.2 – Machine à états

Exercice

- Spécifier une pile (bornée) avec une machine de Mealy

TM00.3.2 – Machine à états

Machine Mealy : exemple d'une pile (bornée)



TM00.3.2 – Machine à états

Caractéristiques d'une machine de Mealy

- non hiérarchique
- un état dénote l'état complet du système
- le système est dans un seul état à la fois
- une transition est atomique; elle ne peut être décomposée

TM00.3.2 – Machine à états

Séquence : définitions

- S* dénote l'ensemble de toutes les séquences (incluant la séquence vide ε) formées à partir d'éléments de S
- S+ dénote l'ensemble de toutes les séquences non vides formées à partir d'éléments de S
- S* = S+ union {ε}
- On dénote par t1t2 la concaténation des séquence t1 et t2
- Les séquences sont aussi appelées « traces »

TM00.3.2 – Machine à états
Machine de Mealy : sémantique des traces

Definition 2.2. *The* behavioral abstraction *(*semantics*) of a Mealy machine*

$$\Sigma = (S, I, O, \delta, \gamma, s_0)$$

is the function $g_\Sigma : I^+ \to O$ *defined by the following recursive equations, where* $d_\Sigma : I^\star \to S$ *is an auxiliary function,* $x \in I$ *and* $t \in I^\star$.

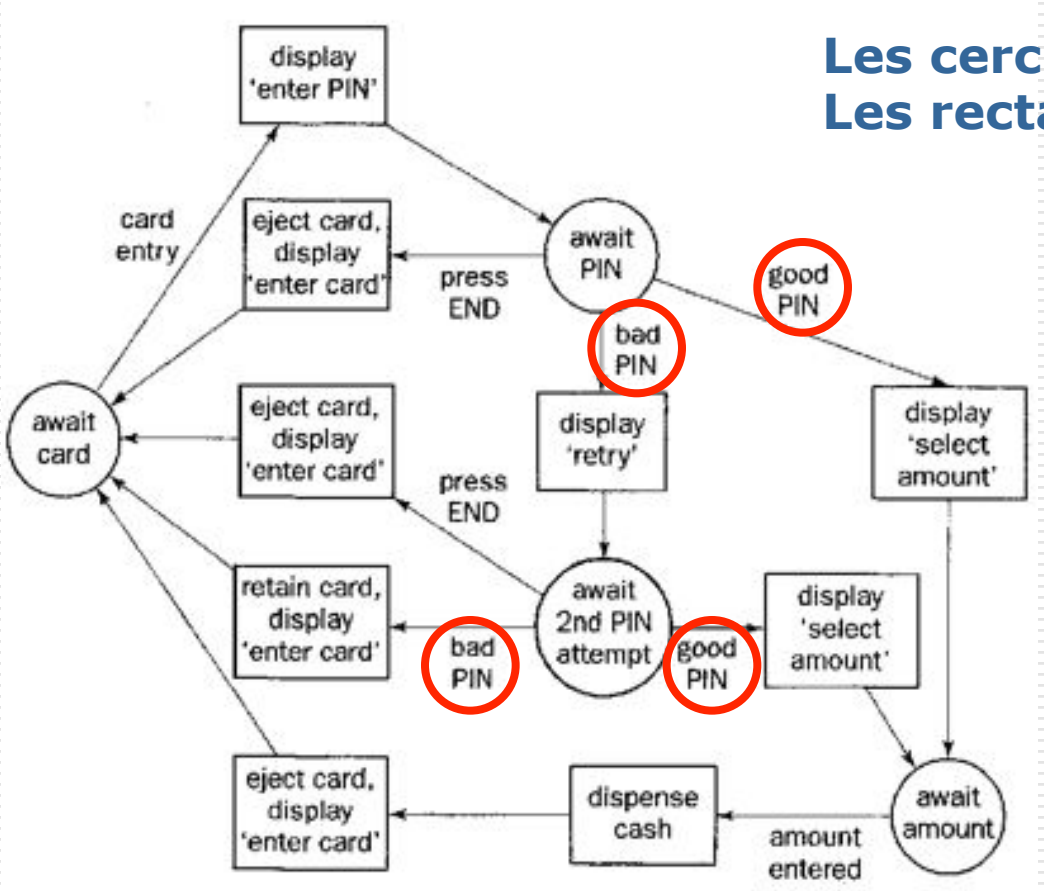
$$d_\Sigma(\tau) = s_0, \qquad d_\Sigma(tx) = \delta(d_\Sigma(t), x), \qquad g_\Sigma(tx) = \gamma(d_\Sigma(t), x).$$

Two Mealy machines

$$\Sigma = (S, I, O, \delta, \gamma, s_0) \text{ and } \Sigma' = (S', I, O, \delta', \gamma', s_0{}')$$

are equivalent *if and only if* $g_\Sigma(t) = g_{\Sigma'}(t)$ *for all* $t \in I^+$.

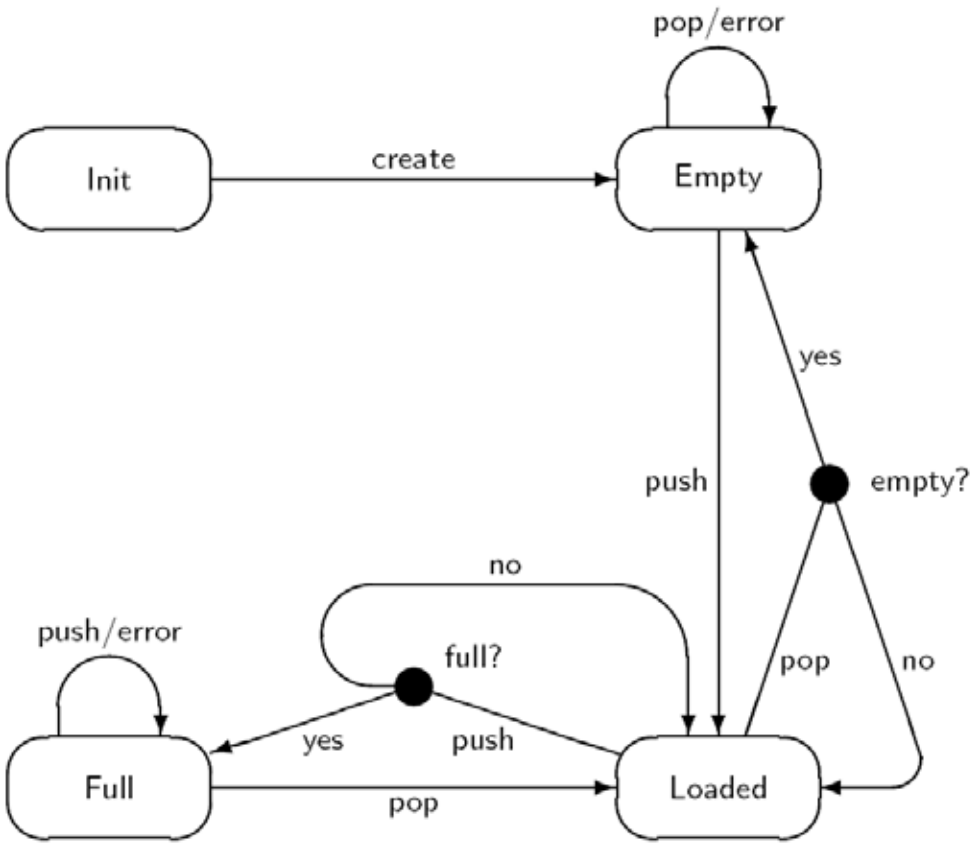
TM00.3.2 – Machines à états
Variante STD : Exemple d'un guichet bancaire



Les cercles représentent les états
Les rectangles, les actions

Comment déterminer good ou bad?

TM00.3.2 – Machines à états
Variante UML : Exemple d'une pile

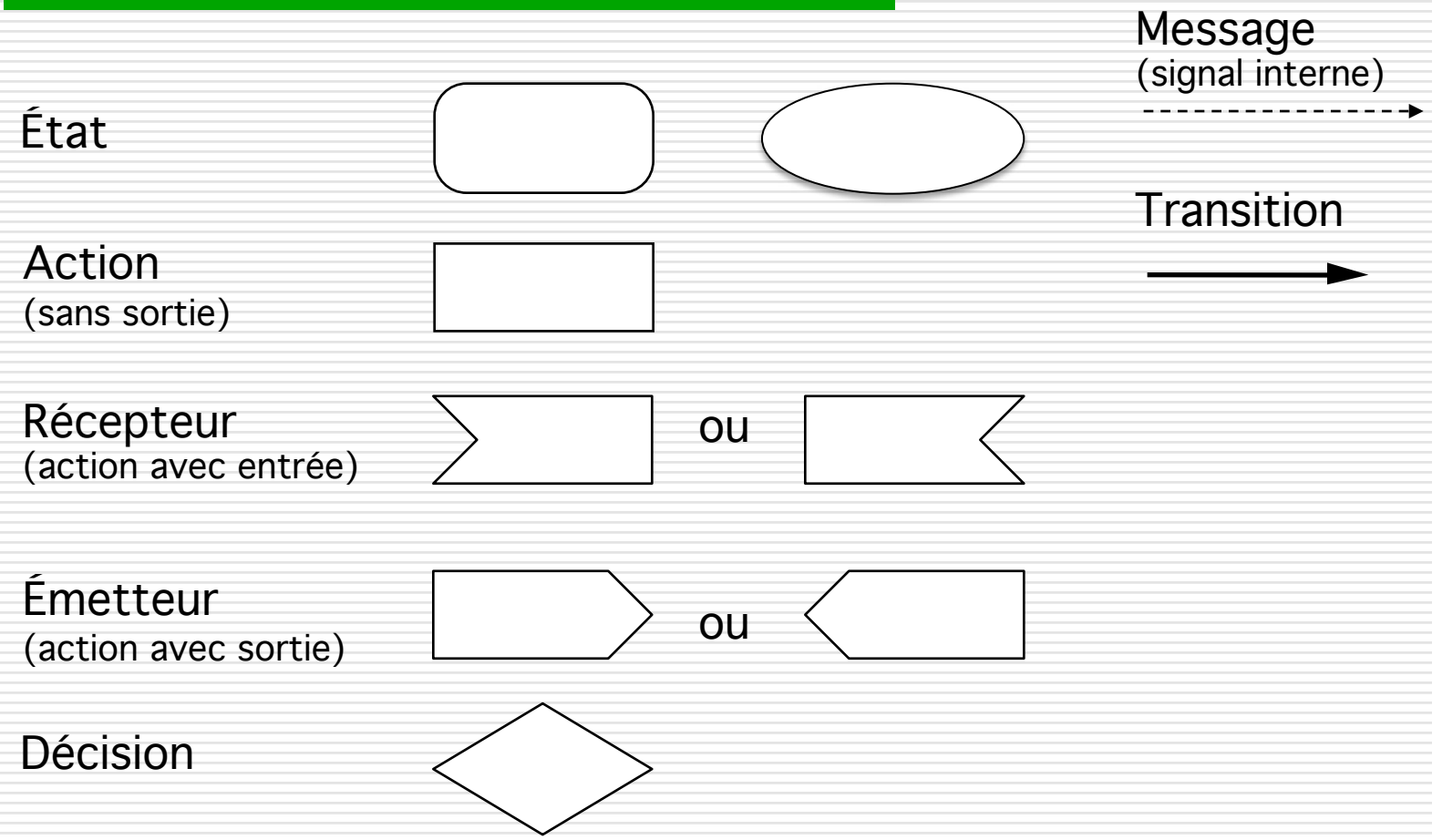


TM00.3.3 – Diagrammes SDL
Origines

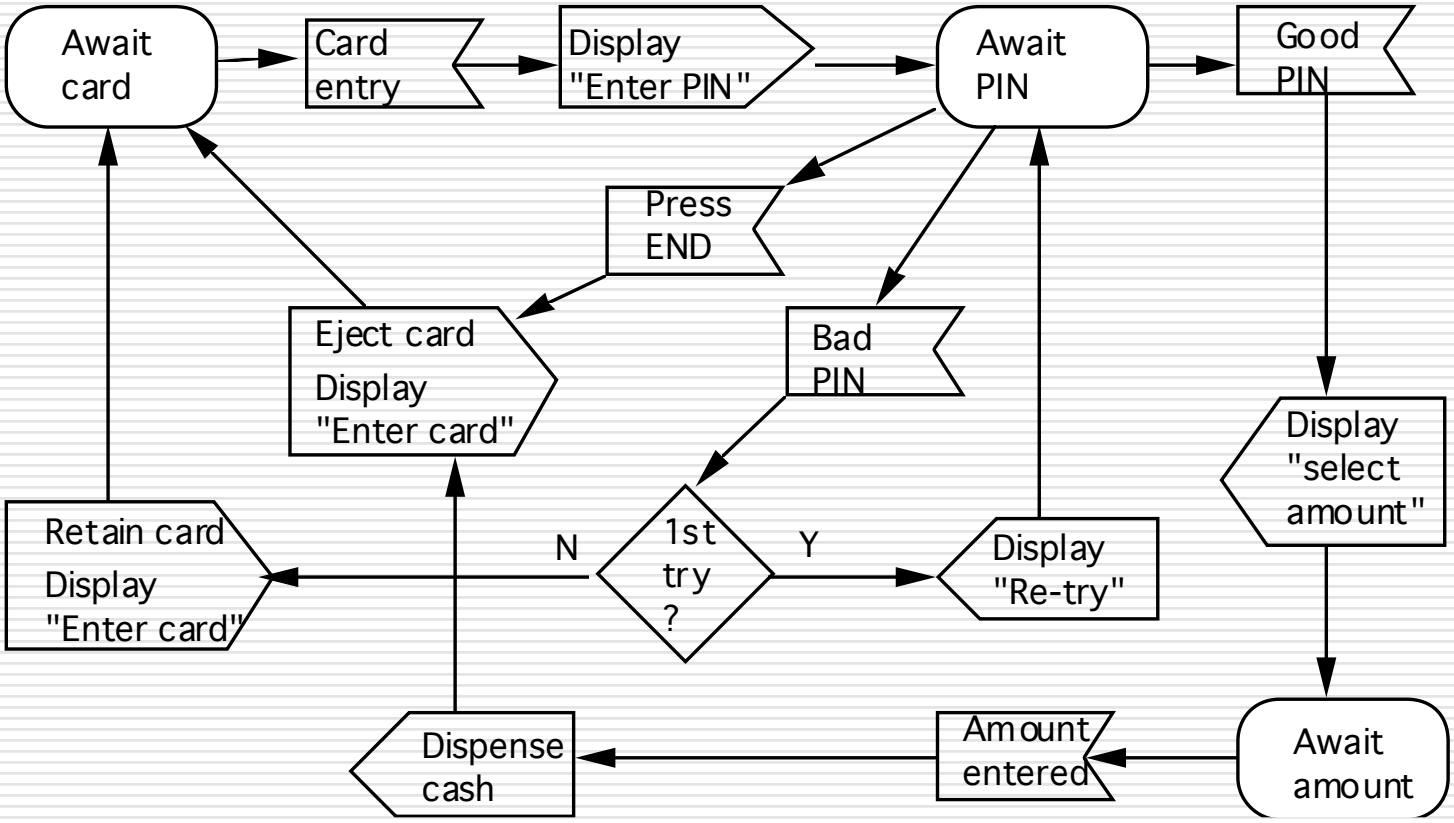
- SDL - Specification and Description Language.
- Norme CCITT, puis ITU-T.
- Évolution soutenue de 1972 à 2007.
- Concepts fondamentaux
  - Agent, signal, processus, procédure, type abstrait
- Modèle
  - Agents concurrents communiquant par queues de signaux priorisés

TM00.3.3 – Diagrammes SDL

Notation



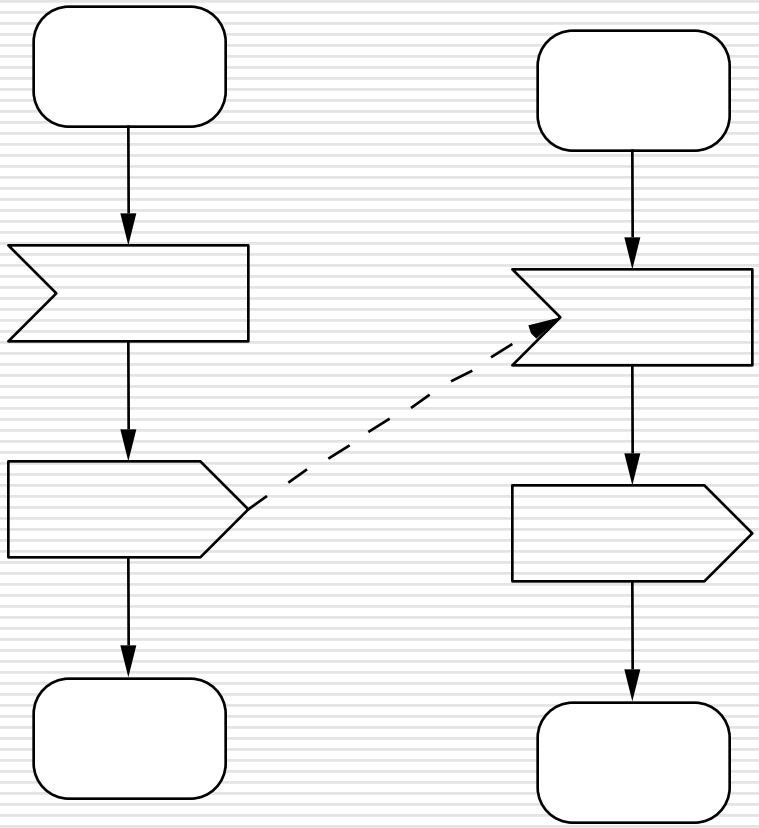
TM00.3.3 – Diagrammes SDL
Exemple - Guichet bancaire



TM00.3.3 – Diagrammes SDL

Concurrence

un trait pointillé dénote un envoi de message



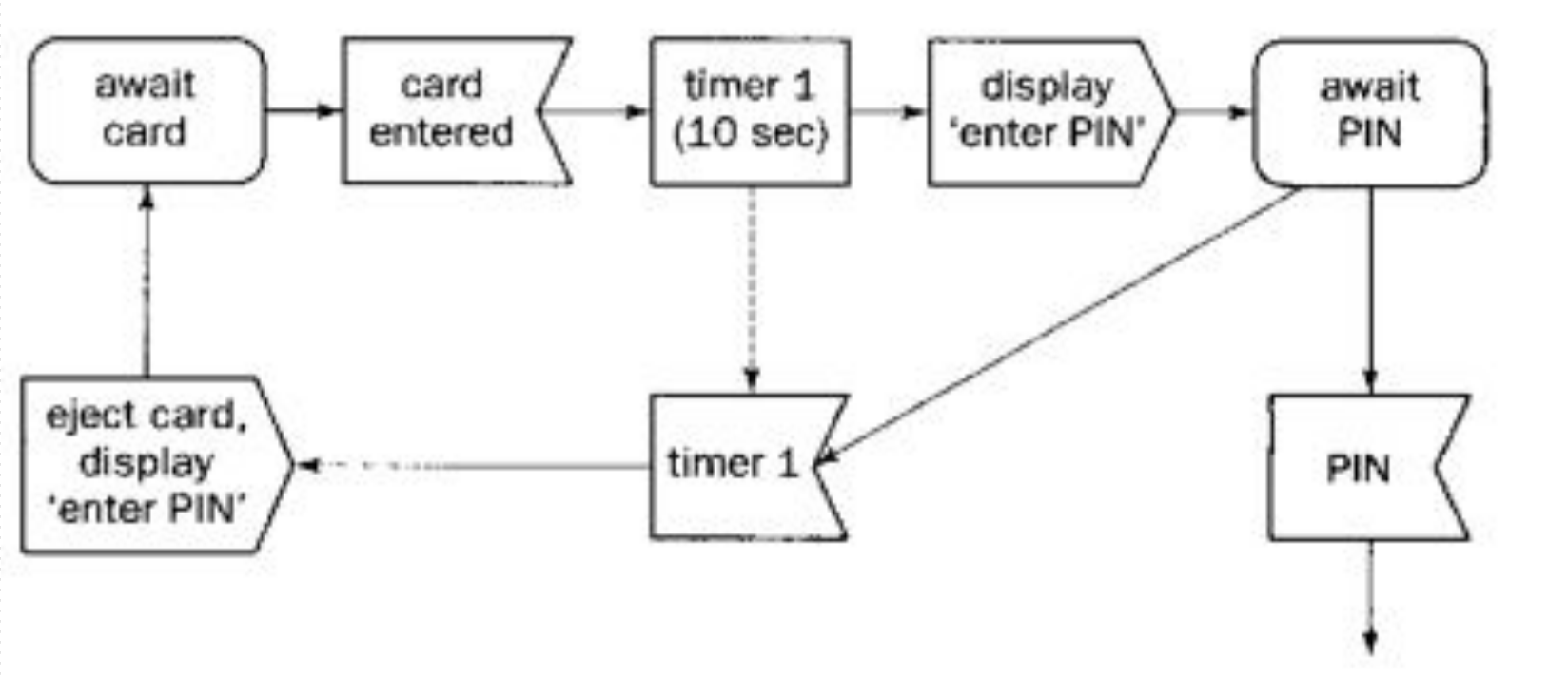
TM00.3.3 – Diagrammes SDL

Temporisation

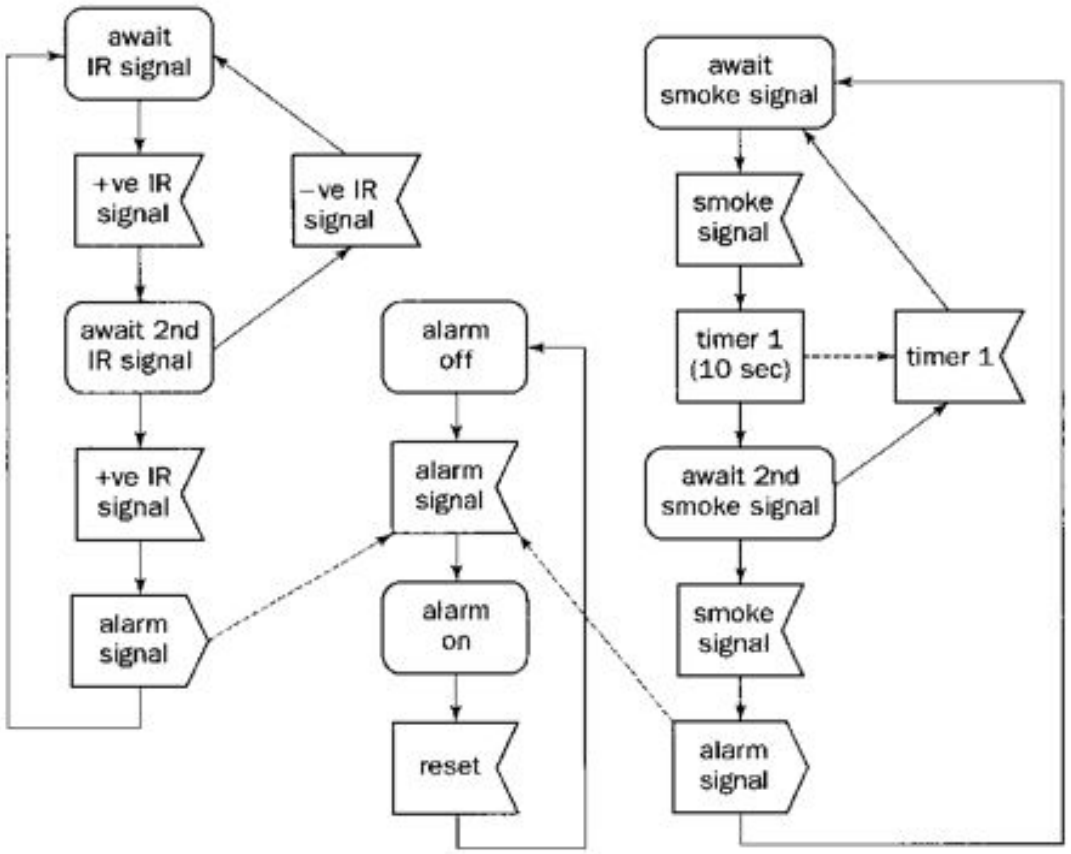
- Une temporisation vise à définir un comportement lorsqu'aucun des signaux attendus dans un état Y n'est reçu dans un délai prescrit **t**.
- Soit S-in<1>, …, S-in<n>, les signaux attendus par l'état Y
- Soit X l'état antérieur unique de Y (au besoin, introduit à cet effet)
- Soit X-in, X-act et X-out les éventuelles actions de la transaction de X vers Y
- La temporisation est composée de deux actions : une action simple (**A**) et un émetteur (**D**).
- L'action **A** entre X-act et X-out amorce le temporisation.
- L'émetteur **D** est placé sur sa propre transition sortante de Y.
- À l'expiration du délai **t**, **A** envoie un signal à **D** qui déclenche les actions à la suite de **D**.
- Si l'un des S-in est reçu avant l'expiration du délai **t**, **A** est supendue de telle sorte que **D** ne sera pas franchi.

TM00.3.3 – Diagrammes SDL

Temporisation - exemple



TM00.3.3 – Diagrammes SDL
Exemple - alarme vol-incendie



TM00.3.3 – Diagrammes SDL

Exercice 8

- Spécifier le comportement d'un livre avec SDL
- Prenez en compte les prêts, les réservations et les retards

UNIVERSITÉ DE SHERBROOKE

TM00.3.3 – Diagrammes SDL

Plusieurs autres possibilités

- Définir des agents échangeant des signaux
- Définir, créer et supprimer des processus
- Définir et appeler des procédures
- Définir des types abstraits
- …
- voir norme ITU-T Z.100 (11/2007)

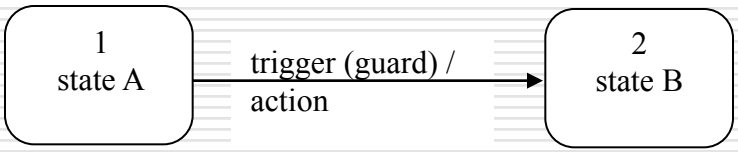
TM00.3.4 – Diagrammes d'états généralisés
Propriétés

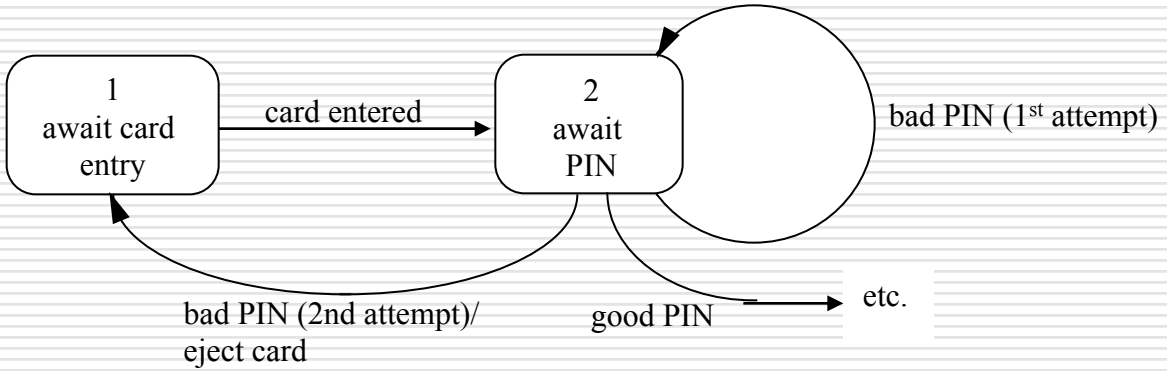
- Machine à état concurrente avec
  - Hiérarchisation
  - Actions associées aux transitions et aux états
  - Condition associée aux transitions

- En anglais, « state chart »

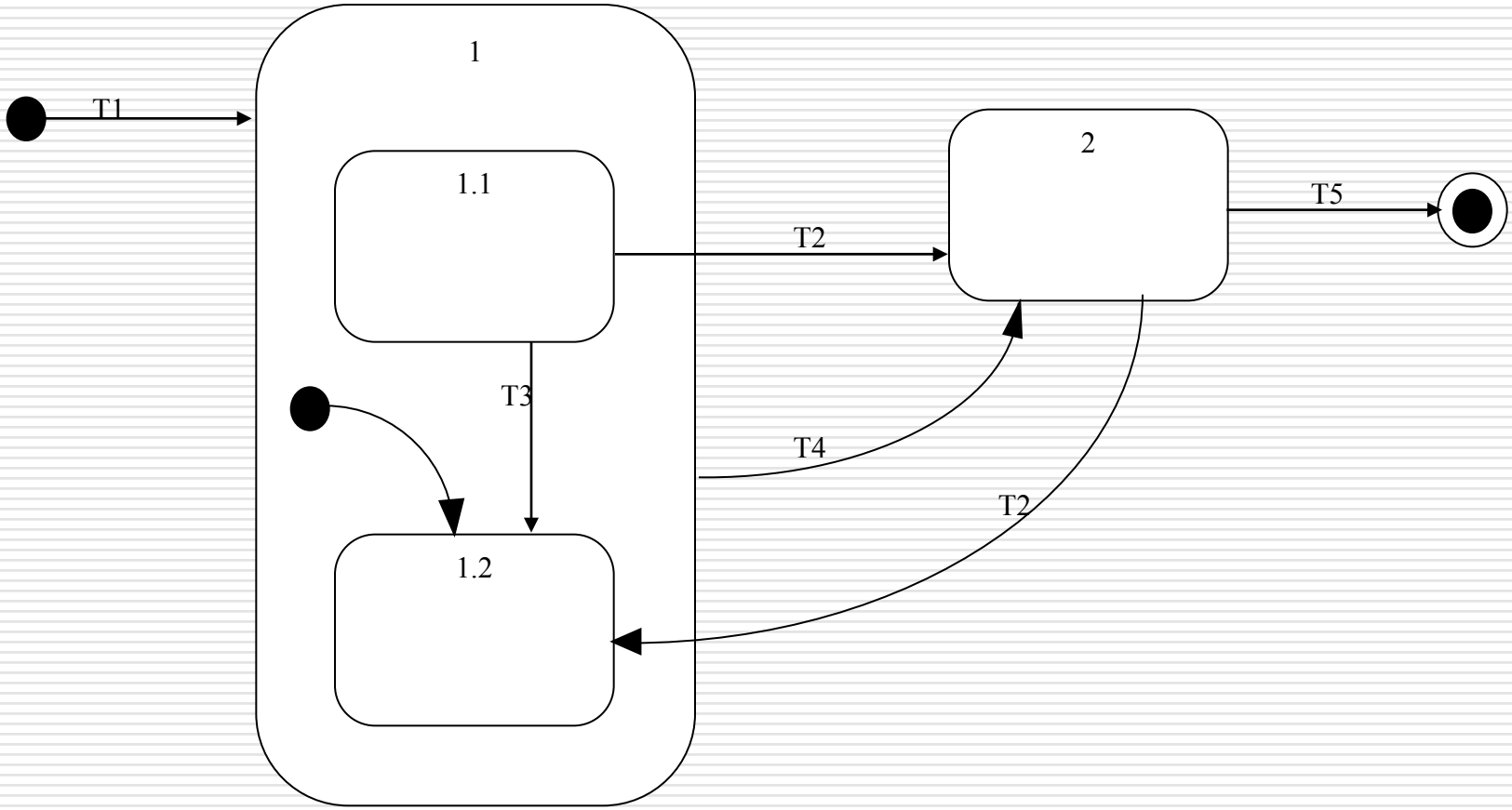
TM00.3.4 – Diagrammes d'états généralisés

Notation

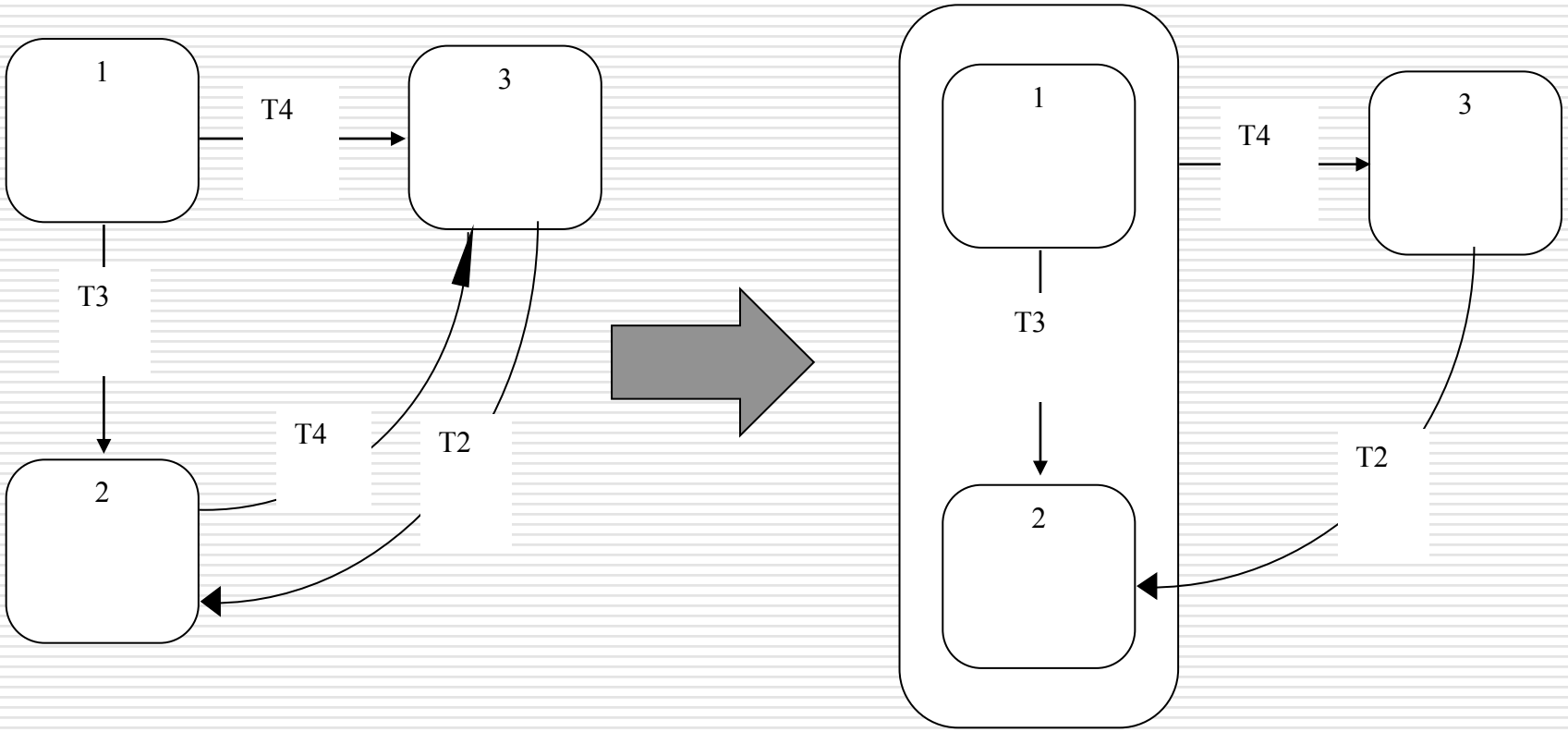




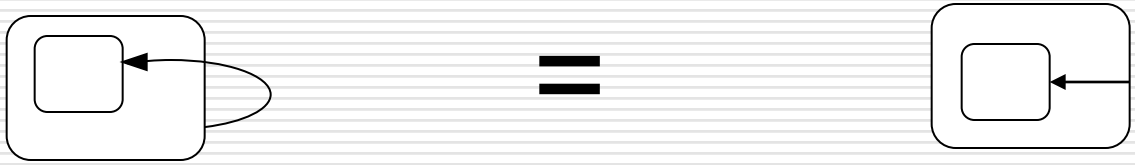
TM00.3.4 – Diagrammes d'états généralisés
État hiérarchique



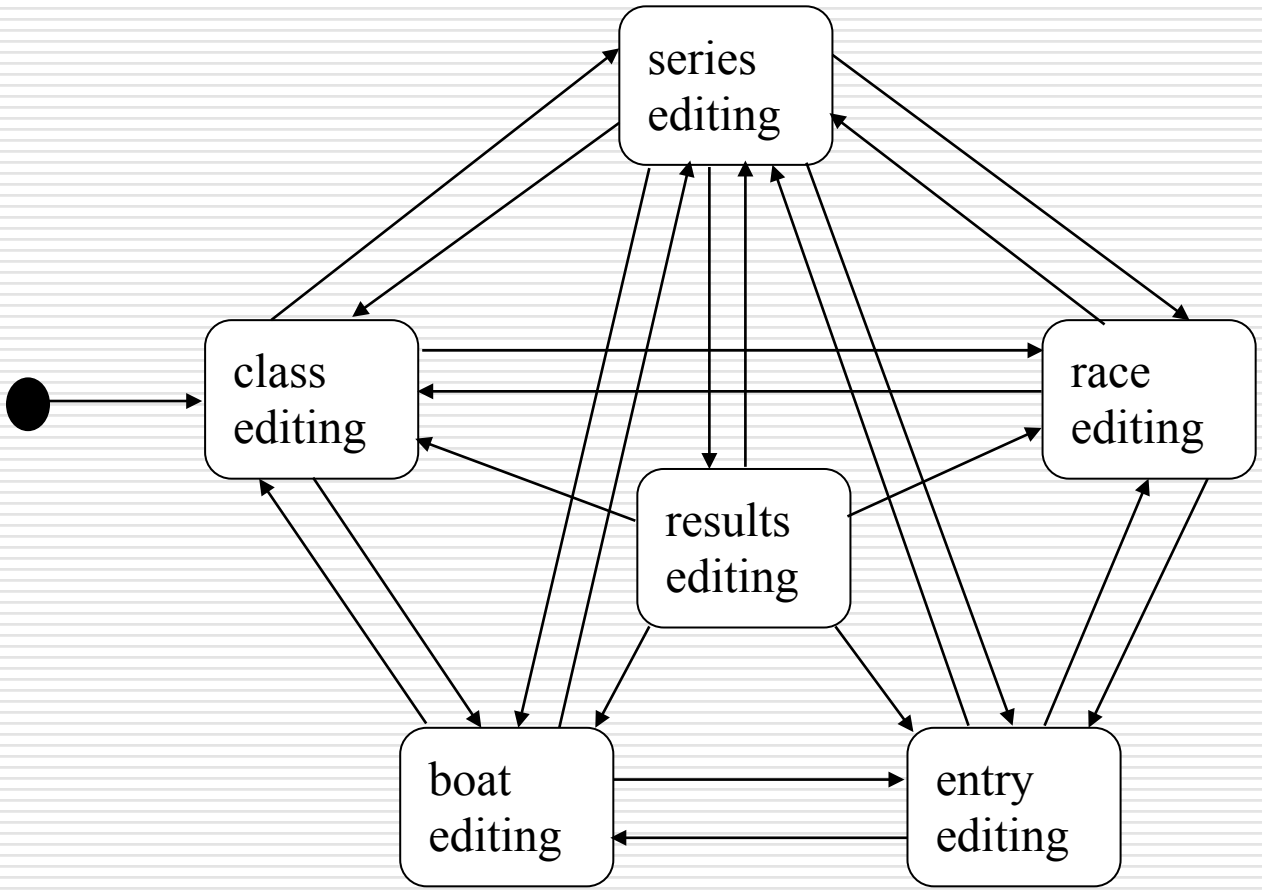
TM00.3.4 – Diagrammes d'états généralisés
Hiérarchisation



TM00.3.4 – Diagrammes d'états généralisés
Notations équivalentes

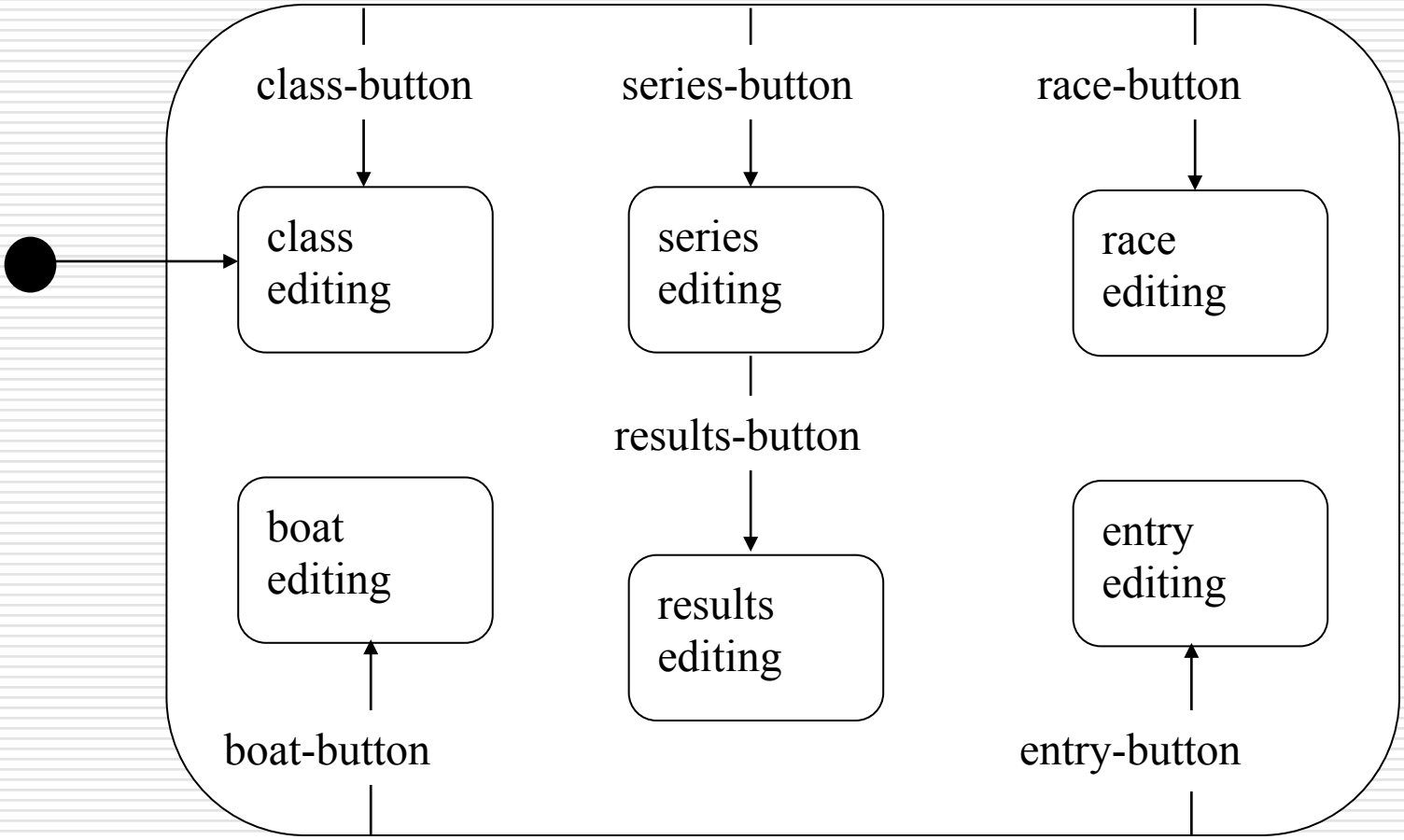


TM00.3.4 – Diagrammes d'états généralisés
Exemple - édition

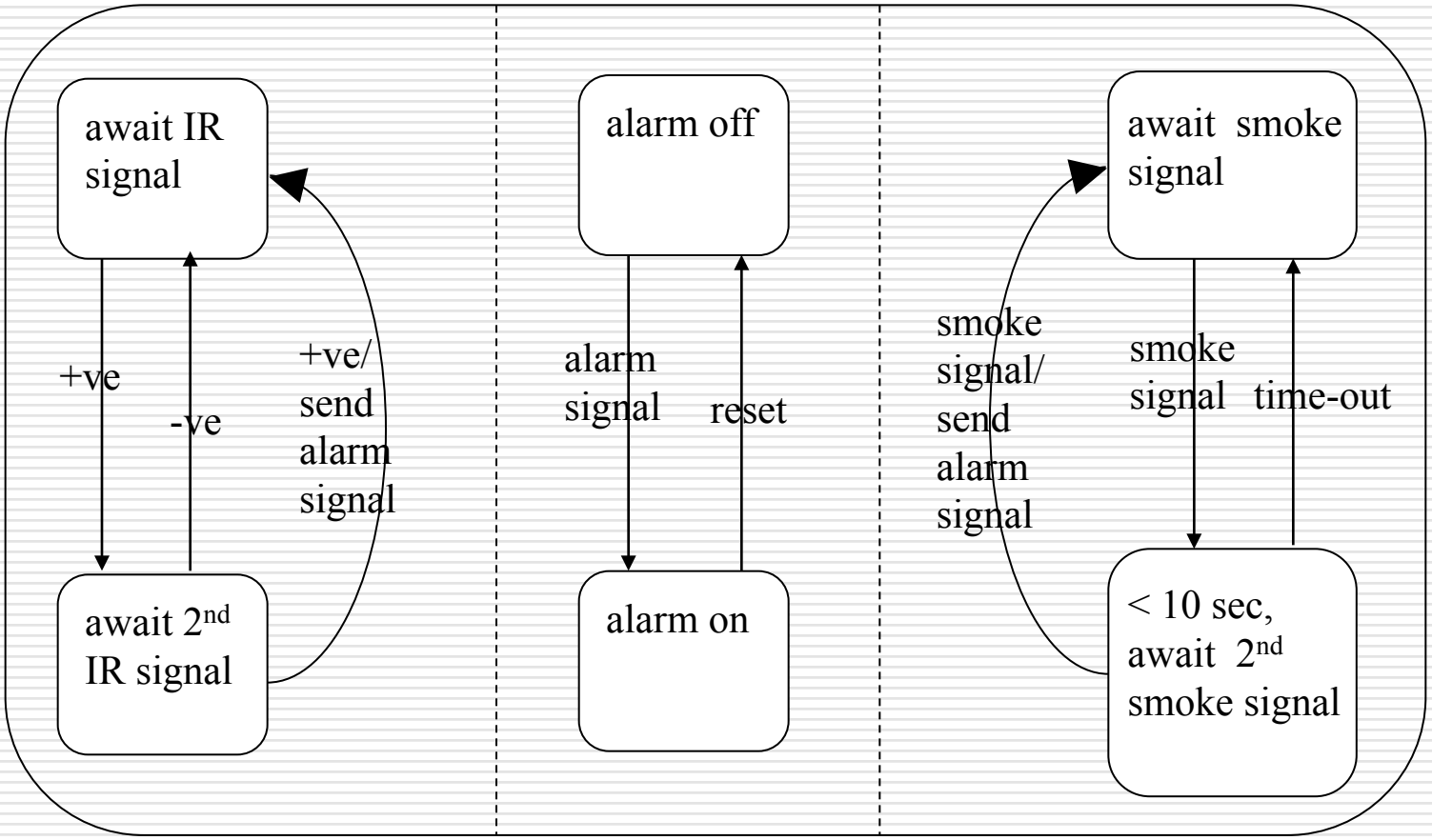


TM00.3.4 – Diagrammes d'états généralisés

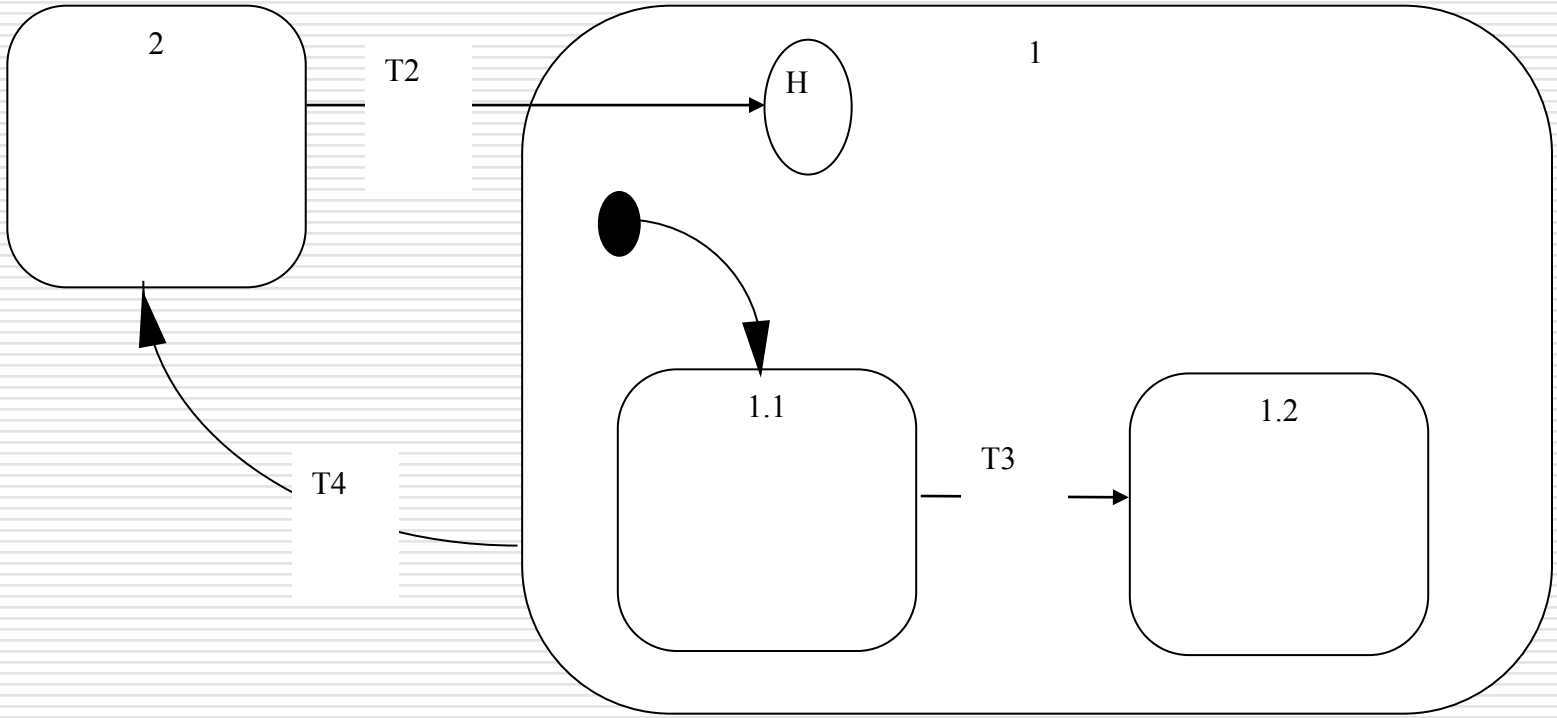
Exemple – édition avec hiérarchisation



TM00.3.4 – Diagrammes d'états généralisés
Concurrence



TM00.3.4 – Diagrammes d'états généralisés
Historique



TM00.3.4 – Diagrammes d'états généralisés (DEG)

Exercice 9

- Donnez un DEG pour le système de guichet bancaire
- Donnez un DEG pour le POTS

- POTS : « *Plain old telephone system* » le téléphone avant le numérique, le cellulaire, les services ajoutés… et les pannes!

TM00.3.4 – Diagrammes d'états généralisés (DEG)

Exercice 10

- À l'aide d'un DEG, spécifiez un four micro-onde ayant les caractéristiques suivantes :
  - trois niveaux de température (low, medium, high)
  - chronomètre
  - fonction de dégel (3 étapes de durée égale : la première à high, la deuxième à medium et la dernière à low)
  - programmation jusqu'à 3 étapes où l'utilisateur peut spécifier la température et la durée
  - définissez les touches appropriées et réalistes pour permettre à l'utilisateur d'utiliser ces fonctions
  - par souci de simplicité, ne spécifiez pas l'affichage à l'écran; spécifiez seulement les actions de contrôle pour l'élément chauffant et les alarmes sonores (fin de cuisson ou fin d'une temporisation, par exemple)
  - décrivez le traitement des actions sous forme de pseudo-code quand cela est nécessaire
  - déclarez les variables d'état que vous utilisez dans les actions

TM00.3.4 – Diagrammes d'états généralisés (DEG)

Exercice 11

- Utilisez un DEG pour spécifier un aspect de votre projet de session
- Décrivez le problème sous forme textuelle et donnez le DEG
- La taille du problème doit être similaire à celle du micro-onde
- Soyez suffisamment précis pour qu'un observateur externe qui ne connait pas du tout le problème puisse interpréter votre spécification sans ambiguïté

TM00.4 – Réseaux de Petri

Références

- Bray section 12.7 - pages 283 à 290

TM00.4 – Réseaux de Petri
Présentation

- Défini par Carl Petri (1962)
- Orienté état-transition, mais distinct des automates et machines de Mealy
- Graphe formé de
  - ensemble de places P
  - ensemble de transitions T
  - marquage initial à l'aide de jetons

TM00.4 – Réseaux de Petri
Définition

- Un réseau de Petri R est un tuple (P, T, Pré, Post) où :
  - P est l'ensemble des places;
  - T est l'ensemble des transitions;
  - Pré ::= P x T -> N, où Pré(p,t) est l'étiquette de l'arc allant de p à t; elle dénote le nombre minimal de jetons nécessaires dans p pour franchir t
  - Post ::= P x T => N, où Post(p,t) est l'étiquette de l'arc allant de t à p; elle dénote le nombre de jetons ajouté à p après le franchissement de t

TM00.4 – Réseaux de Petri

Marquage

- Un marquage indique le nombre de jeton de chaque place
  - M ::= P -> N

TM00.4 – Réseaux de Petri

Transition

- Une transition t peut être franchie ssi pour chaque arc entrant, le nombre de jetons de la place est supérieur ou égal de jetons de l'arc

$$\forall\, p \in P : Pre(p,t) >= M(p)$$

TM00.4 – Réseaux de Petri

Franchissement d'une transition

- Le franchissement d'une transition t entraîne un nouveau marquage M' tel que

$\forall\ p \in P$:

$$M'(p) = M(p) - Pré(p,t) + Post(p,t)$$

TM00.4 – Réseaux de Petri
Franchissement simultané

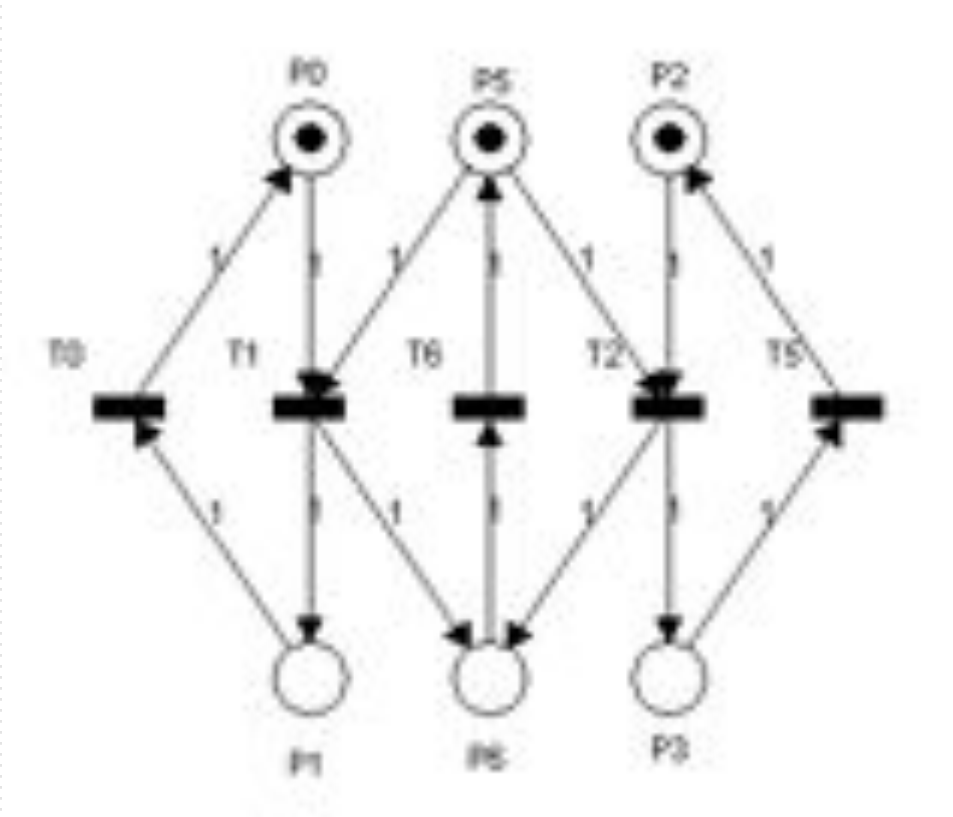
- Si plusieurs transitions peuvent être franchies, elles sont franchies dans un ordre non déterministe, certaines pouvant l'être simultanément (voire toutes)
- Il y a conflit si plus d'une transition peut être franchie pour une même place d'origine
- En cas de conflit, la sélection des transitions franchies est non déterministe

TM00.4 – Réseaux de Petri

Outil de simulation de réseau de Petri

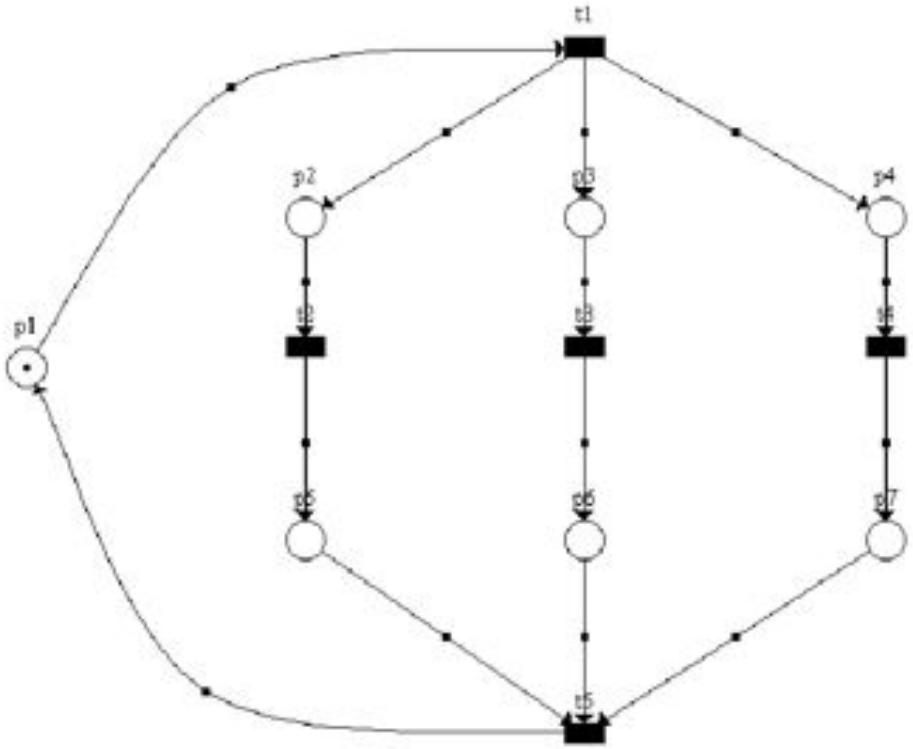
- HP Sim
  - permet de créer des réseaux de Petri et de simuler leur exécution
  - disponible sur le serveur

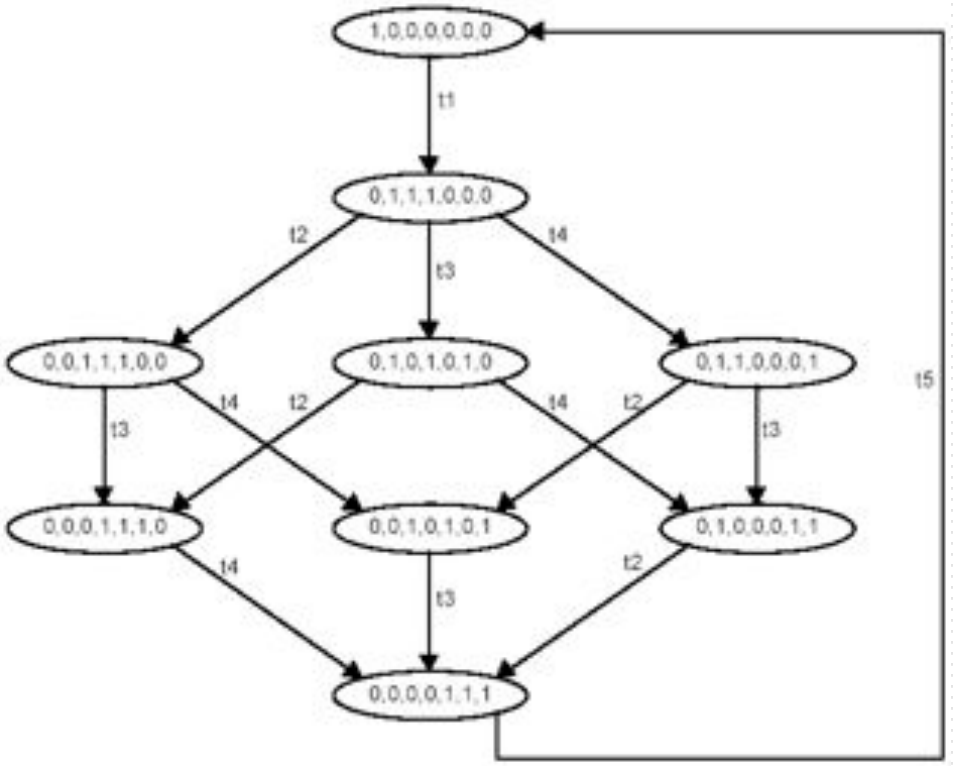
TM00.4 – Réseaux de Petri
Exemple 1



TM00.4 – Réseaux de Petri

Exemple 2





TM00.4 – Réseaux de Petri

Exercices

Donnez un réseau de Petri pour les concepts suivants

- fork
- join
- meet (synchronisation)
- message asynchrone
- appel-retour asynchrone
- exclusion mutuelle
- compteur
- M écrivains N lecteurs
- tampon borné
- protocole du bit alterné

TM00.4 – Réseaux de Petri
TP5 - partie 1

- Donnez un réseau de Petri modélisant le service de prêts avec réservation d'une bibliothèque
- Accompagnez votre texte d'une discussion sur les forces et les faiblesses des réseaux de Petri comme outil de modélisation

TM00.4 – Réseaux de Petri
TP5 - partie 2

- Complétez le réseau de Petri du protocole du bit alterné
  - ajouter la perte de message (un message de données ou bien un ack)
  - l'émetteur détecte la perte en utilisant un timeout

TM00.5 – Diagrammes d'activités

Références

- Leffingwell, chapitre 24
- Tout cours d'informatique des années 1950 à 1976…
- Une vieille notation ressuscitée par UML
- Elle était autrefois connues sous les noms d'ordinogramme et d'organigramme

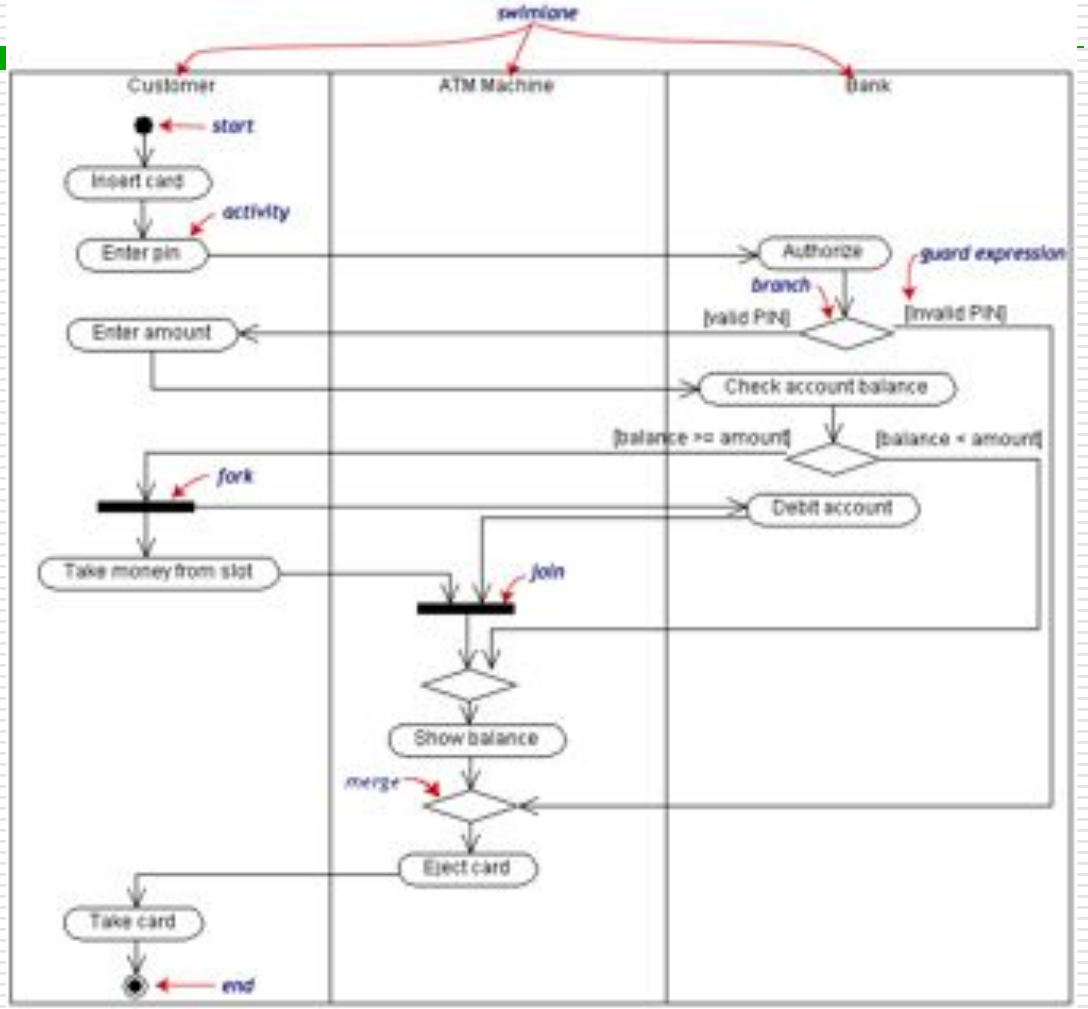
TM00.5 – Diagrammes d'activités

Notation

- notation standard
- +
- fork
- join

TM00.5 – Diagrammes d'activités

Exemple



TM00.6 – Pseudo-code déterministe « à la Wirth »

```
<code> ::=
    début <énoncés> fin
<énoncés> ::=
    { <énoncé> }
<énoncé> ::=
      <commande>
    | <conditionelle>
    | <itération>
    | <sélection>
    | <quantification>
<conditionnelle> ::=
    si <condition> alors
        <énoncés>
    sinon
        <énoncés>
<itération> :==
    tant que <condition> faire
        <énoncés>
<sélection> ::=
    choisir <expression> parmi
        { <garde> : <énoncés> }
    autrement
        <énoncés>
<quantification> ::=
    pour <id> dans <ensemble> faire
        <énoncés>
<commande> :==
    « une directive non ambigüe »
<expression> :==
    « une expression arithmétique non ambigüe »
<condition> :==
    « une expression booléenne non ambigüe »
<garde> :==
    <ensemble>
<id> :==
    « une variable libre »
<ensemble> ::=
    « un ensemble de valeurs non ambigu »
```

TM00.6 – Pseudo-code
non déterministe « à la Dijkstra »

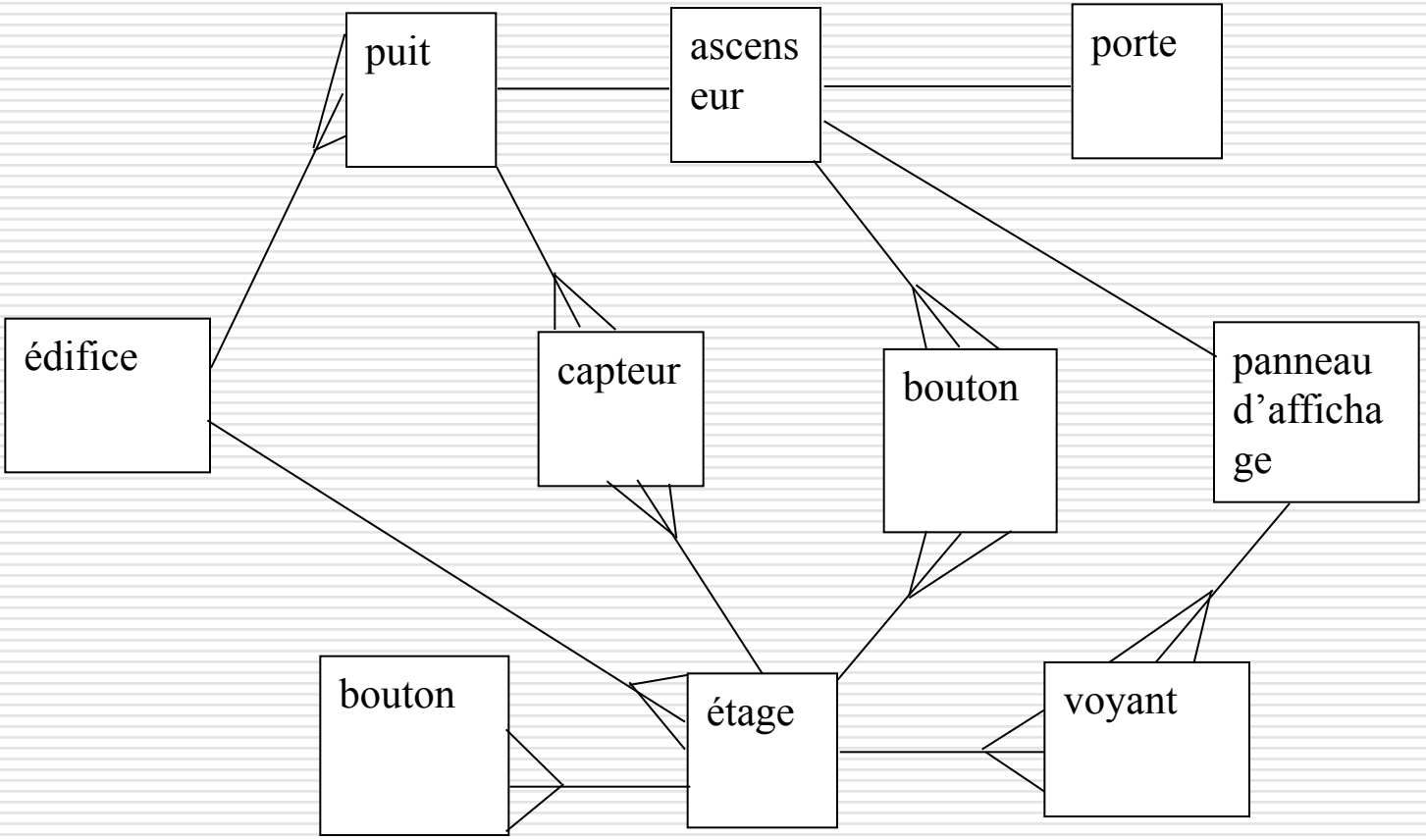
<code> ::=
début <énoncés> **fin**
<énoncés> ::=
{ <énoncé> }
<énoncé> ::=
<commande>
| <itération>
| <sélection>
<sélection> ::=
if { <garde> **:** <énoncés> } **fi**
<itération> ::=
do { <garde> **:** <énoncés> } **od**
<commande> :==
« une directive non ambigüe »
<garde> :==
« une expression booléenne non ambigüe »

TM00.7 – MCD
Références

- ...

TM00.7.1 - DER

Exemple de MCD de type ER (Chen)



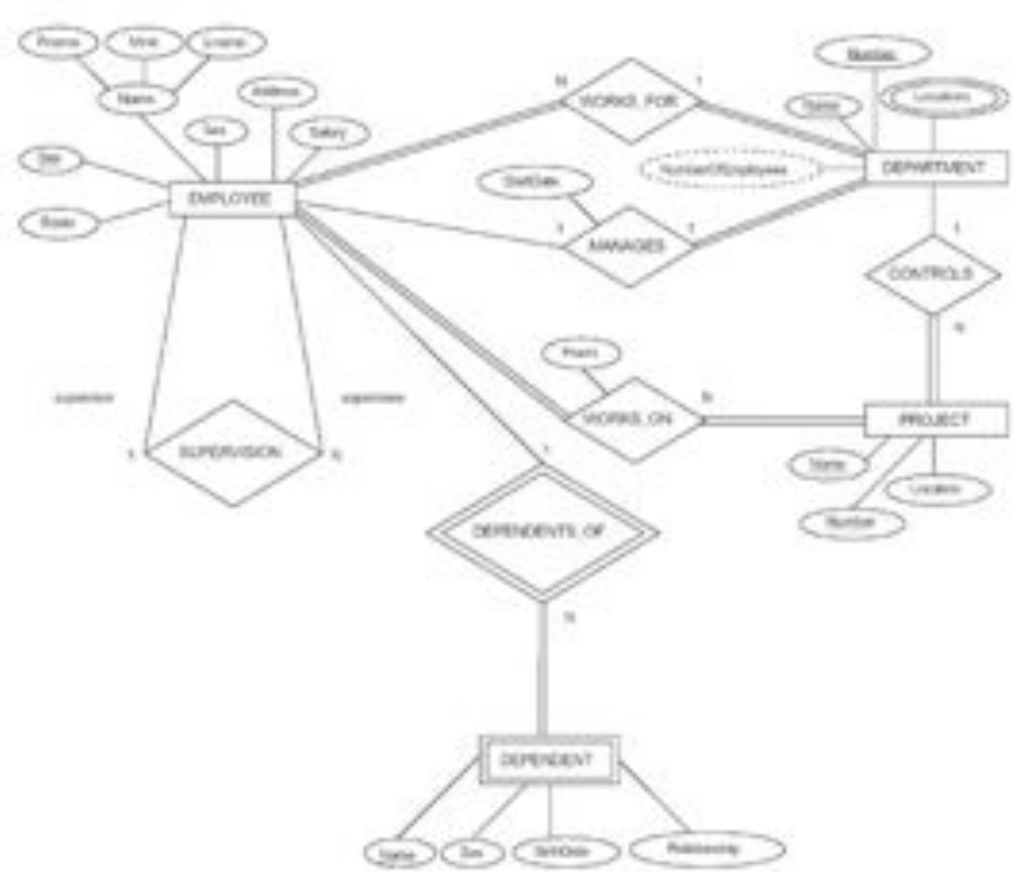
TM00.7.1 - DER

Exemple de MCD de type ER (Chen)

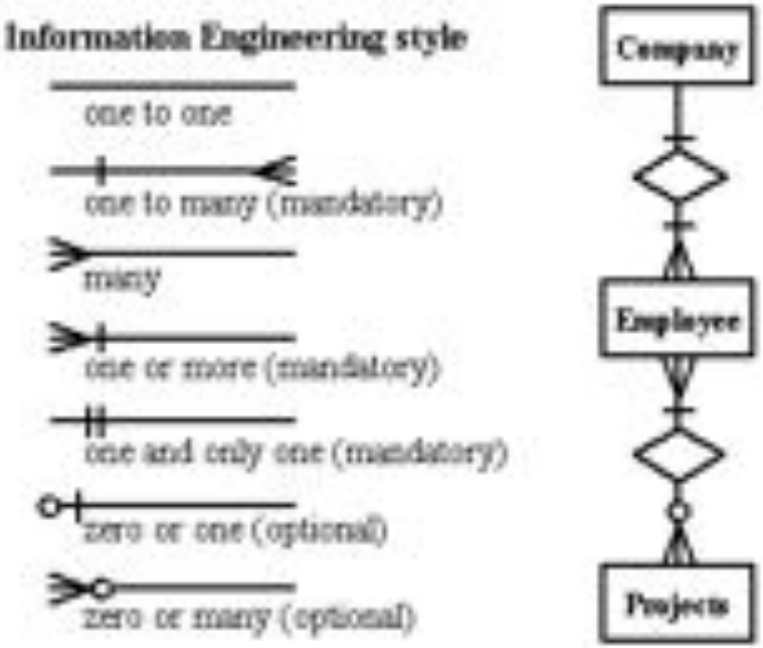
- où est le panneau d'affichage de l'étage?
- qu'est-il arrivé aux portes de l'étage?
- faut-il vraiment distinguer les boutons à l'étage et ceux dans l'ascenseur?

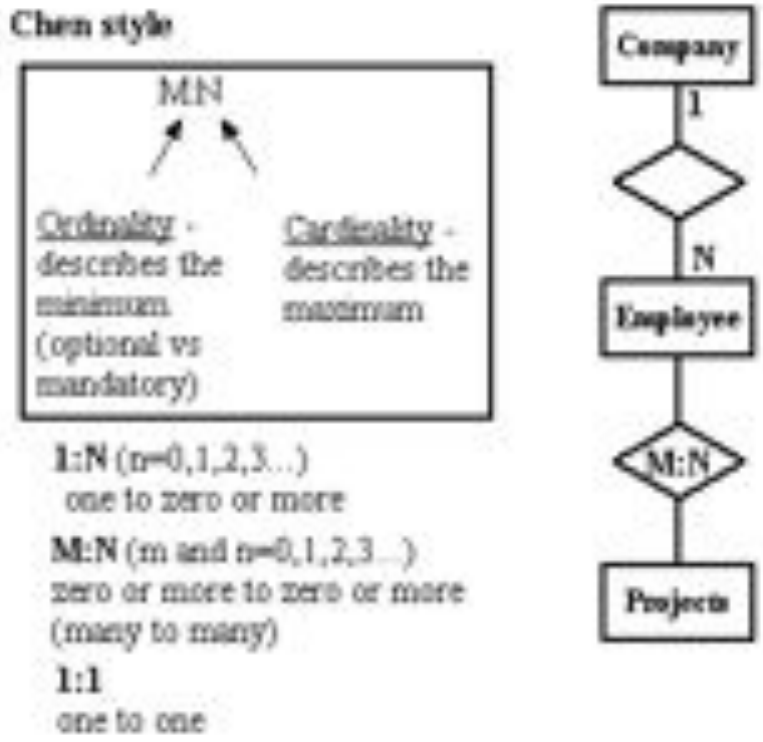
TM00.7.1 - DER

Exemple de MCD de type ER (Yourdon)



TM00.7.1 - DER
Notation diagrammes ER

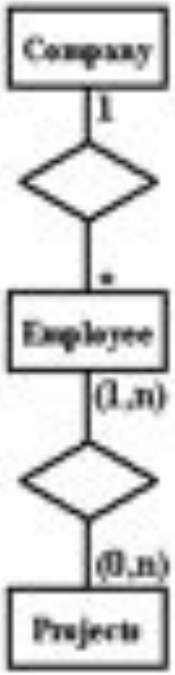




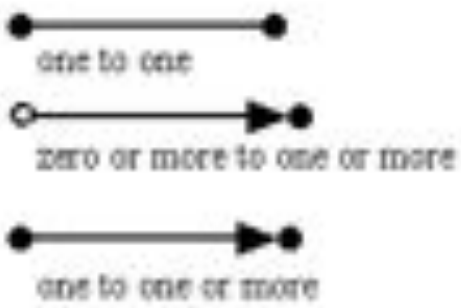
TM00.7.1 - DER
Notation diagrammes ER

Martin style

1 - one, and only one (mandatory)
***** - many (zero or more - optional)
1...* - one or more (mandatory)
0...1 - zero or one (optional)
(0,1) - zero or one (optional)
(1,n) -one or more (mandatory)
(0,n) - zero or more (optional)
(1,1) - one and only one (mandatory)



Bachman style



TM00.7.1 - DER
Notation étendue

- Notation d'Abrial
- Introduction des concepts
  - d'union (disjointe ou conjointe),
  - d'intersection
- Voir Elmasri

UNIVERSITÉ DE SHERBROOKE

TM00.7.2 – UML-C
Présentation

- ...

TM00.7.3 - DFD
Notation

- Diagrammes de flux de données
  - plusieurs variantes de notation (Gane & Sarson, Yourdon, SSADM)
- Décomposition hiérarchique des fonctions
- Représentation des fonctions (services)
  - interfaces (entrées et les sorties)
  - traitement
  - dépôt de données
  - flux (origines et destinations)

TM00.7.3 - DFD
Recette

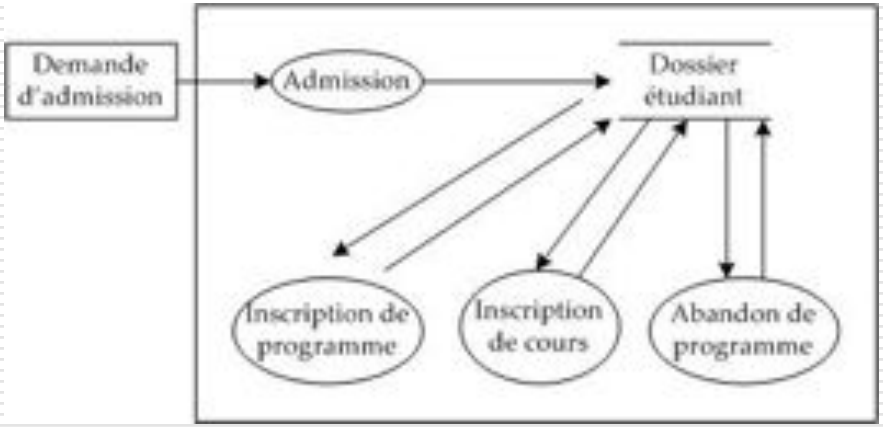
- le DFD0 est le diagramme de contexte
- s'assurer de rendre compte de toutes les entrées et toutes les sorties
- un affinement de niveau ne doit porter que sur un processus, toutes ses interactions doivent être reportées
- tous les symboles (acteurs, processus, dépôt, flux) doivent être étiquetés

TM00.7.3 - DFD
Niveau d'abstraction

- En général, il est souhaitable de ne décomposer que les « services »
  - un service est un processus qui ne peut être spécifié qu'en le décomposant en processus plus élémentaires
  - une fonction est un processus qui peut être spécifié sans être décomposé
- En conséquence
  - la spécification d'un service nécessite le développement d'un DFD de niveau plus élevé
  - la spécification d'une fonction est exprimé par un pseudo-code

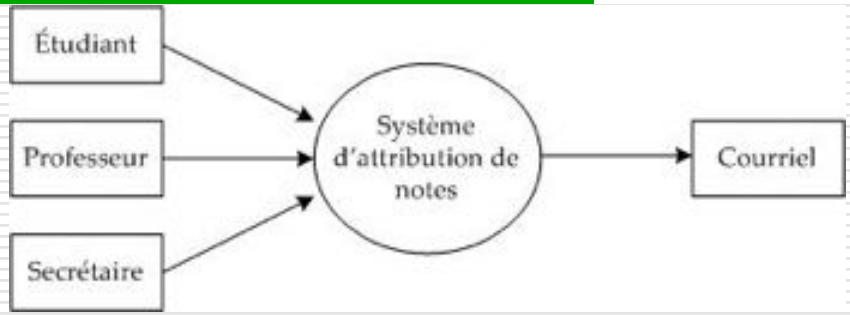
TM00.7.3 - DFD

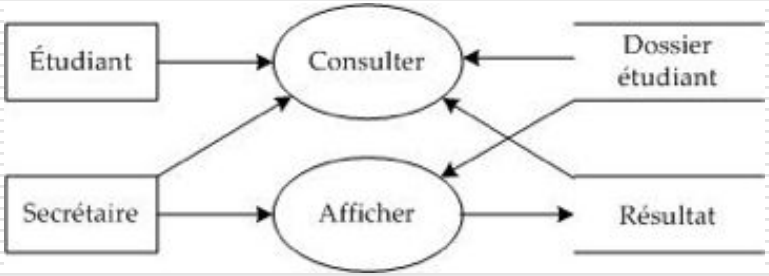
Exemple - saisie de notes, niveaux 0 et 1

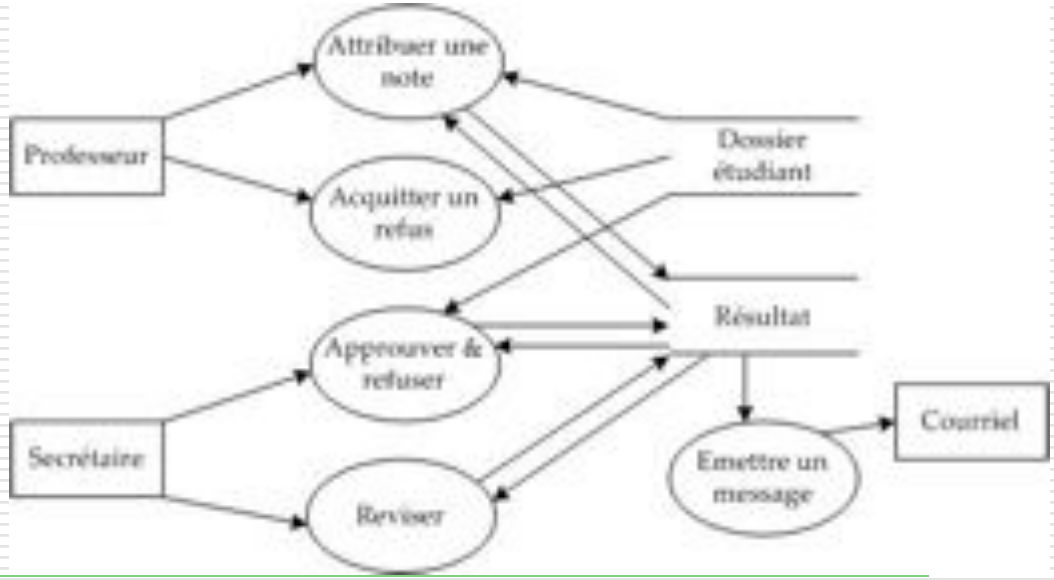


TM00.7.3 - DFD

Exemple - saisie de notes, niveau 2

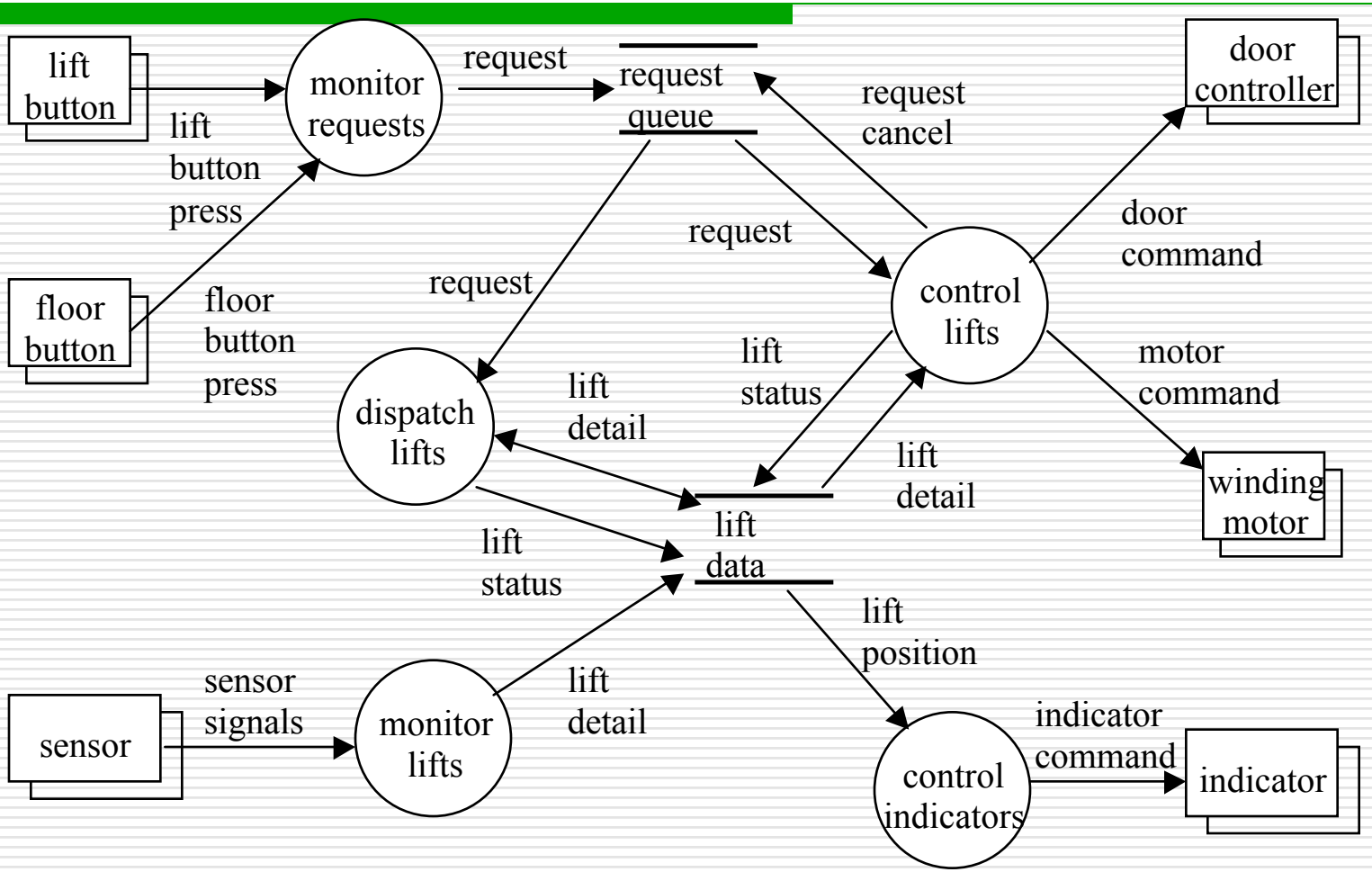




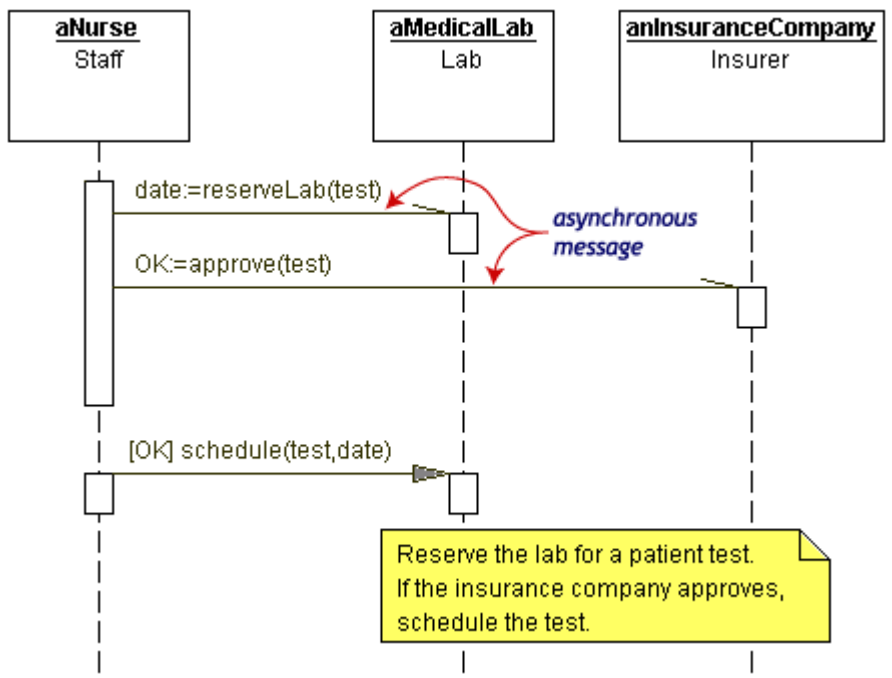


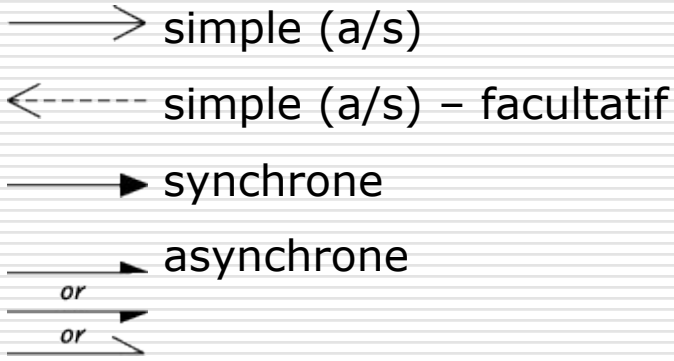
TM00.7.3 - DFD

Exemple – ascenseur (compléter et corriger)

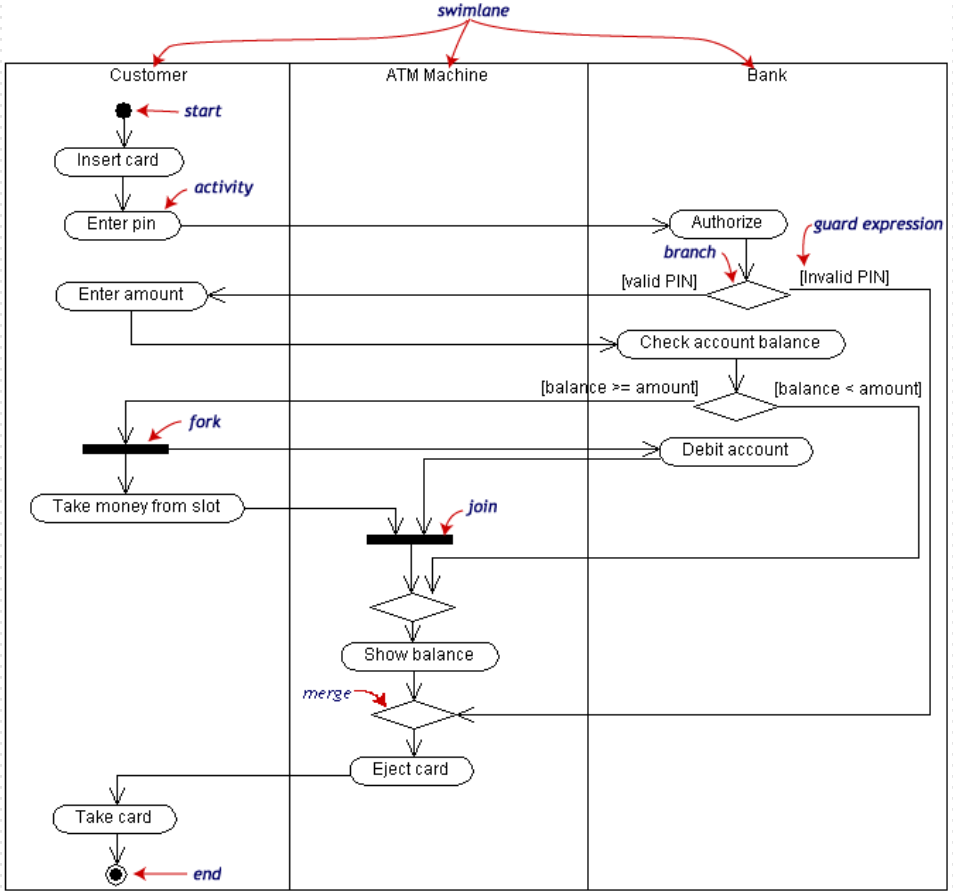


TM00.7.4 – UML-S
Diagramme de séquence





TM00.7.5 – UML-A
Diagramme d'activités



UNIVERSITÉ DE SHERBROOKE

TM00.7.6 – UML-T

Diagramme de transition

- Voir [UML]

TM00.7.7 – UML-X
Diagramme de collaboration

- Voir [UML]

TM00.8 – Maquettages et prototypage

Références

- Voir IFT 215

TM00.9 – Divers

Références

- Diagrammes de structures (DS)
- Grammaires de tâches (GT)
- Diagrammes d'états de Jackson (JSD)

à suivre...