

# Introduction aux automates

## *Principes et exemples*

Pascal André

IRIN

Université de Nantes

2 rue de la Houssinière ; B.P. 92208

44322 Nantes Cedex 03

Pascal.Andre@irin.univ-nantes.fr

# Plan de l'exposé

1. Les notions de base
2. Les propriétés
3. Modélisation de base
4. Extensions

# Plan de l'exposé

1. Les notions de base
2. Les propriétés
3. Modélisation de base
4. Extensions

# Les notions de base

Machine à états et variante [AV01]

## Définition 1.1 (Automate)

*Un **automate** est un quadruplet  $M = (\Sigma, K, \delta, K_0)$  où  $\Sigma$  (l'alphabet d'entrée) est un ensemble non vide,  $K$  est un ensemble non vide (les états),  $\delta : (K \times \Sigma) \leftrightarrow K$  la relation de transition,  $K_0 \subseteq K$  l'ensemble des états initiaux.*

## Définition 1.2 (système de transitions)

*Un **système de transitions** est défini par un ensemble d'états, un ensemble de transitions, une fonction d'origine et une fonction de destination. Ces deux fonctions délivrent un état à partir d'une transition. Un système de transition étiqueté associe une étiquette à chaque transition.*

# Les notions de base

## Définition 1.3 (Chemin, Trace)

*Un **chemin** est une suite acceptable de transitions. La **trace** d'un chemin est la suite des entrées qui lui sont associées.*

Bon nombre de propriétés des systèmes de transitions sont exprimées sur les traces (en utilisant des expressions régulières par exemple).

# Les notions de base

## Définition 1.4 (entrée vide)

*Une entrée vide, notée  $\varepsilon$ , est parfois ajoutée à  $\Sigma$ . Elle représente l'absence d'étiquette sur la transition.*

Une transition étiquetée par  $\varepsilon$  est franchissable sans nécessiter l'occurrence d'une entrée. L'automate est alors forcément non-déterministe.

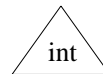
Les entrées peuvent être classées en internes/externes. Ceci correspond à une notion de visibilité ou d'observabilité dans les traces.

# Les notions de base

## Représentation d'états No 1



*initial*

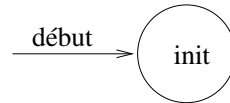


*intermédiaire*

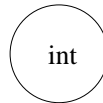


*final*

## Représentation d'états No 2 [ASU91]



*initial*

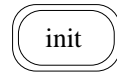


*intermédiaire*

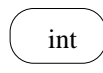


*final*

## Représentation d'états No 3 [Arn92]



*initial*

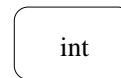


*intermédiaire*

## Représentation d'états No 4 (OMT)[RBP+91]



*initial*

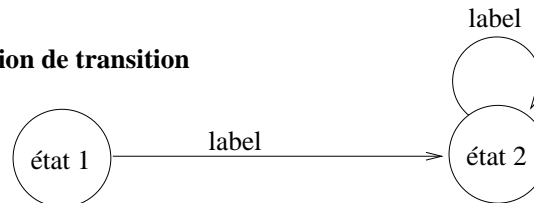


*intermédiaire*



*final*

## Représentation de transition



# Les notions de base

## Différentes sortes d'automates

1. Avec/sans états finaux
2. Avec/sans gardes, paramètres pour les entrées
3. Avec/sans fonction de sortie
4. Déterministe ou pas : la relation de transition devient une fonction
5. Typage des entrées, des sorties, des états
6. Composition et hiérarchisation des automates
7. Prise en compte du temps

# Plan de l'exposé

1. Les notions de base
2. **Les propriétés**
3. Modélisation de base
4. Extensions

# Les propriétés

## 1. Propriétés propres aux automates

- fini
- déterministe
- minimal
- équivalence

## 2. Propriétés liées à la dynamique des systèmes

- Accessibilité
- Réinitialisabilité
- Vivacité
- Blocage
- Famine, Equité

# Les propriétés

## Expression des propriétés

- Logique des prédicats
- Logiques temporelles
- Algorithmes
- etc.

# Plan de l'exposé

1. Les notions de base
2. Les propriétés
3. **Modélisation de base**
4. Extensions

# Modélisation de base

## Etats, transitions

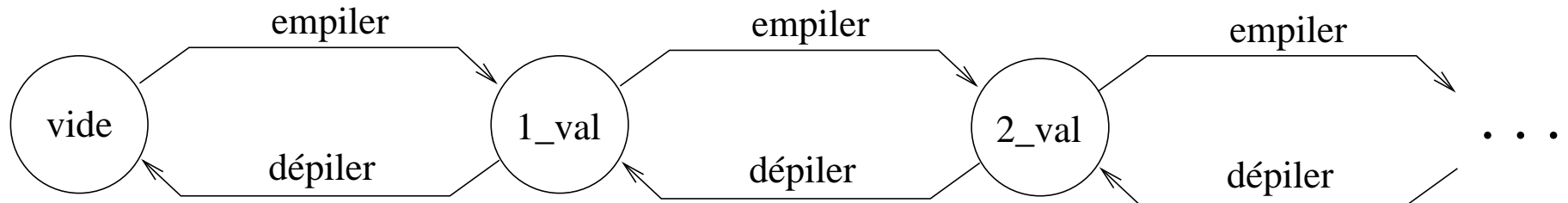


Figure 2 : *Automate de la pile non bornée*

1. Pour dépiler *i.e.* effectuer une transition étiquetée par `dépiler`, il faut avoir empilé auparavant *i.e.* être passé par une transition `empiler`.
2. Les empilements et dépilements se font dans un ordre absolument quelconque tant que la propriété 1 est vérifiée.

# Modélisation de base

Les deux propriétés précédentes (1) et (2) sont facilement démontrables par une analyse des traces de cet automate. Ce que nous ne démontrons pas, c'est que l'élément dépilé est le dernier élément empilé. Cet automate ne permet pas de savoir ce que contient la pile, ni quel est l'élément au sommet de cette pile. Tout ceci n'est pas grave, tant que le but est de montrer l'évolution de la pile dans le temps : son **comportement dynamique**.

# Modélisation de base

Le problème majeur de cette pile est qu'elle possède une infinité d'états. Une réponse consiste à borner le nombre d'états (le nombre d'emplacements de la pile).

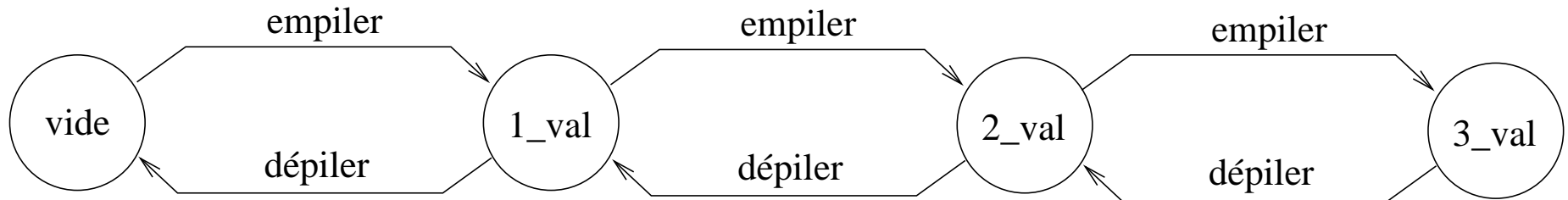


Figure 3 : *Automate de pile bornée à trois emplacements*

# Modélisation de base

Un autre problème est que nous ne connaissons pas la nature des éléments empilés. Une réponse consiste à enrichir le vocabulaire d'entrée en ajoutant pour toutes les valeurs possibles d'entiers  $val$  l'entrée  $empiler(val)$ .

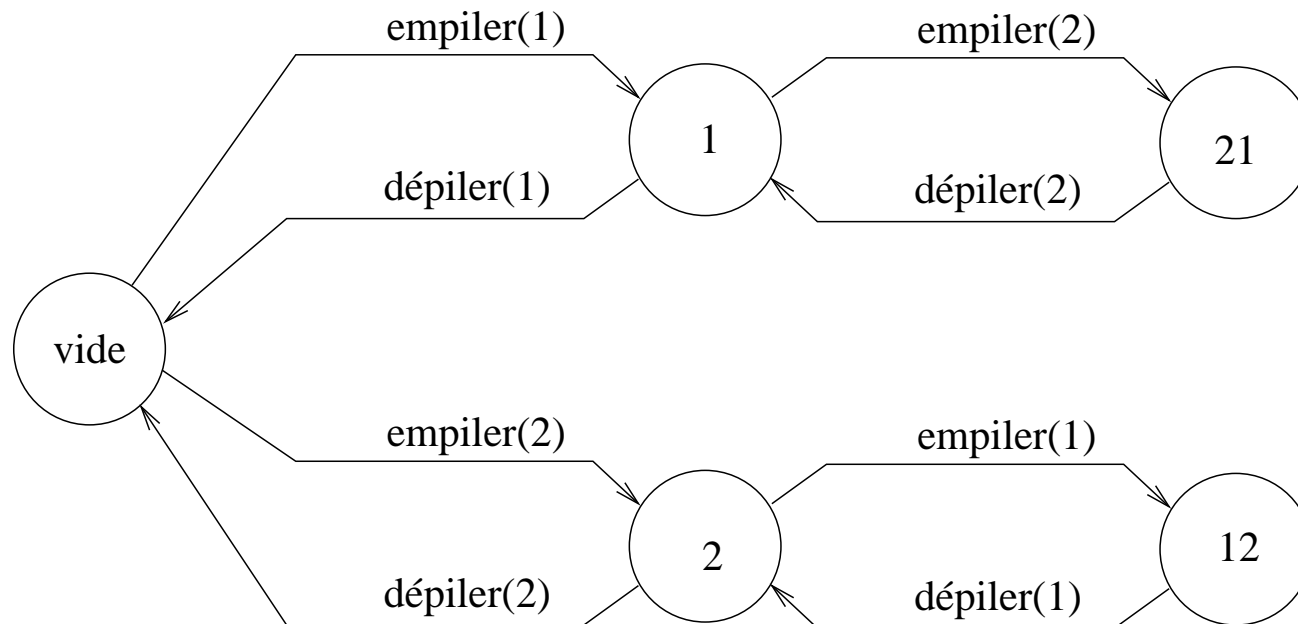


Figure 4 : *Automate de pile bornée d'entiers à deux emplacements*

# Modélisation de base

La figure précédente illustre cette méthode pour deux entiers. C'est la méthode utilisée par de nombreux outils pour vérifier les propriétés d'une spécification (*model checking*). Il est évident que l'utilisation de paramètres de type simples, comme les entiers naturels, engendre une explosion combinatoire du nombre d'états et de transitions. Une solution proposée est la vérification symbolique de modèles (*symbolic model checking*).

# Modélisation de base

## Limites

- Explosion combinatoire du nombre de transitions pour des valeurs différentes (exemple de la pile à deux emplacements) : **explosion verticale**.
- Explosion combinatoire du nombre d'états pour de gros systèmes : **explosion horizontale**.
- Pauvreté (relative) de la description :
  - Gardes
  - Typage des entrées
  - Expressions de sorties (traitements, événements)
  - Informations d'état
- Gérer la concurrence, gérer la complexité : composer des automates, hiérarchiser les états

# Plan de l'exposé

1. Les notions de base
2. Les propriétés
3. Modélisation de base
4. **Extensions**

# Extensions

Entrée générique

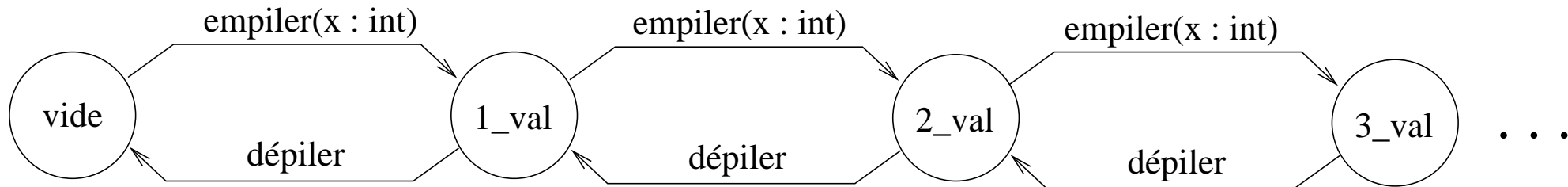


Figure 5 : *Automate de pile avec transition générique non bornée*

# Extensions

Une **garde** est un prédicat associé à une transition. Une transition est dite **passante** si la garde associée est instanciée à vrai dans l'état source.

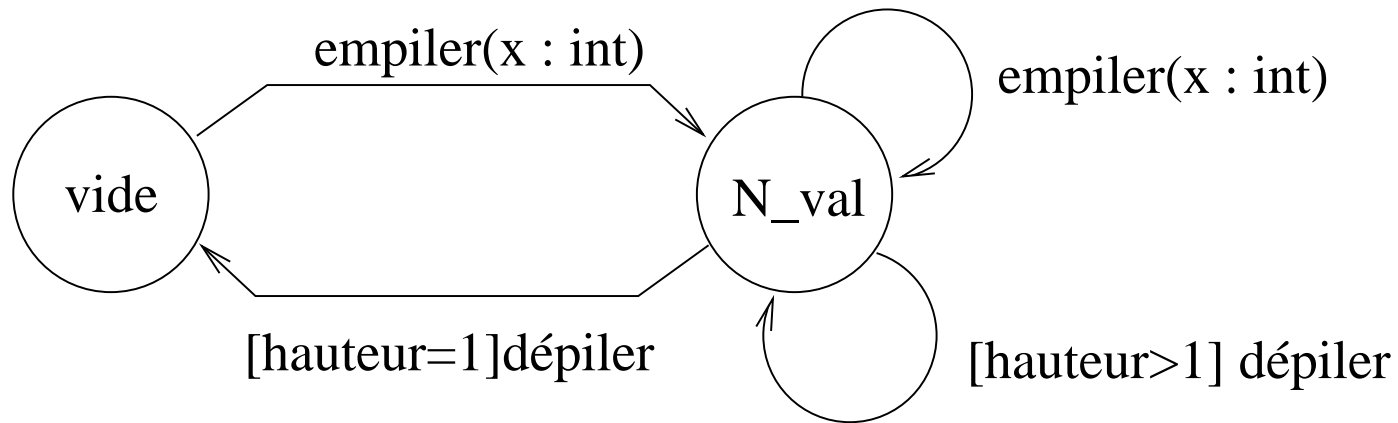
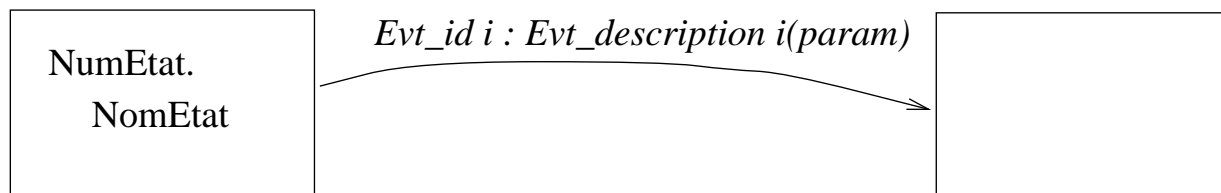


Figure 6 : *Automate de pile non bornée avec gardes*

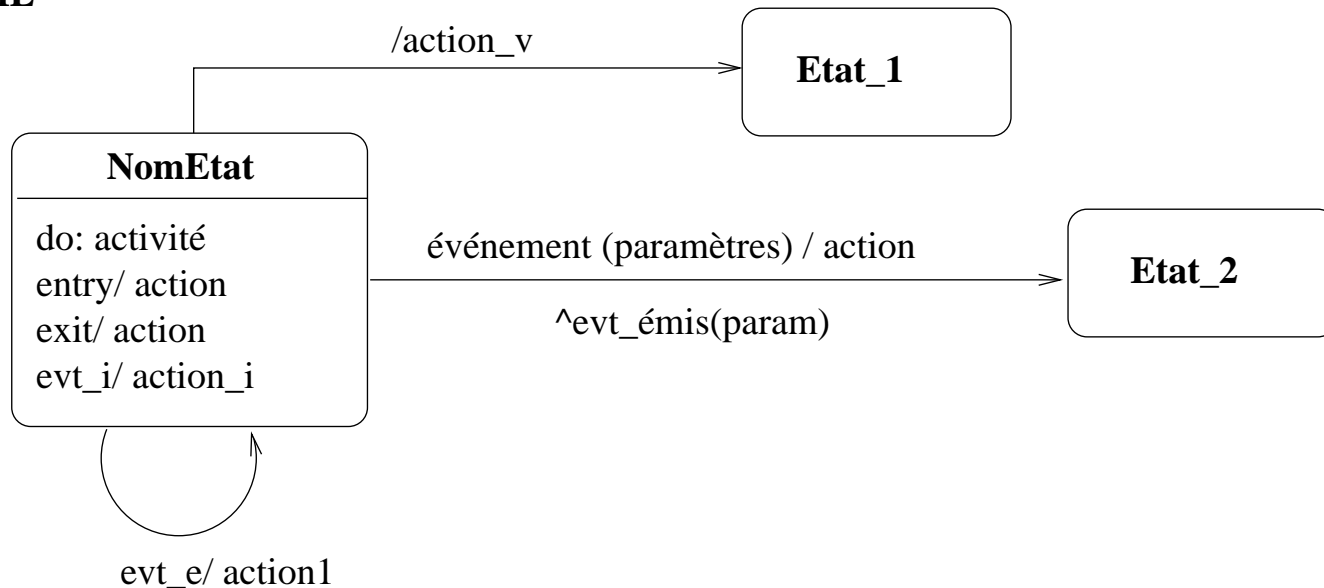
# Extensions

## Fonction de sortie SHLAER & MELLOR



Generate Evt\_id1 : Evt\_description1(param)  
Generate Evt\_id2 : Evt\_description2(param)

## UML



# Extensions

La fonction de sortie peut être associée aux :

- états - SHLAER & MELLOR. Les événements sont émis vers le système. Pas d'ordre dans les émissions.
  - transitions - SDL. Les événements sont émis vers des canaux de communications. Ordre possible.
  - états et transitions - UML. Les événements sont émis vers des objets dont la classe est précisée. Pas d'ordre.
- actions - activités.

1. Communication bipoint, diffusion
2. Synchronisation
3. Interne / externe

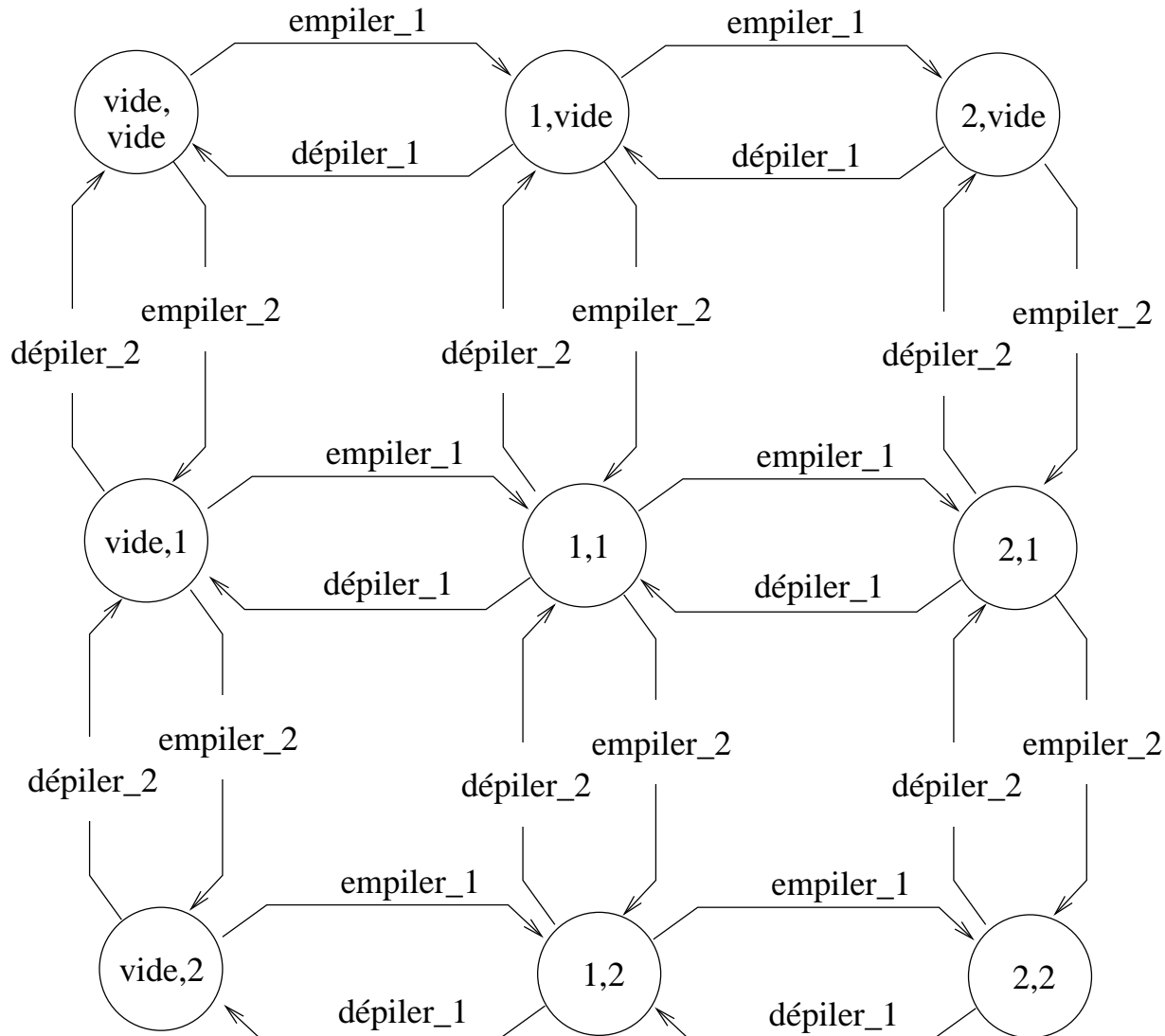
# Extensions

## Etats

1. Un état représente une étape dans le cycle de contrôle. Interprétation usuelle.
2. Un état est un processus. Activités
3. Un état est un automate. Structuration hiérarchique des automates (détails).
4. Un état résulte d'un produit d'automates concurrents (détails).
5. Un état est un ensemble de valeurs.

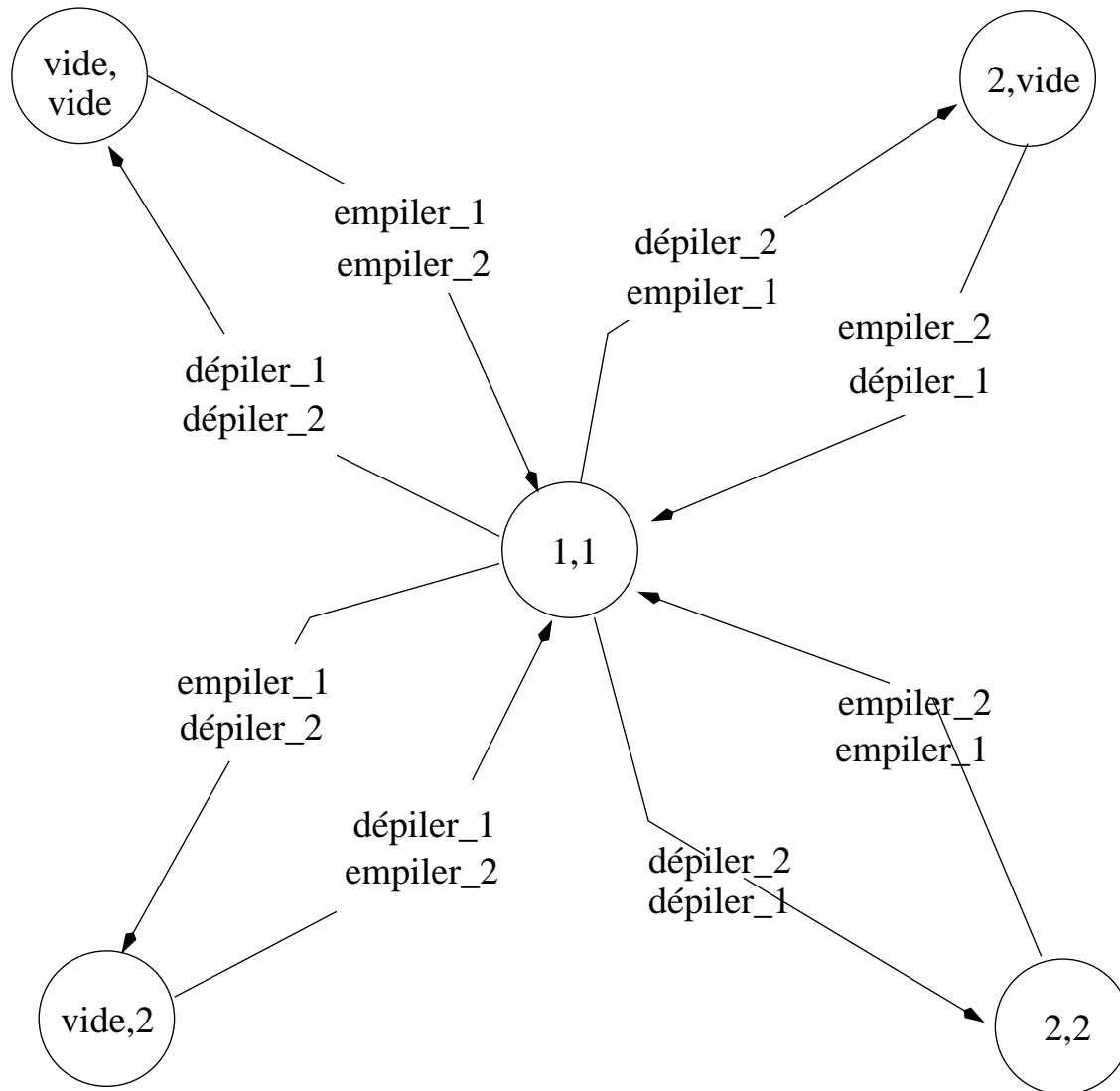
# Extensions - composition

Produit libre



# Extensions - composition

Produit cartésien



# Extensions - composition

Produit synchronisé 1/3

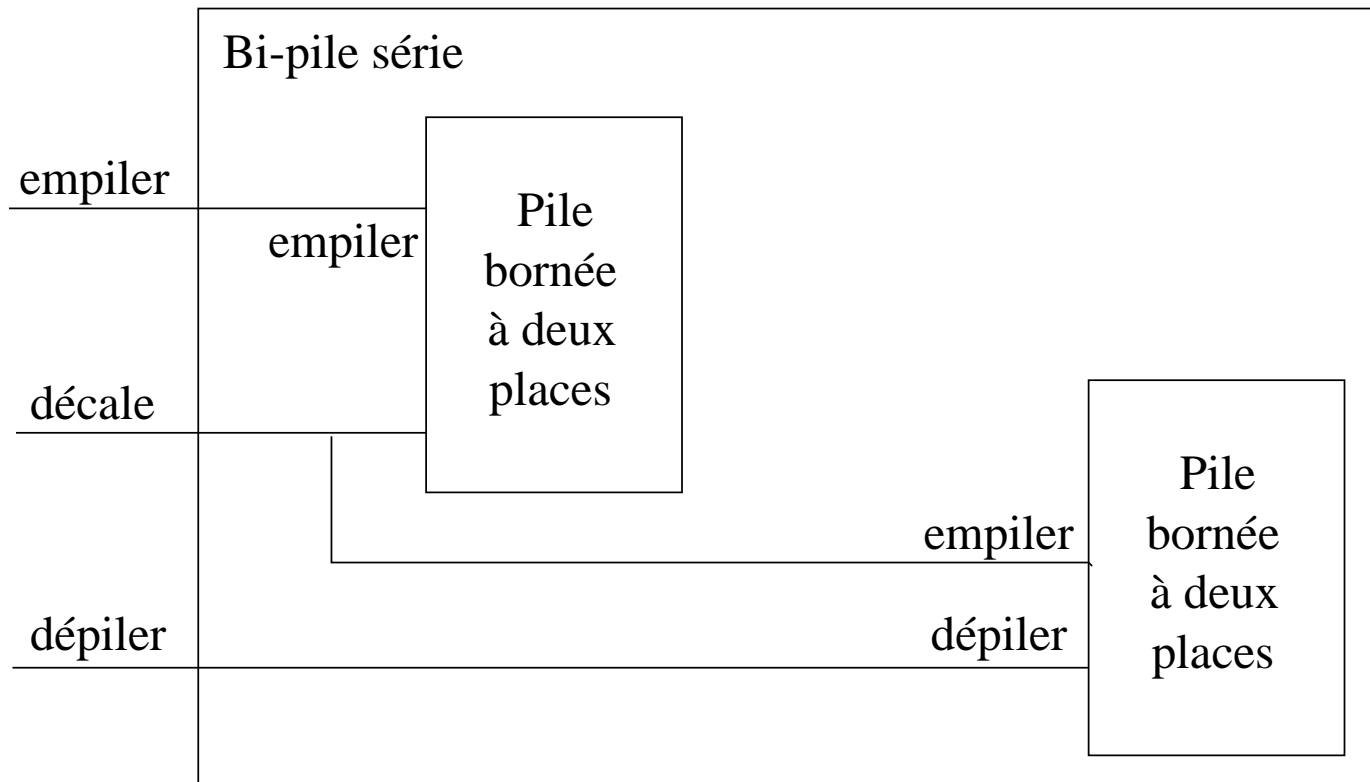


Figure 7 : *Piles en série*

# Extensions - composition

Produit synchronisé 2/3

Le vecteur de synchronisation s'écrit :

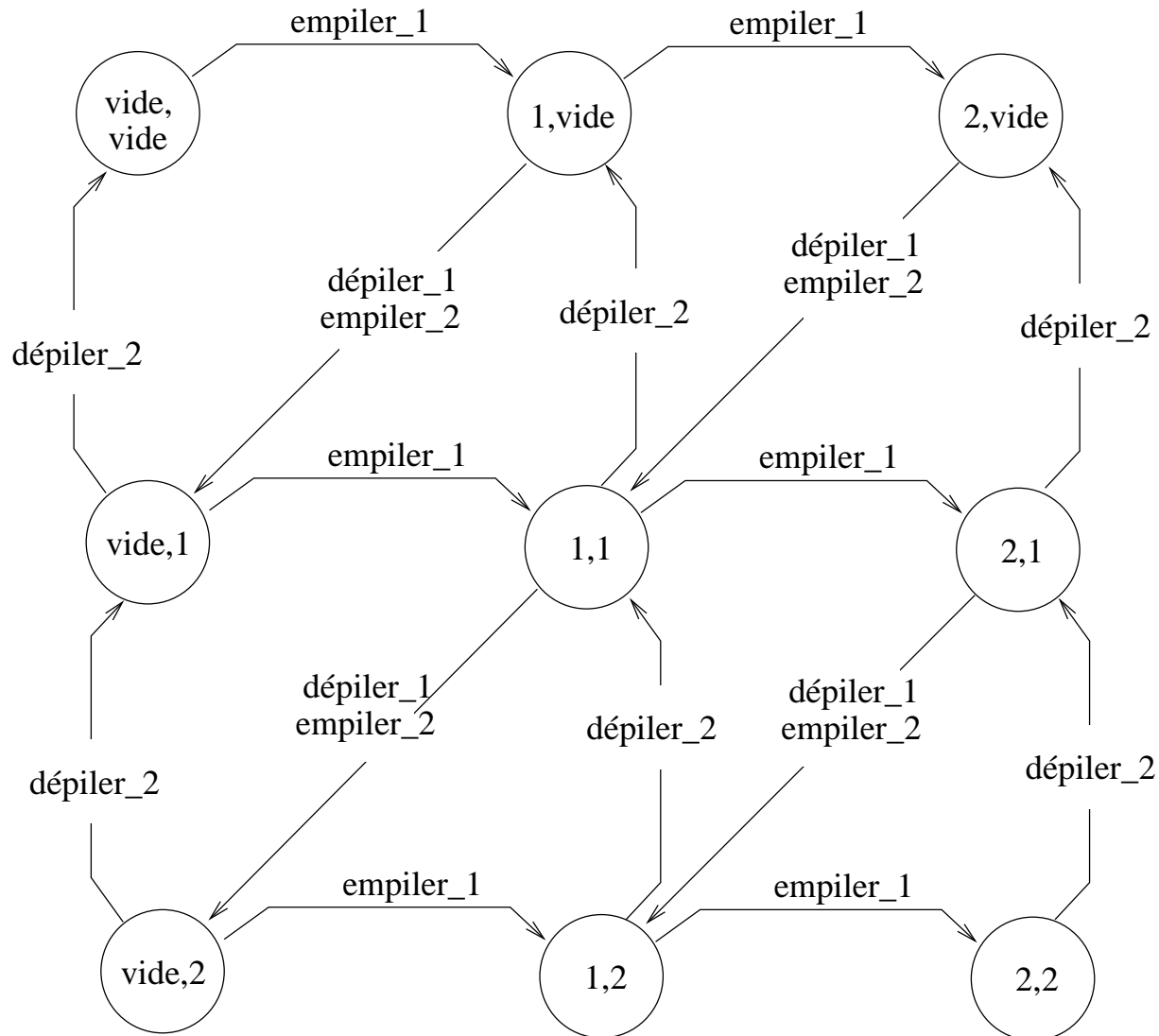
empiler :< empiler  $\varepsilon$ >

décaler :< dépiler empiler>

dépiler :<  $\varepsilon$  dépiler>

# Extensions - composition

## Produit synchronisé 3/3



# Extensions - communication

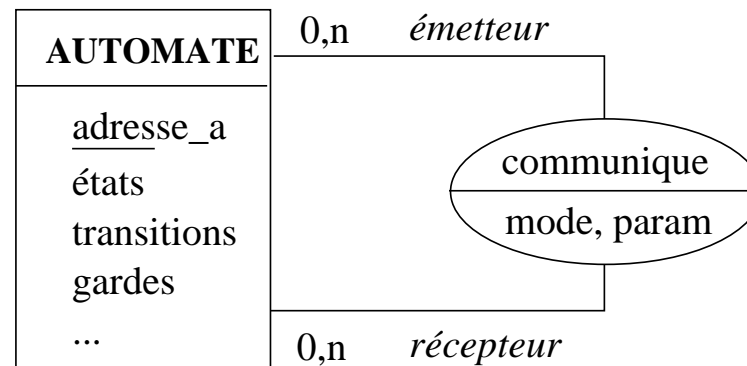
## Liaison 1/2

Emetteur	Récepteur	Commentaire
1	1	liaison <b>bi-point</b>
1	n	liaison <b>multi-point</b> ou <b>diffusion</b>
n	1 ou n	liaison <b>multiplexée</b>

# Extensions - communication

## Liaison 2/2

①



②

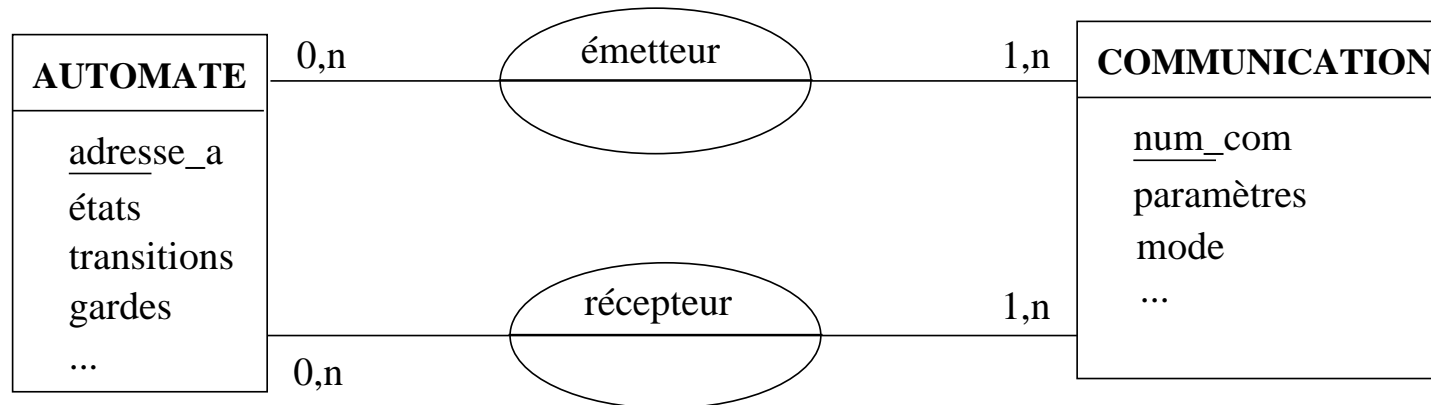


Figure 8 : *Représentation E/A de la communication*

# Extensions - communication

1. Mode de communication
  - (a) Synchrone.
  - (b) Asynchrone.
2. Identification des automates.
3. Passage de paramètres (cf fonctions d'entrée et de sortie)

# Extensions - hiérarchisation

*Statecharts* de Harel

Un état contient un ou plusieurs automates.

- Super-état, super-transition.
- Factorisation.

# Extensions - hiérarchisation

## Hiérarchique séquentiel

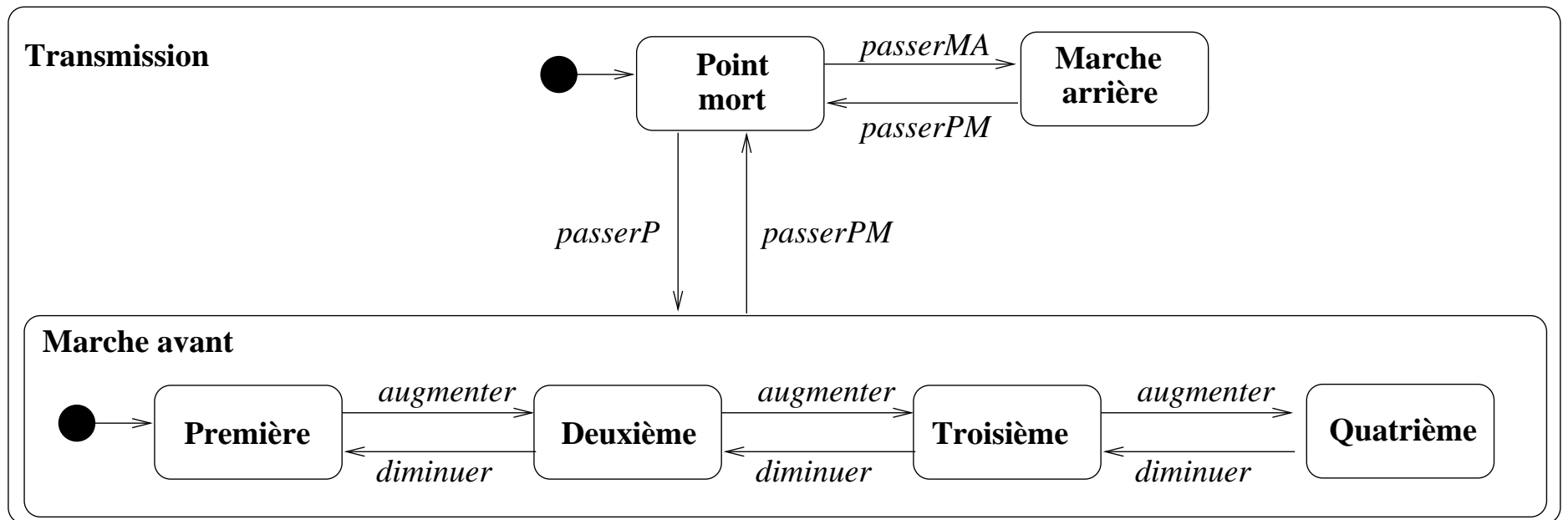


Figure 9 : Automate hiérarchique de la boîte de vitesse

# Extensions - hiérarchisation

Hiérarchique séquentiel

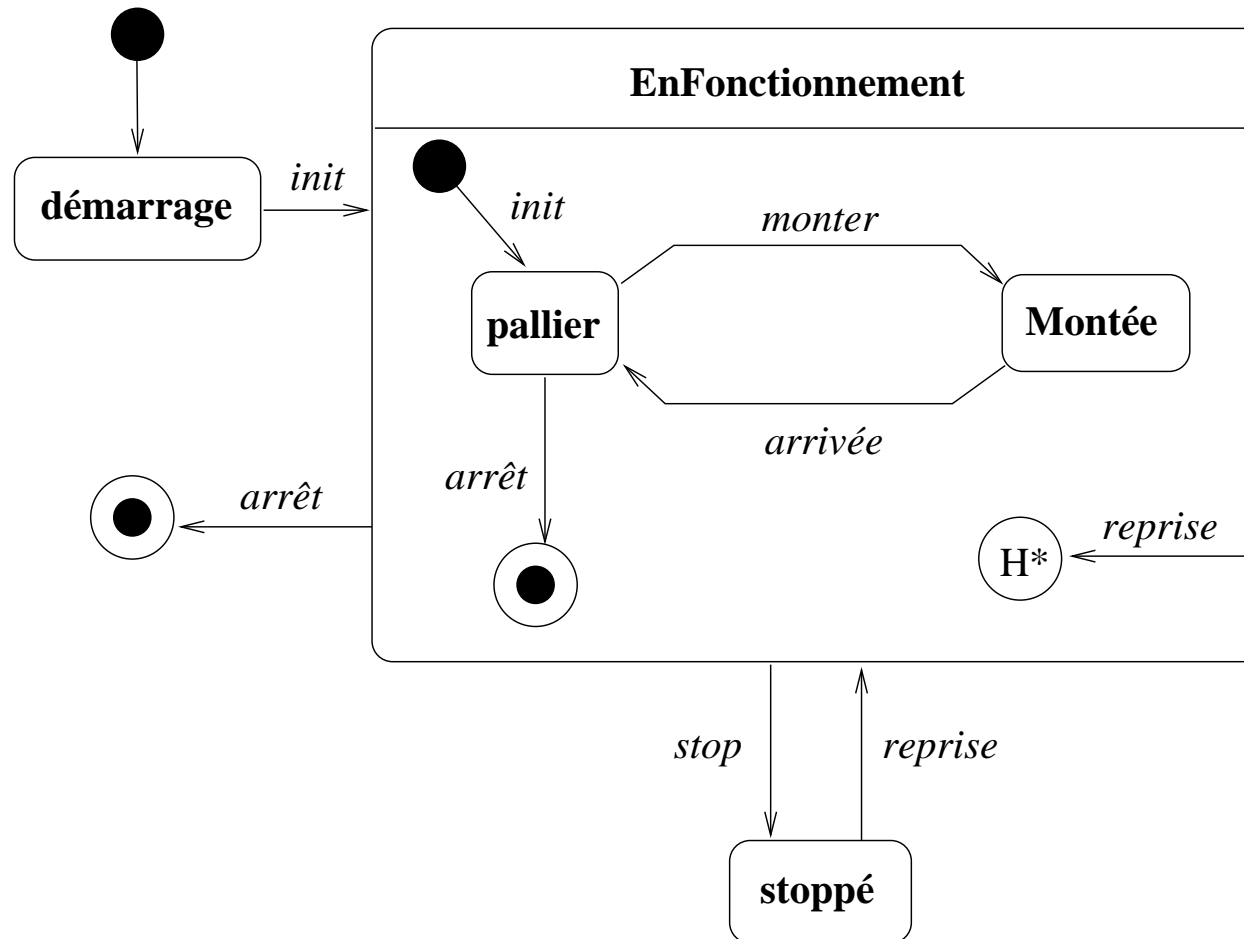


Figure 10 : Automate hiérarchique d'un ascenseur

# Extensions - hiérarchisation

## Hiérarchique concurrent

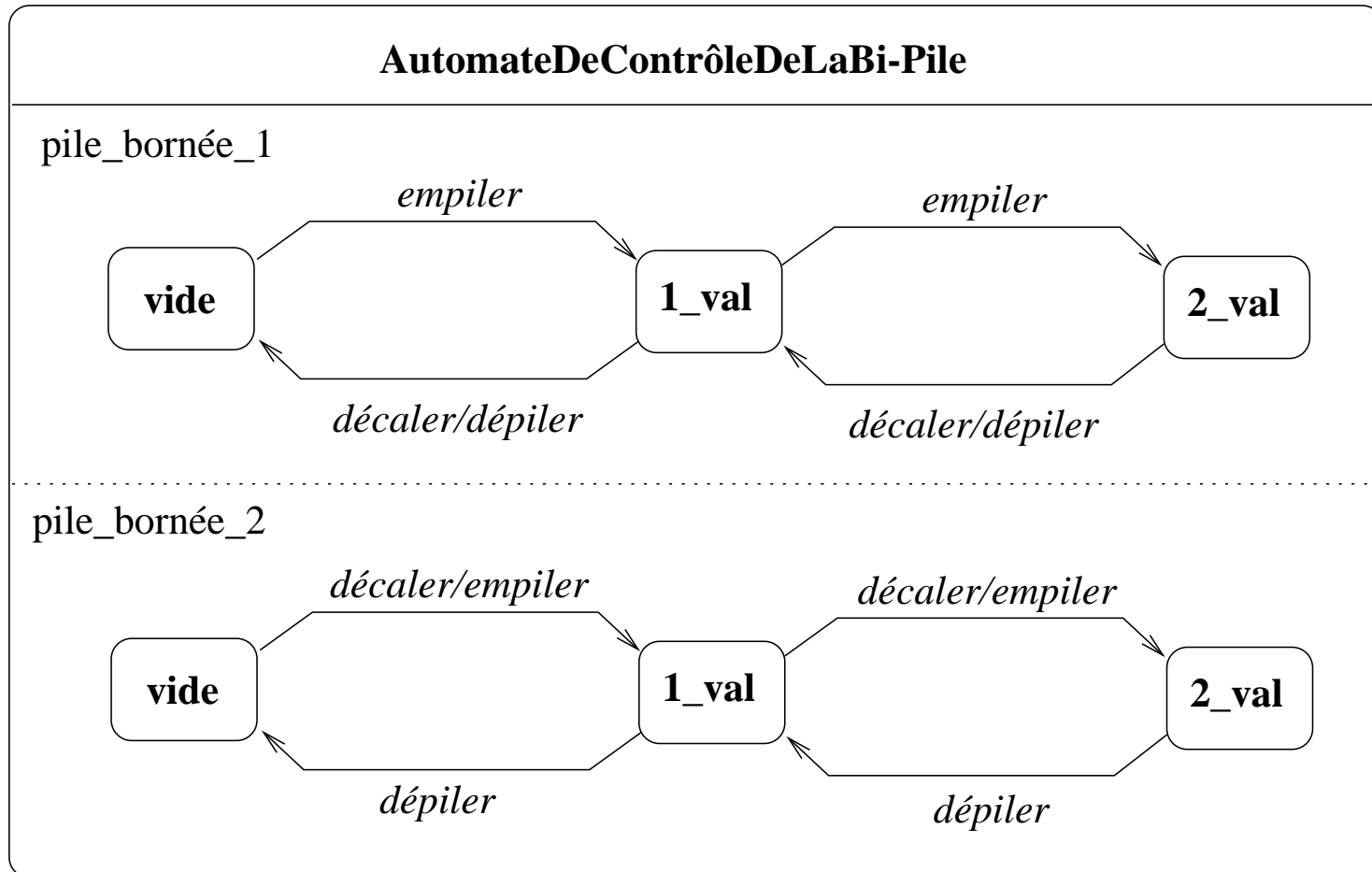


Figure 11 : Automate hiérarchique de la bi-pile

# Démarche de modélisation

1. Approche ascendante = **composition**
2. Approche descendante = **hiérarchisation**

# Bibliographie sommaire

- [AV01] Pascal André and Alain Vailly. *Conception de Systèmes d'Information, Panorama des méthodes et des techniques*, volume 1 of *Collection Technosup*. Editions Ellipses, 2001. ISBN 2-7298-0479-X.

# References

- [AV01] Pascal André and Alain Vailly. *Conception de Systèmes d'Information, Panorama des méthodes et des techniques*, volume 1 of *Collection Technosup*. Editions Ellipses, 2001. ISBN 2-7298-0479-X.