Génie Logiciel

Chapitre 5
Diagrammes UML : vue dynamique

Partie 2: Diagrammes d'états/transitions UML

Niveau: 3ième année Licence informatique

Année: 2025/2026

Diagrammes d’états/transitions UML

Introduction

- Un diagramme d'états-transitions décrit le comportement interne d'un objet à l'aide d'un automate à états finis.

- Un objet peut passer par une série d'états pendant sa durée de vie.

- Les objets changent d'état en réponse à des événements extérieurs donnant lieu à des transitions entre états.

- Sauf cas particuliers (lorsque il y a concurrence), à chaque instant, chaque objet est dans un et un seul état.

Etat et transition

- Les états sont représentés par des rectangles aux coins arrondis (ou en compartiments).

- Les transitions sont représentées par des arcs orientés liant les états entre eux.

- Certains états, dits « composites », peuvent contenir des sous-diagrammes.

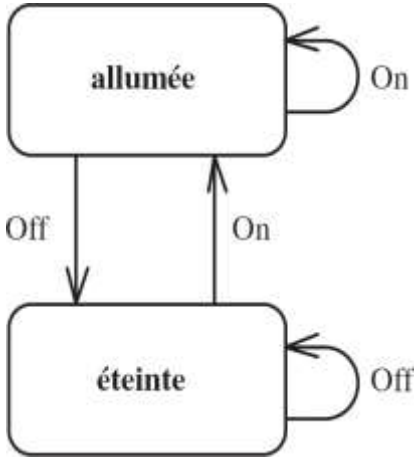


Diagramme d'états-transitions

- L'organisation des états et des transitions pour un classeur donné est représentée dans un diagramme d'états-transitions.

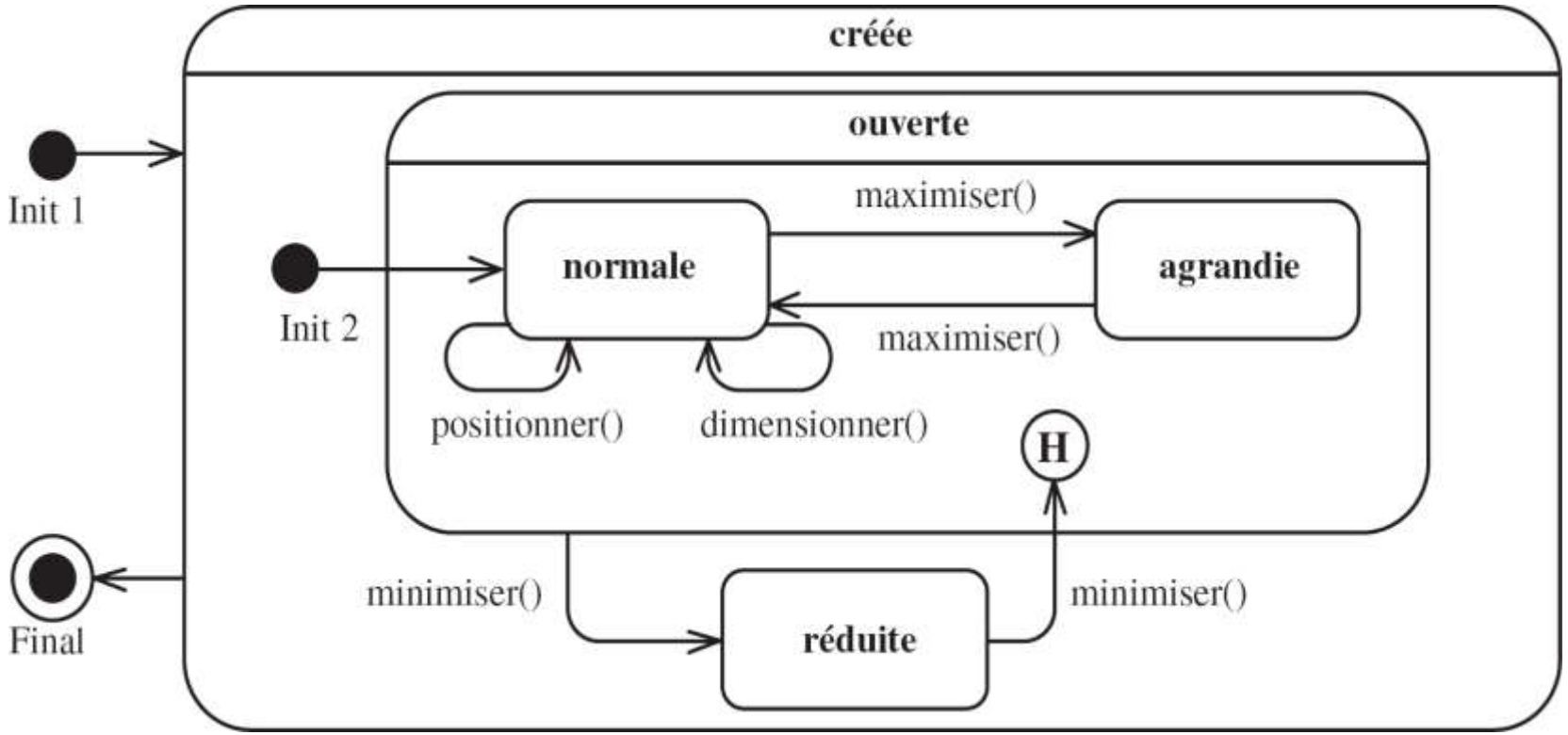
- Le modèle dynamique comprend plusieurs diagrammes d'états.

- Attention !!!

- Chaque diagramme d'états ne concerne qu'une seule classe.

- Chaque automate à états finis s'exécute concurremment et peut changer d'état de façon indépendante des autres.

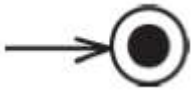
Exemple de diagramme d'états-transitions



Etat initial et état final

- L'état initial est un pseudo-état qui définit le point de départ par défaut pour l'automate ou le sous-état.
  - Lorsqu'un objet est créé, il entre dans l'état initial.

- L'état final est un pseudo-état qui indique que l'exécution de l'automate ou du sous-état est terminée.



Evénement déclencheur

- Un évènement est quelque chose qui se produit pendant l'exécution d'un système et qui mérite d'être modélisé.

- Un diagramme d'états-transitions spécifie les réactions à des évènements.

- Un évènement se produit à un instant précis et est dépourvu de durée.

- Quand un événement est reçu, une transition peut être déclenchée et faire basculer l'objet dans un nouvel état.

Types d’evènement

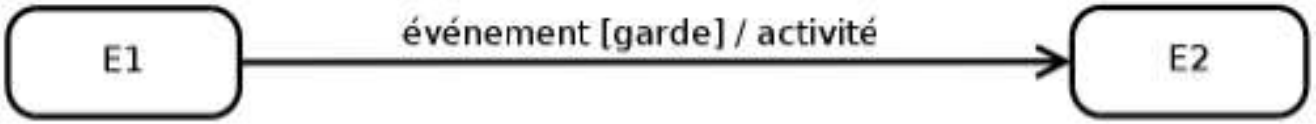
- Signal
  - ✓ Un message asynchrone reçu par l’objet.
  - ✓ Exemple : signalErreur envoyé par un capteur déclenche la transition vers l’état EnPanne.
- Appel
  - ✓ Déclenché lorsqu’une méthode est appelée sur l’objet.
  - ✓ Exemple : l’appel minimiser() fait passer l’objet de Ouverte à Réduite.
- Changement
  - ✓ Se produit lorsqu’une condition devient vraie.
  - ✓ Exemple : when (temperature > 40) déclenche une transition vers Surchauffe.
- Temporel
  - ✓ Déclenché par le temps (après un délai ou à un instant donné).
  - ✓ Exemple : after(5s) quitte l’état Inactif après 5 secondes.

Transition simple

- Une transition entre deux états est représentée par un arc qui les lie l'un à l'autre.
  - ✓ Elle indique qu'une instance peut changer d'état et exécuter certaines activités, si un événement déclencheur se produit et que les conditions de garde sont vérifiées.
- Sa syntaxe est la suivante :

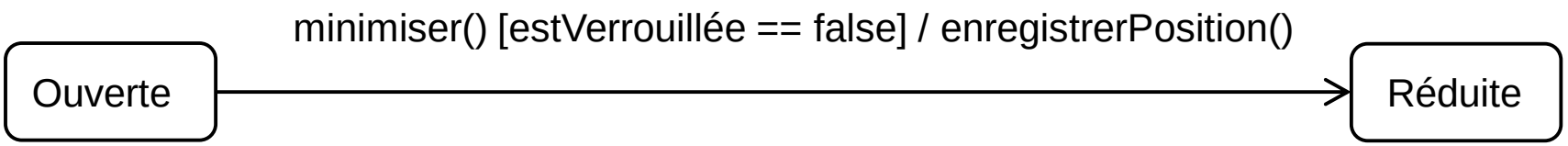
nomEvenement (params) [garde] / activité

- ✓ La garde désigne une condition qui doit être remplie pour pouvoir déclencher la transition,
- ✓ L'activité désigne des instructions à effectuer au moment du tir.



Transition simple (2)

- Exemple



- ✓ Événement : minimiser()
- ✓ Garde : la fenêtre ne doit pas être verrouillée
- ✓ Activité : sauvegarde de la position avant réduction

Transition interne

- Un objet reste dans un état durant une certaine durée et des transitions internes peuvent intervenir.

- Une transition interne ne modifie pas l'état courant, mais suit globalement les règles d'une transition simple entre deux états.

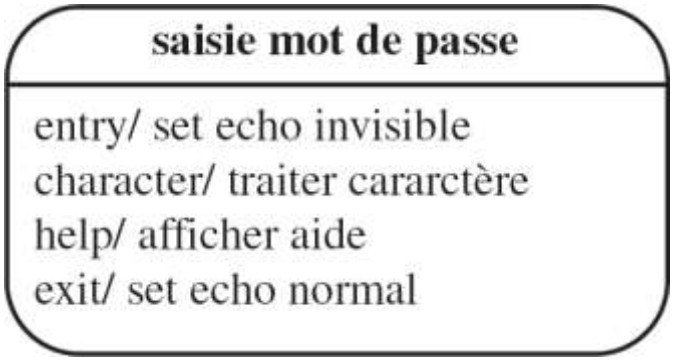
- Trois déclencheurs particuliers sont introduits permettant le tir de transitions internes : entry/, do/, et exit/.

Transition interne- Déclencheurs prédéfinis

- *entry*
  - ✓ définit une activité à effectuer à chaque fois que l'on rentre dans l'état considéré.

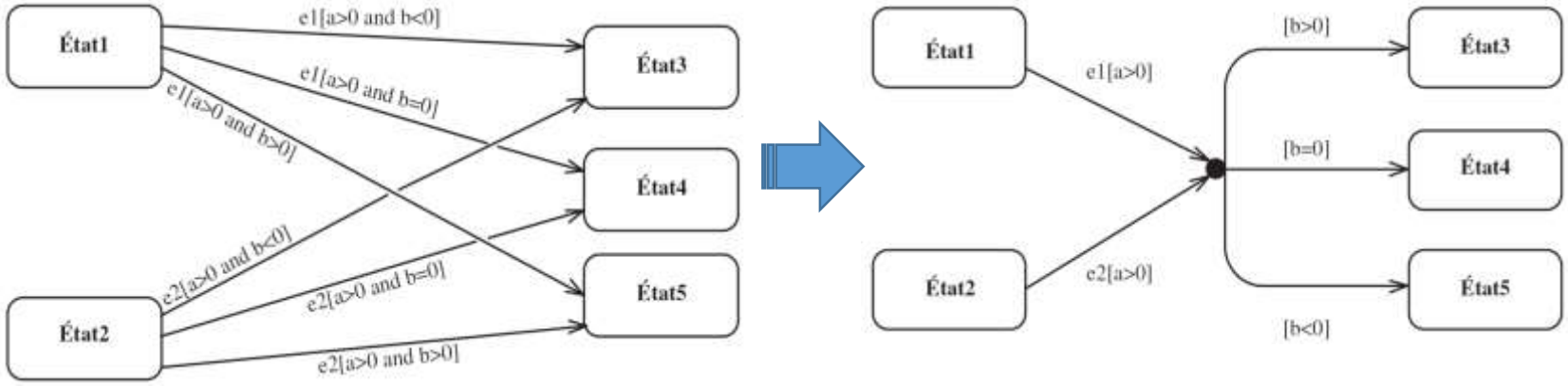
- *exit*
  - ✓ définit une activité à effectuer quand on quitte l'état.

- *do*
  - ✓ définit une activité continue qui est réalisée tant que l'on se trouve dans l'état, ou jusqu'à ce que le calcul associé soit terminé.



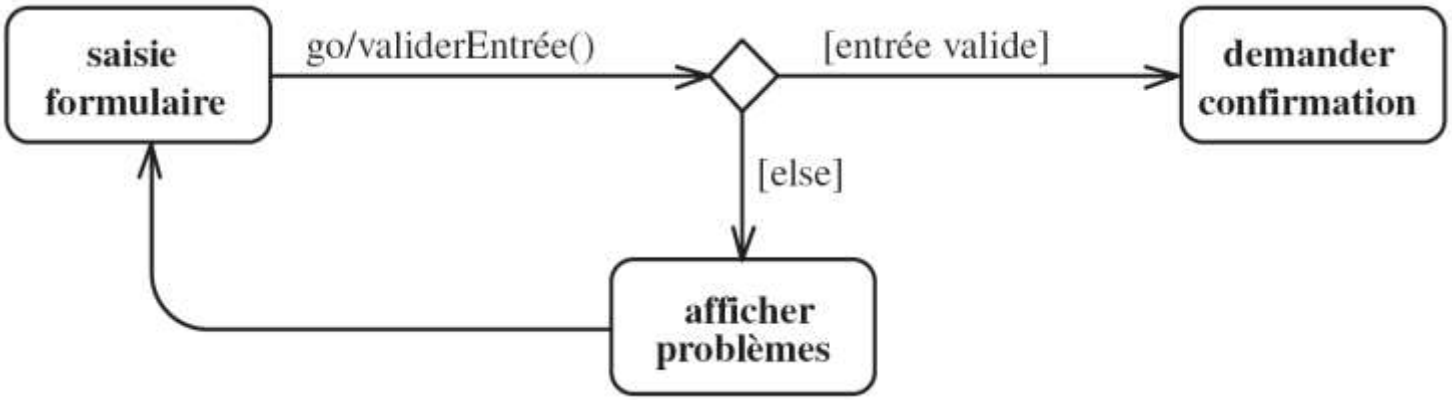
Point de jonction

- Permet de factoriser des segments de transition.
- Objectif : aboutir à une notation plus compacte ou plus lisible des chemins alternatifs.
- Toutes les gardes le long d'un chemin doivent s'évaluer à vrai dès le franchissement du premier segment.



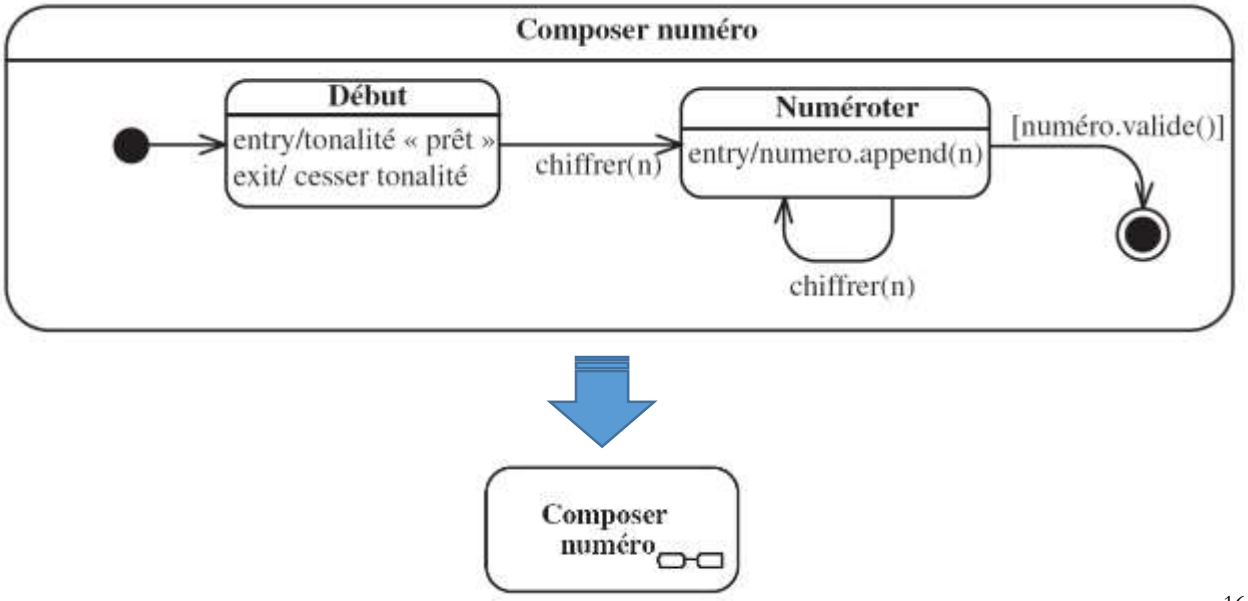
Point de décision

- Possède une entrée et au moins deux sorties.
- Les gardes situées après le point de décision sont évaluées au moment ou il est atteint.
- Une fois le point de décision atteint, au moins un chemin doit être franchissable.



Etat composite

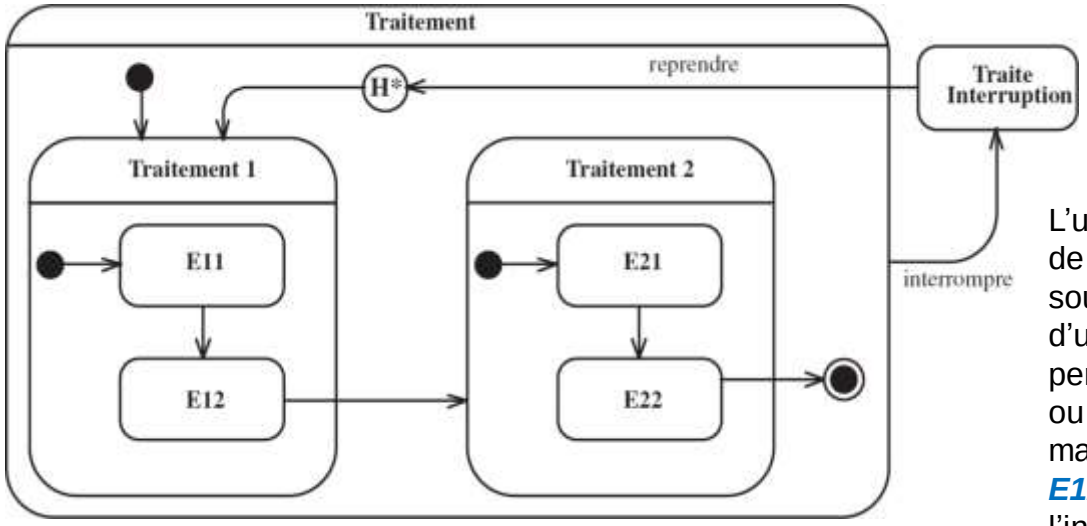
- Est un état décomposé en régions contenant chacune un ou plusieurs sous-états
- Plus d'une région : état orthogonal
  - ✓ Les régions sont concurrentes
- Une seule région : état non orthogonal
- Tout diagramme d'états-transitions est contenu dans un état composite enveloppant



Historique

- Etat historique plat
  - ✓ Un pseudo-état historique est noté par un H cerclé
  - ✓ Une transition ayant pour cible le pseudo-état historique est équivalente à une transition qui a pour cible le dernier état visité dans la région contenant le H

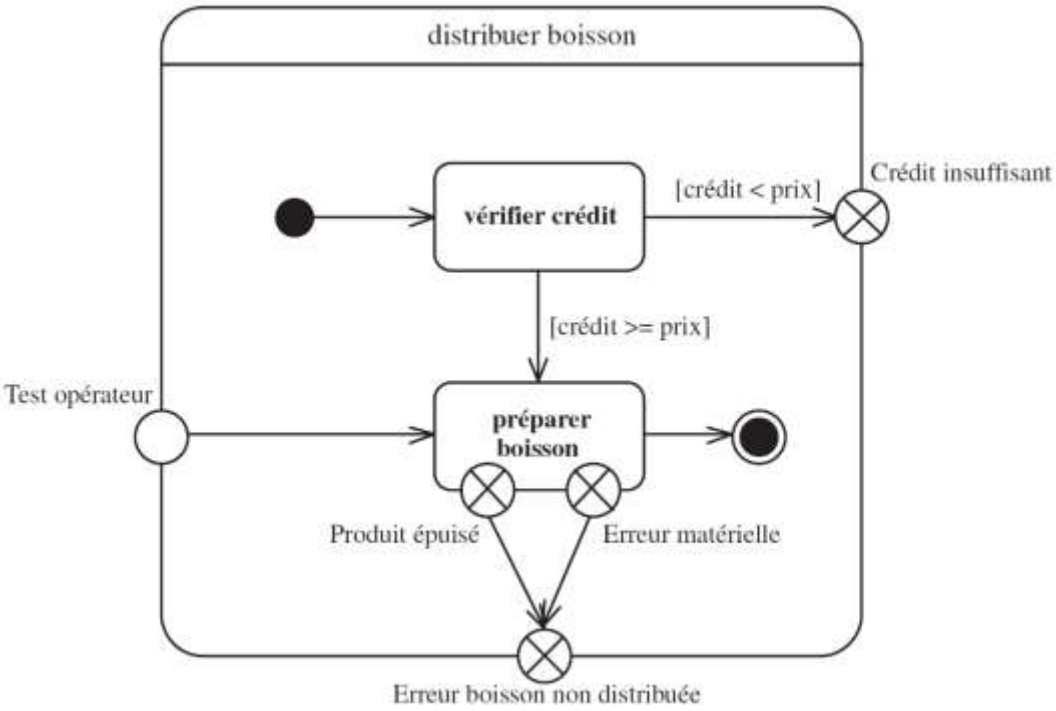
- Etat historique profond
  - ✓ H* désigne un historique profond, cad un historique valable pour tous les niveaux d'imbrication



L'utilisation d'un historique profond permet de retrouver, après une interruption, le sous-état précédent. À la figure, l'utilisation d'un historique de surface H, au lieu de H*, permettrait de retrouver l'état ***Traitement1*** ou ***Traitement2*** dans leur **sous-état initial**, mais pas les sous-états imbriqués ***E11***, ***E12***, ***E21***, ***E22***, qui étaient occupés avant l'interruption.

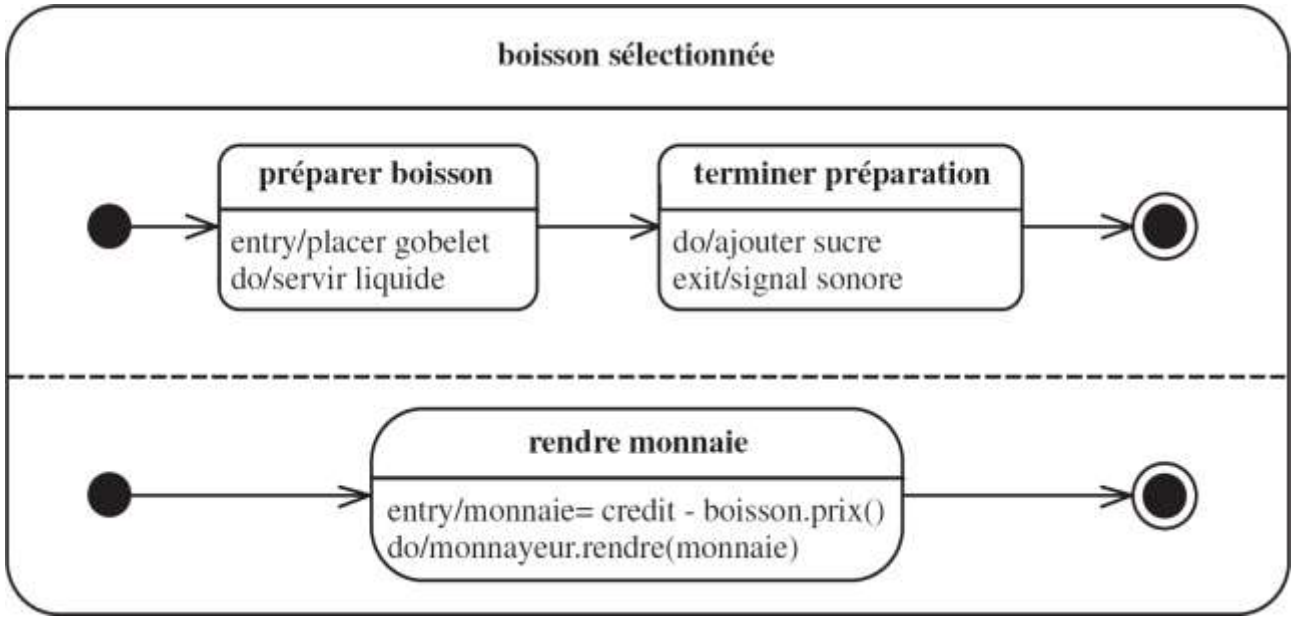
Interface des états composites

- Pour pouvoir représenter un sous état indépendamment d'un macro-état, on a recours à des points de connexion.
  - ✓ Avec un X pour les points de sortie
  - ✓ Vides pour les points d'entrée
- Ces interfaces permettent d'abstraire les sous-états des macro-états (réutilisabilité)



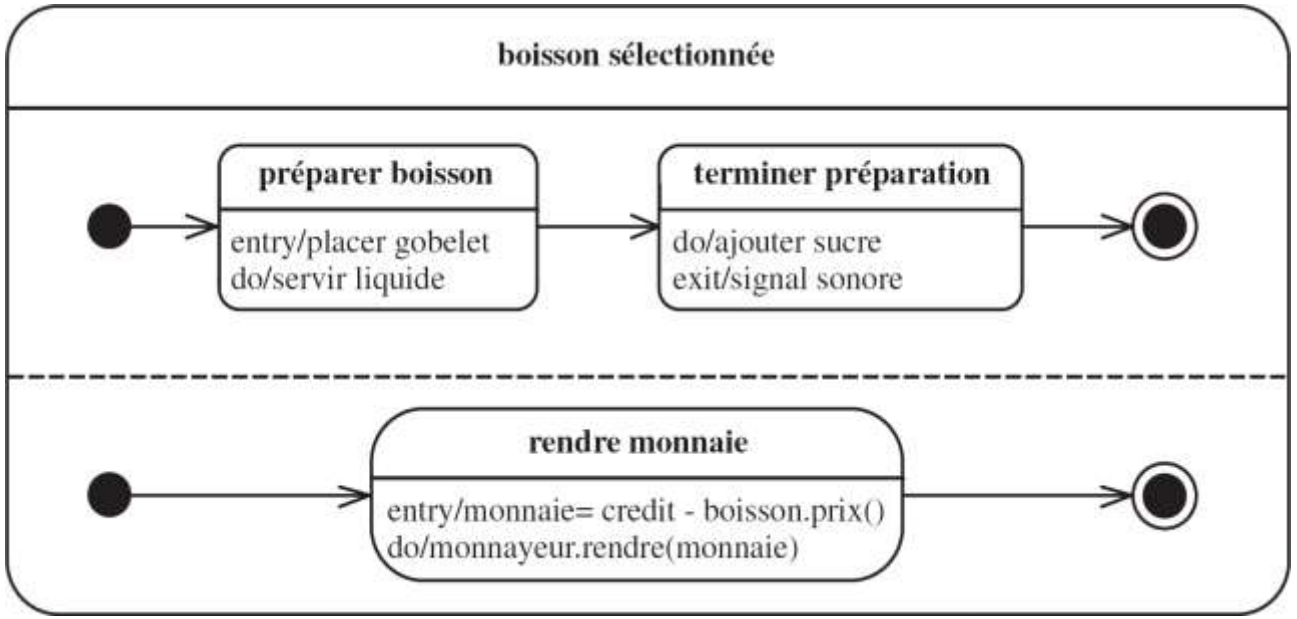
Etat concurrent

- Avec un séparateur en pointillés
  - ✓ On peut représenter plusieurs automates s'exécutant indépendamment
  - ✓ On parle des régions
- Un objet peut alors être simultanément dans plusieurs états concurrents



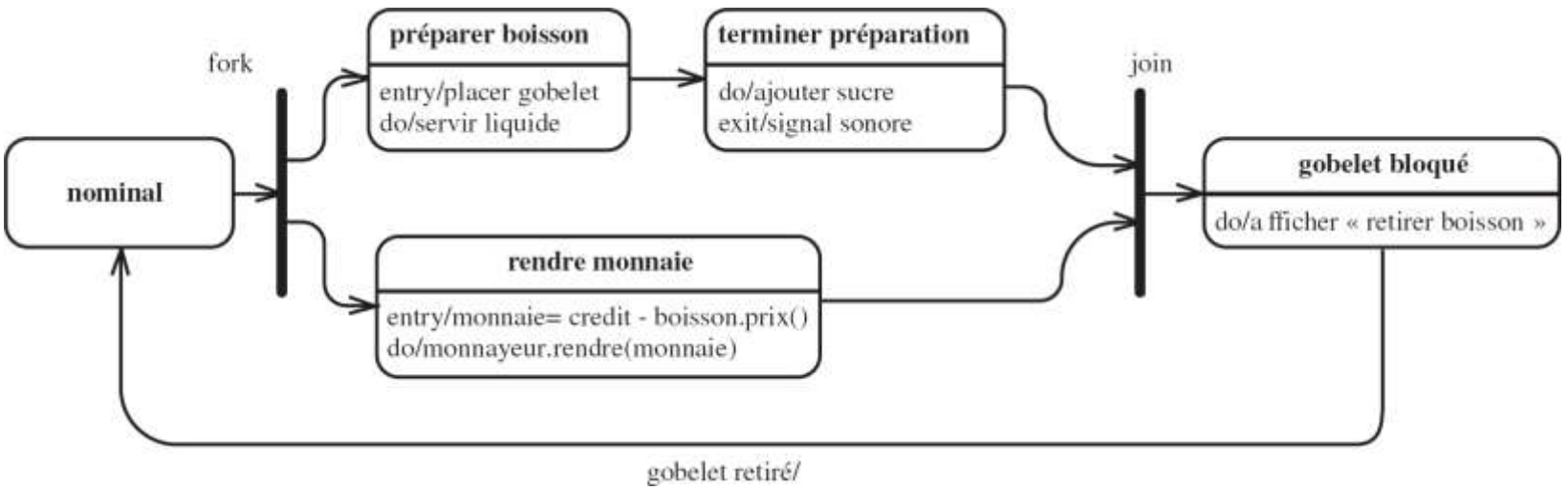
Etat concurrent

- Avec un séparateur en pointillés
  - ✓ On peut représenter plusieurs automates s'exécutant indépendamment
  - ✓ On parle des régions
- Un objet peut alors être simultanément dans plusieurs états concurrents



Transition concurrente

- Une transition Fork correspond à la création des états concurrents
- Une transition Join correspond à une barrière de synchronisation qui supprime la concurrence
  - ✓ L'exécution ne peut continuer qu'une fois que toutes les tâches concurrentes ont atteint le Join.



Questions