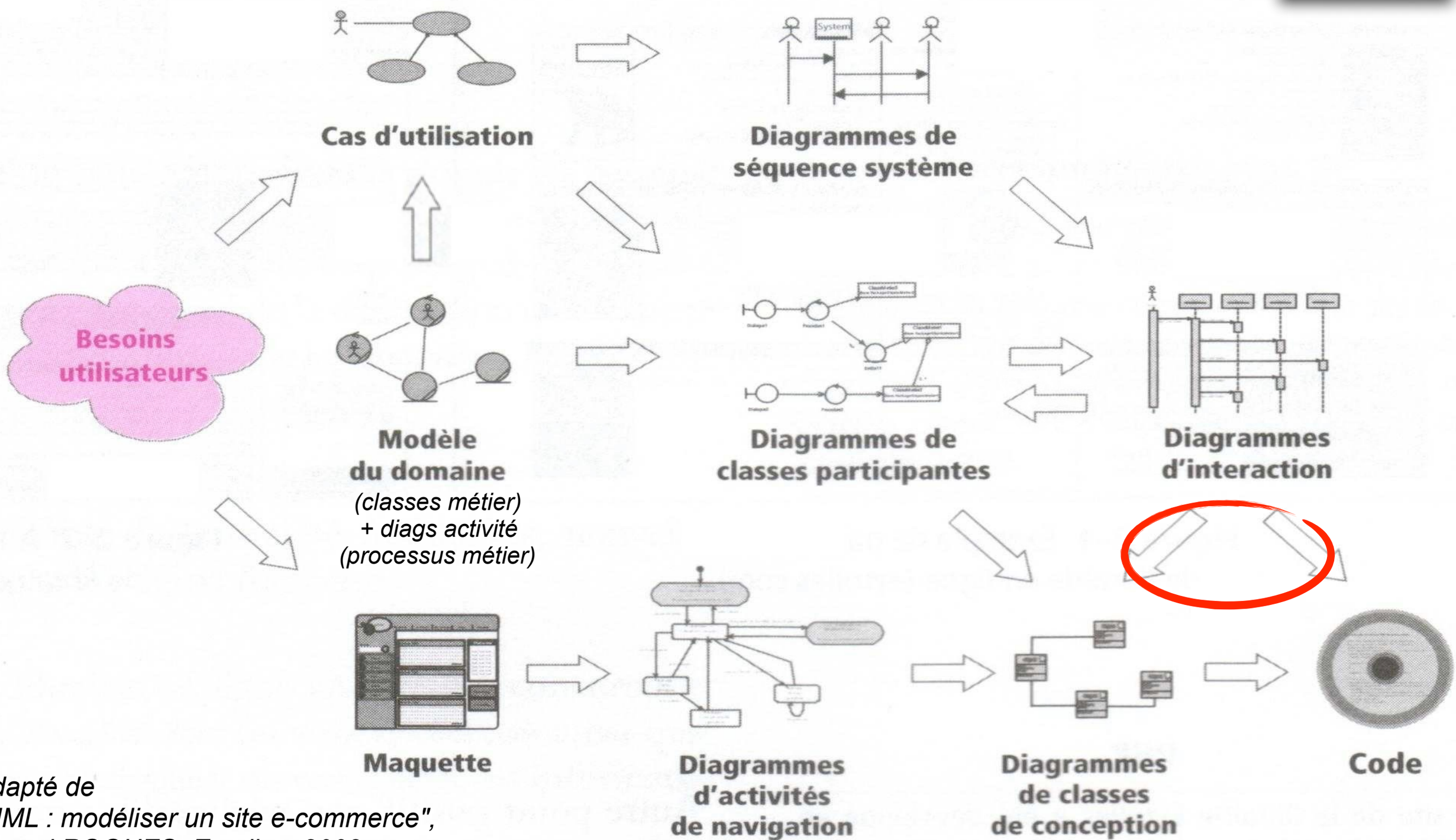
MCO
Diagrammes d'états

Ileana Ober
Université Paul Sabatier
IRIT
http://www.irit.fr/~Ileana.Ober/

Année Universitaire 2019-2020

Une démarche



Adapté de
"UML : modéliser un site e-commerce",
Pascal ROQUES, Eyrolles, 2002

Plan

- Principes de la communication en UML
- Description du comportement en UML
- Notions de base dans les machines à états
- Utilisation des machines à états

Plan

- **Principes de la communication en UML**
- Description du comportement en UML
- Notions de base dans les machines à états
- Utilisation des machines à états

Primitives de communication en UML

- **Signal**

Uni directionnel
Primitive de communication asynchrone
Peut transporter des données
Doit être explicitement définit

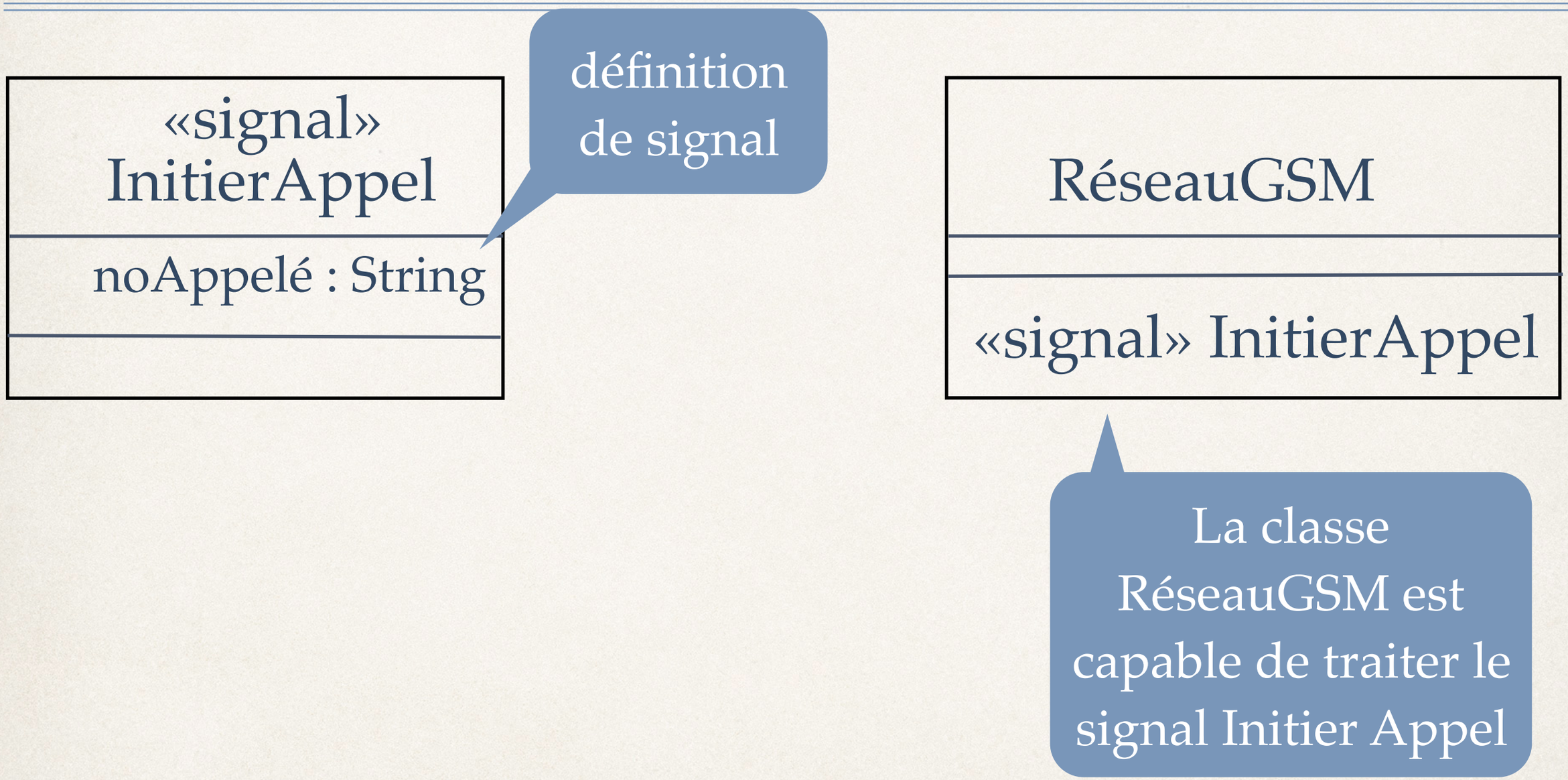
- **Appel d'opération**

Bi-directionnel communication primitive (appel - retour)
L'appelant est bloqué pendant l'appel
Peut transporter des données
En général il est fait sur un objet bien précis

- **File d'attente**

Buffer de communication
Est associé à des instances (objets) ou groupes d'instances
La politique de gestion de la file d'attente est en général du FIFO, mais d'autres politiques peuvent être spécifiées

Exemples - signaux



Mécanismes de communication en UML

- Si dans le modèle *il n'y a pas d'information explicite* liée à la communication (i.e. pas de diagramme d'architecture avec des connecteurs)
  Point à point - entre des objets se connaissent
  (des liens existent, retours d'opération, etc.)
  Diffusion - le message arrive à tous les objets accessibles et disponibles

- Si une *structure de communication existe* - la communication respecte ses contraintes (e.g. spécifiée par un diagramme d'architecture)

Plan

- Principes de la communication en UML
- **Description du comportement en UML**
- Notions de base dans les machines à états
- Utilisation des machines à états

Spécification du comportement en UML

	préscriptif (qu'est-ce que ?)	descriptif (comment?)
niveau système	OCL, DI	MàE, DA
niveau paquetage (ensemble classes)	DI, contraintes OCL	MàE, DA
niveau classe	contraintes OCL	MàE
niveau opération	contraintes OCL	actions, MàE

Machine à états

- Qu'est-ce que c'est?
  - **automate à états finis, qui détaille (décrit)** le comportement
- Contexte
  - système / pacquage - décrit le comportement d'un ensemble d'objets
    - une diagramme d'activités plus complexe et plus détaillé
    - s'utilise surtout dans le cas des systèmes /pacquages statiques (faiblement dynamiques), i.e. peu de création destruction d'objets, peu d'instances /classe
  - classe - décrit le comportement de chaque instance de la classe
    - décrit le comportement détaille en termes de réactions aux événements
    - peut définir la politique de gestion des appels concurrents
  - opération - décrit le comportement de l'objet englobant en réponse aux appels de l'opération
    - décrit le comportement détaille en termes de réactions aux événements

Plan

- Principes de la communication en UML
- Description du comportement en UML
- **Notions de base dans les machines à états**
- Utilisation des machines à états

Notions de base

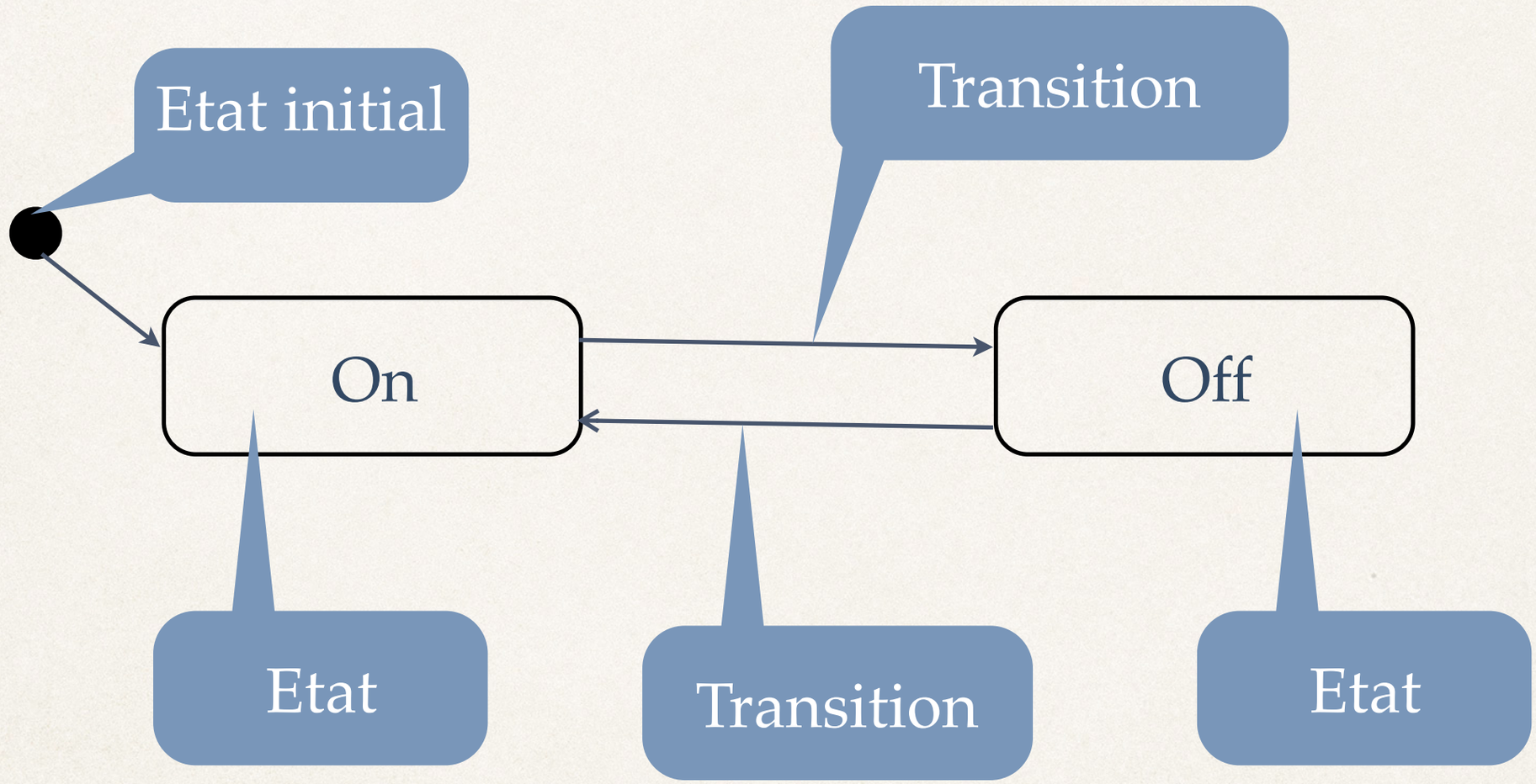
- Etat

Connecté

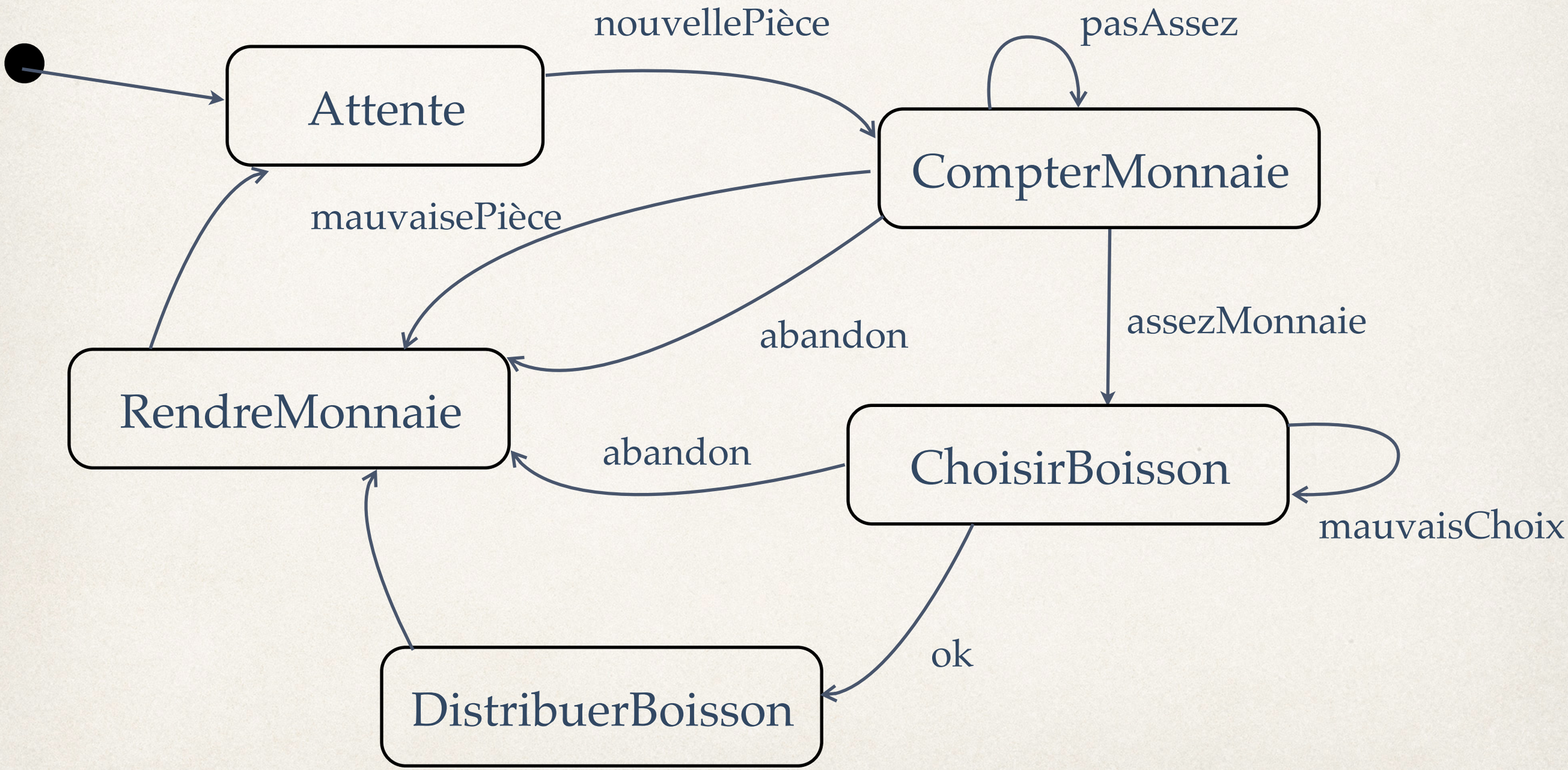
- Transition

- Action *send* terminé *to* caller
- Evénement fermer, timeout, ...
- File d'attente (sans notation explicite)

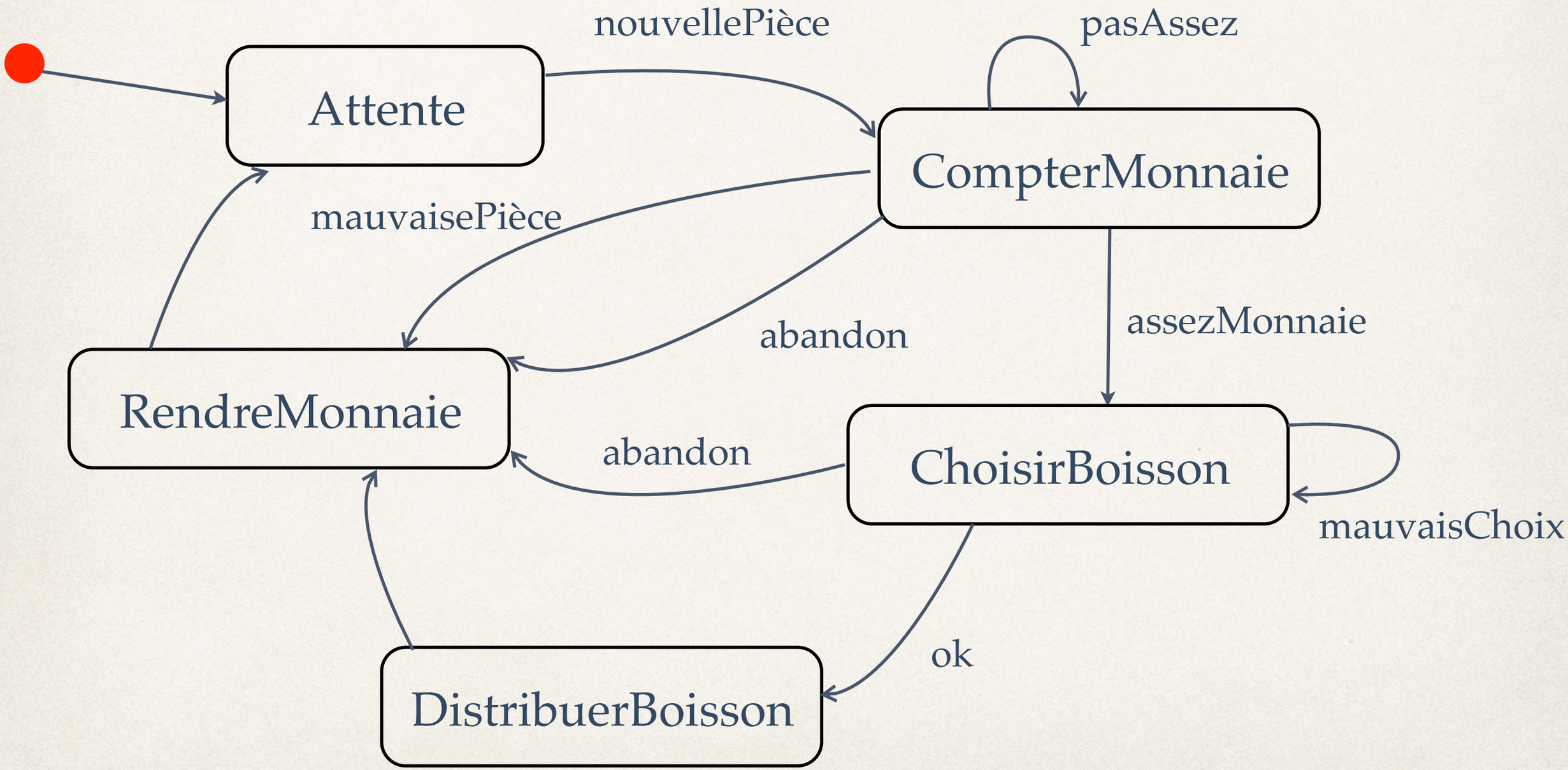
Exemple simple



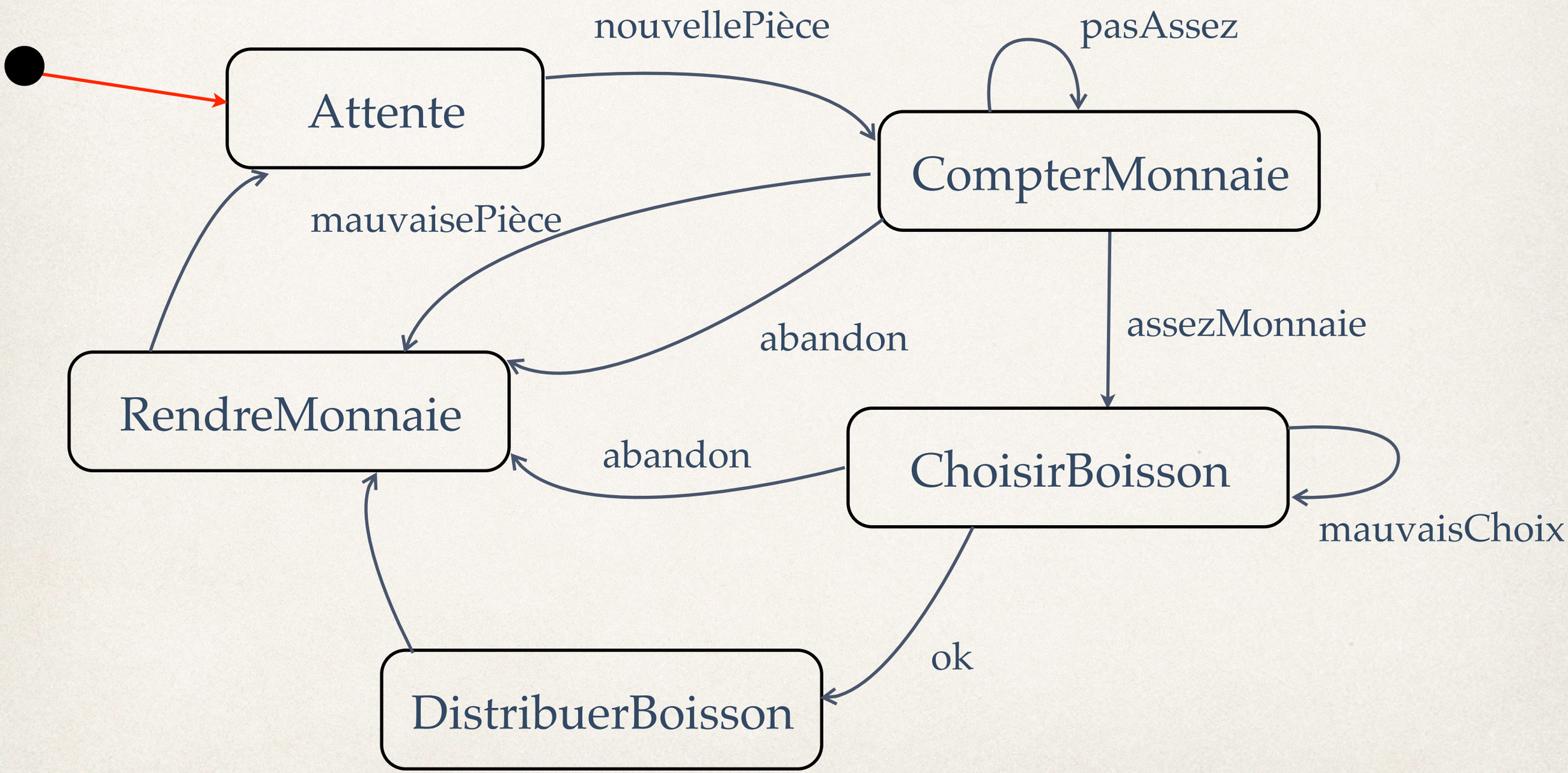
Exemple distributeur boissons



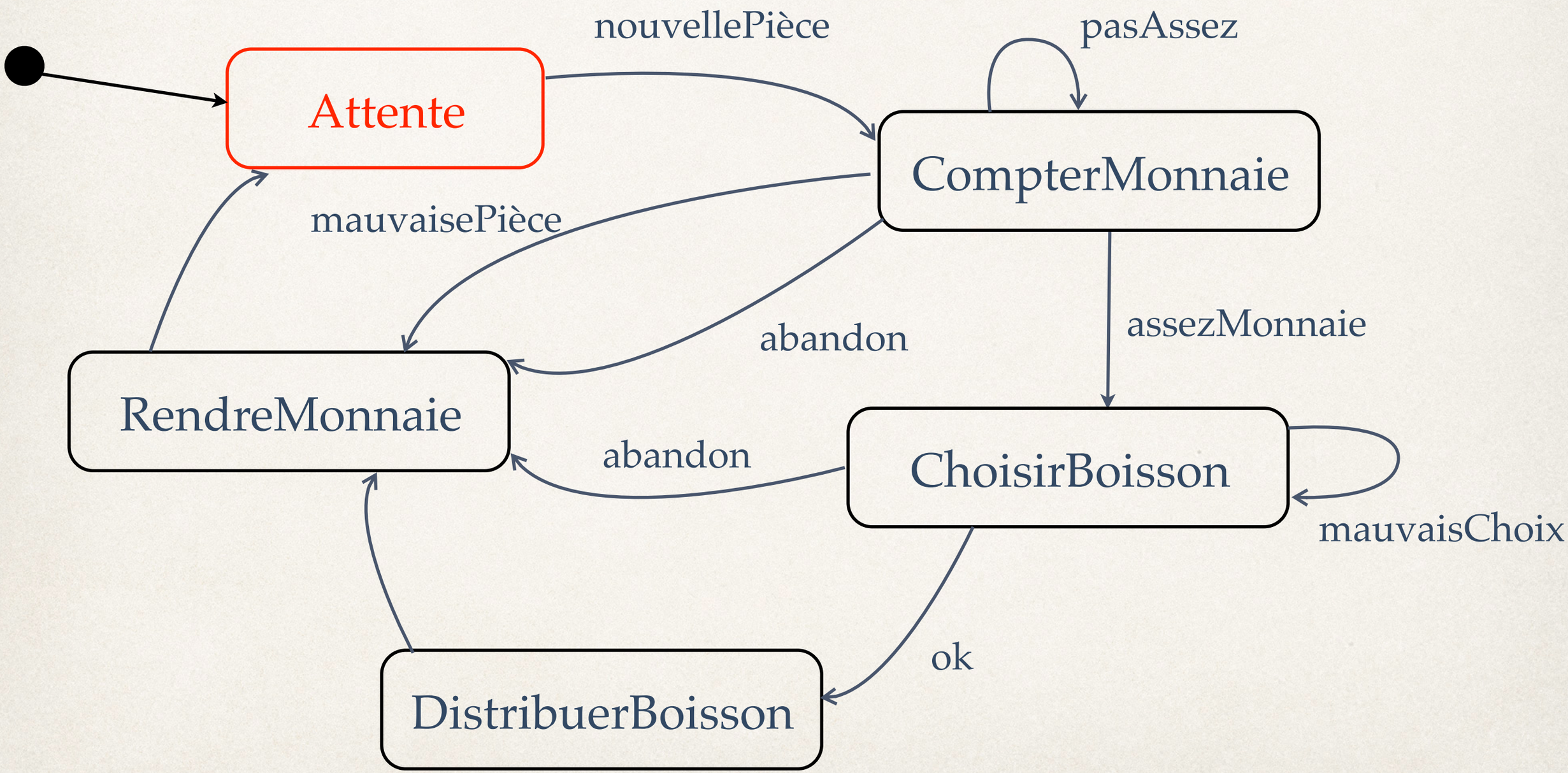
Exemple exécution



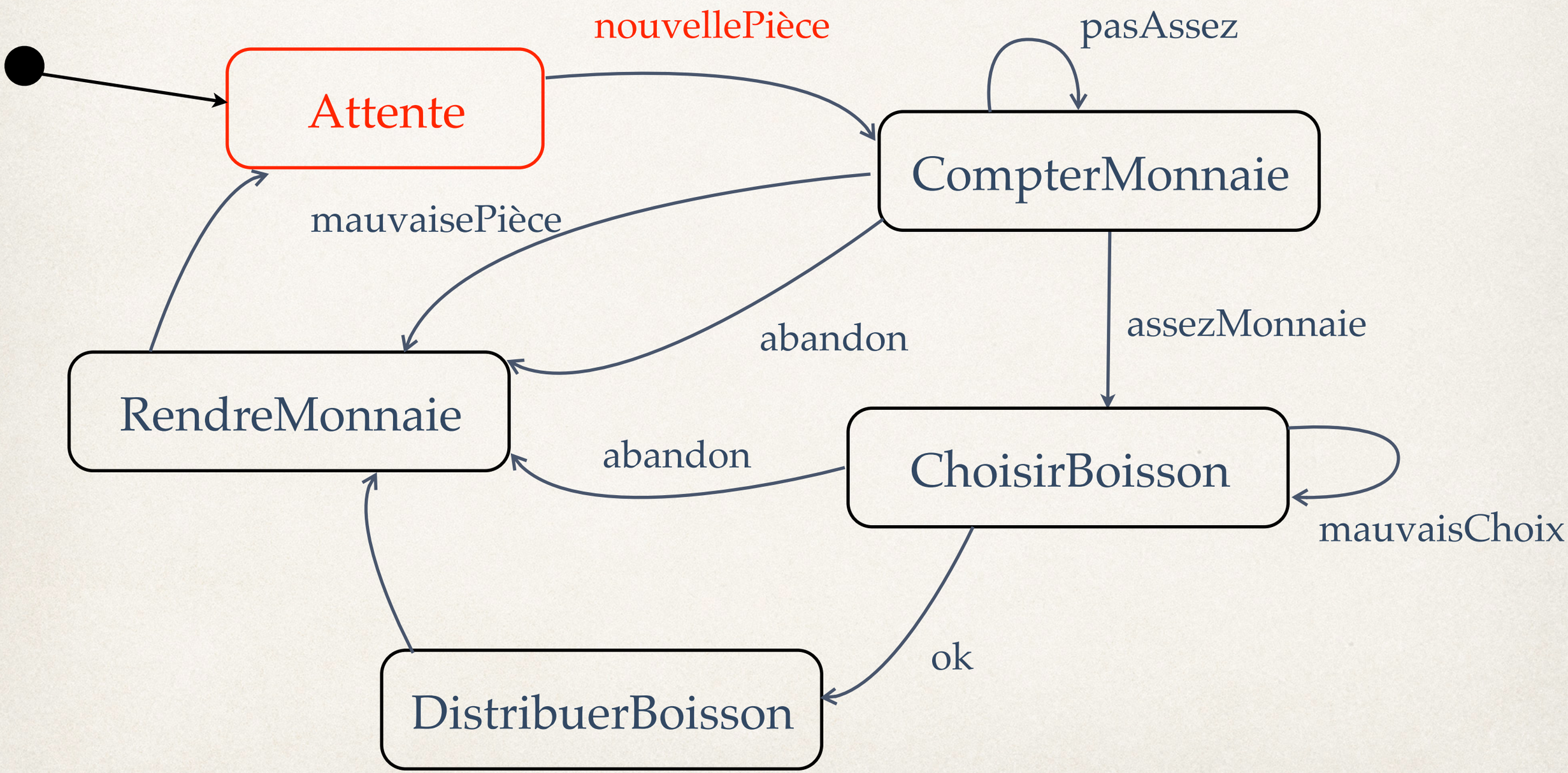
Exemple exécution



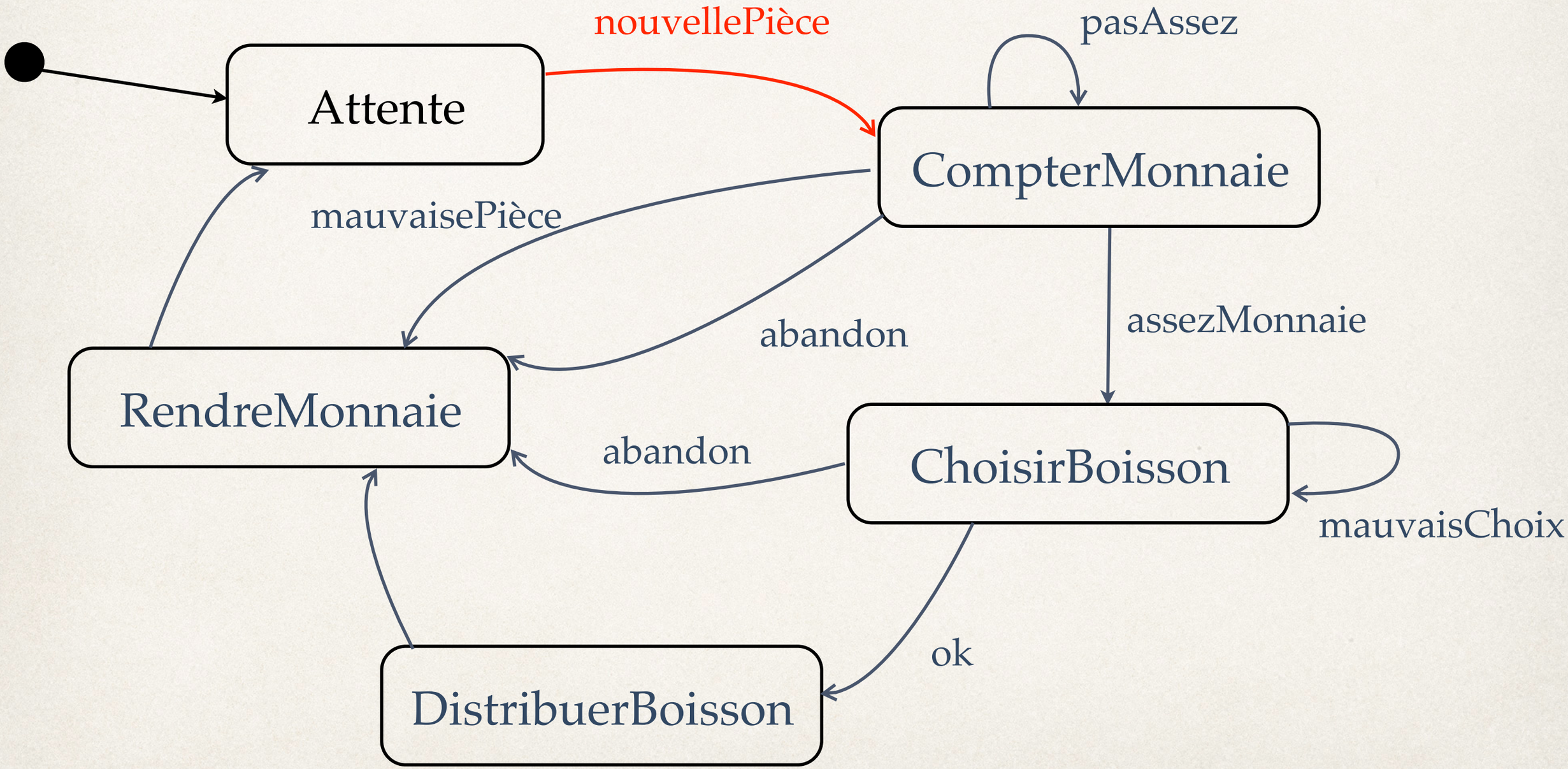
Exemple exécution



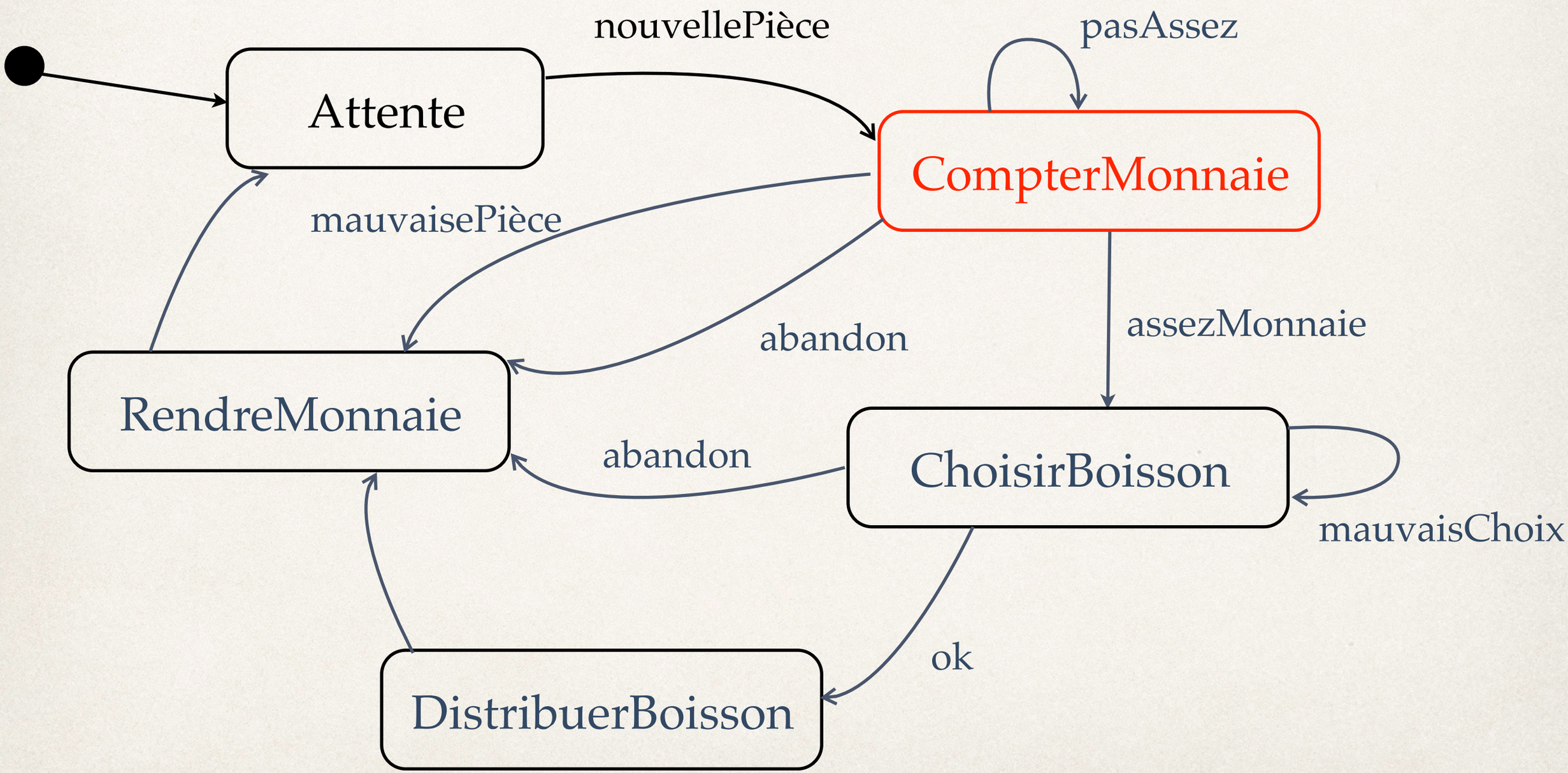
Exemple exécution



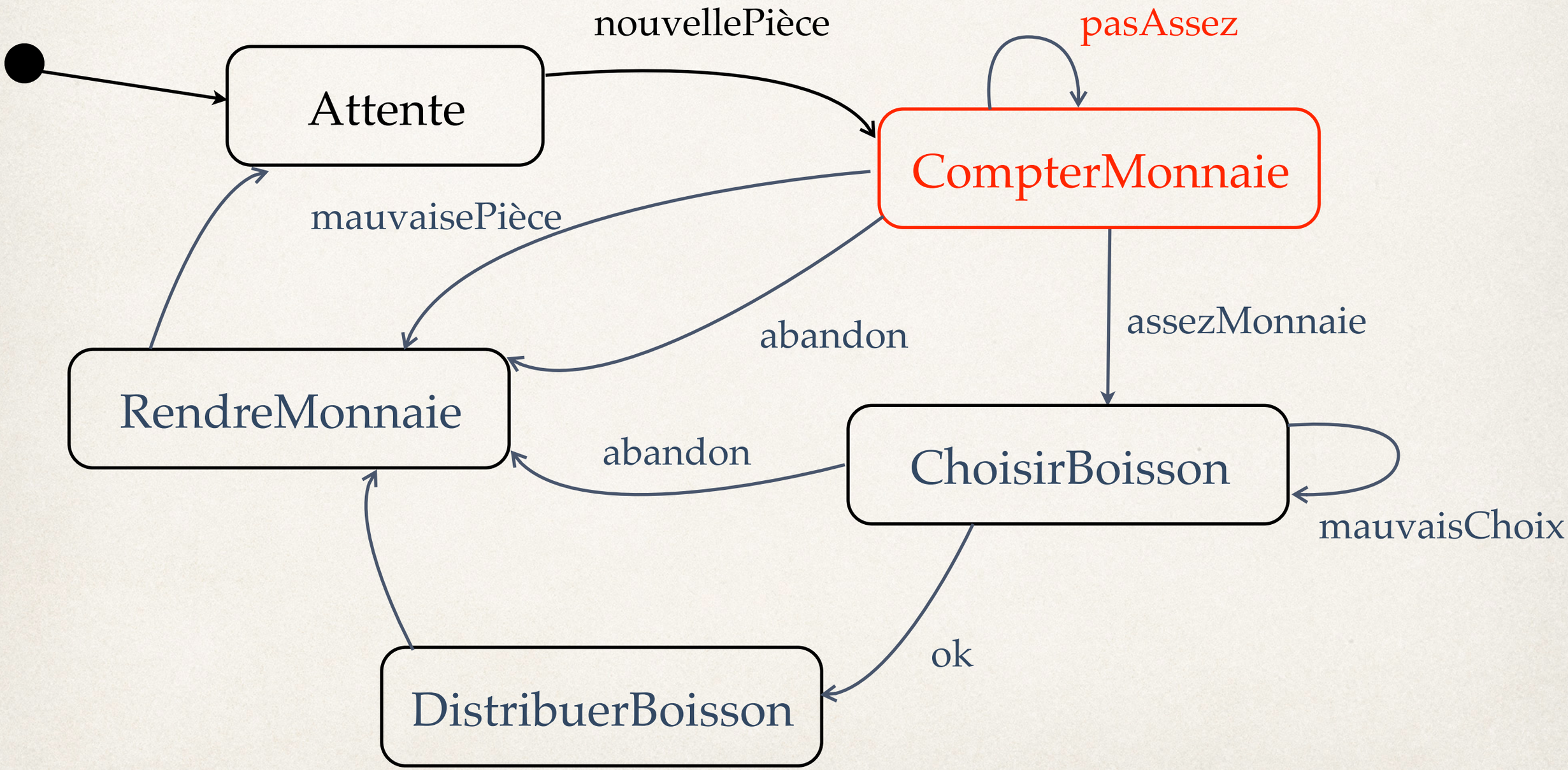
Exemple exécution



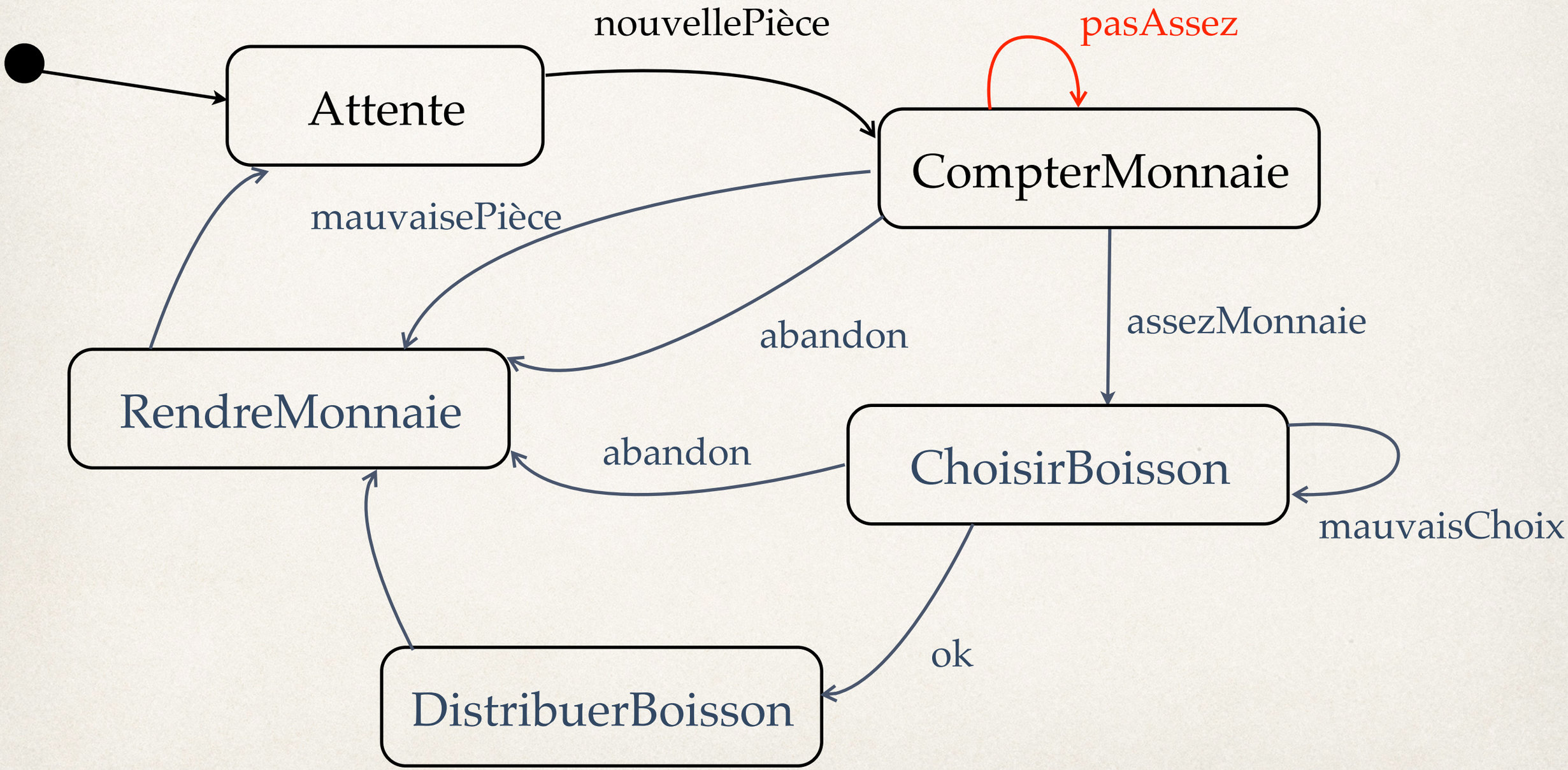
Exemple exécution



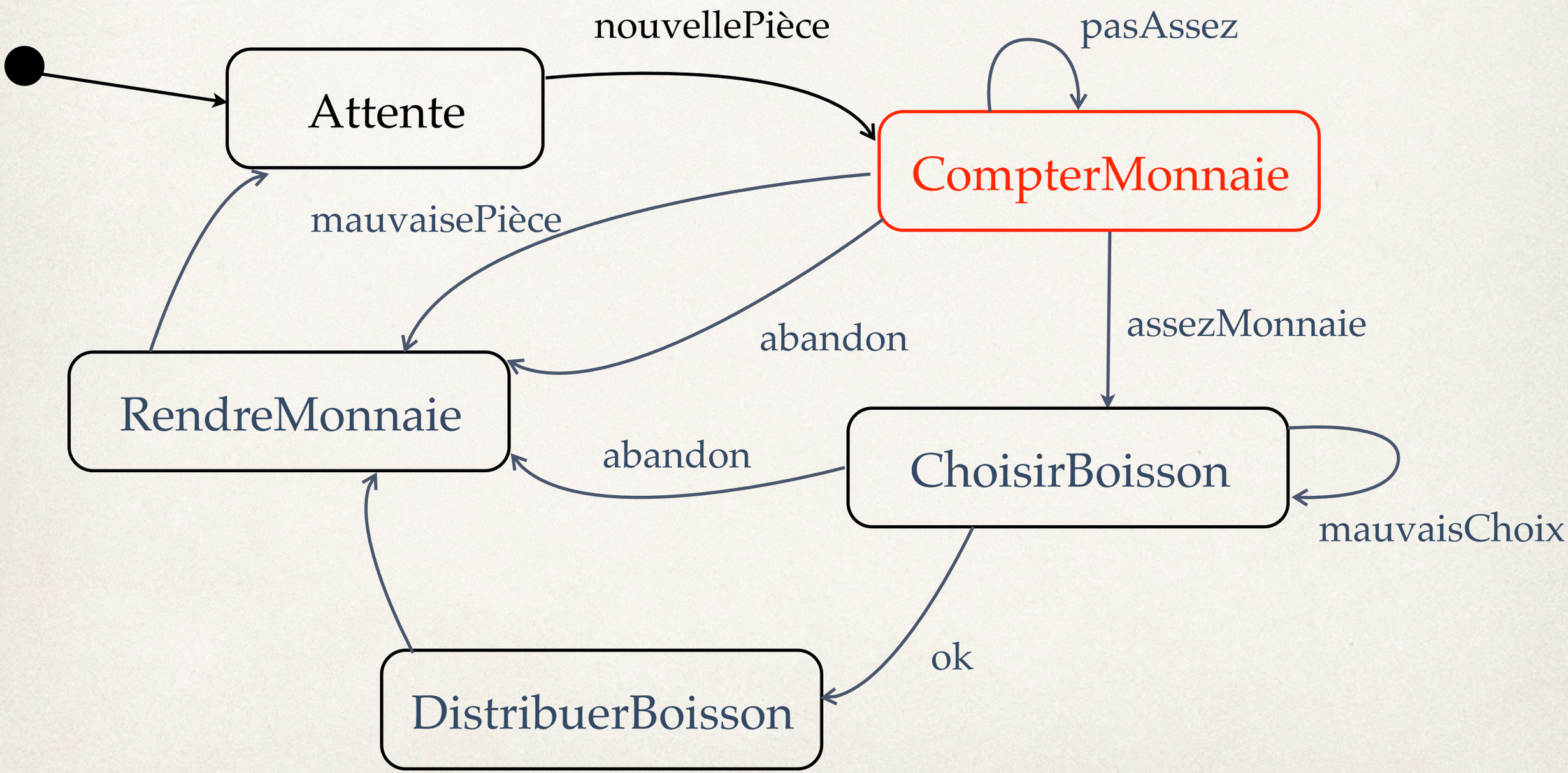
Exemple exécution



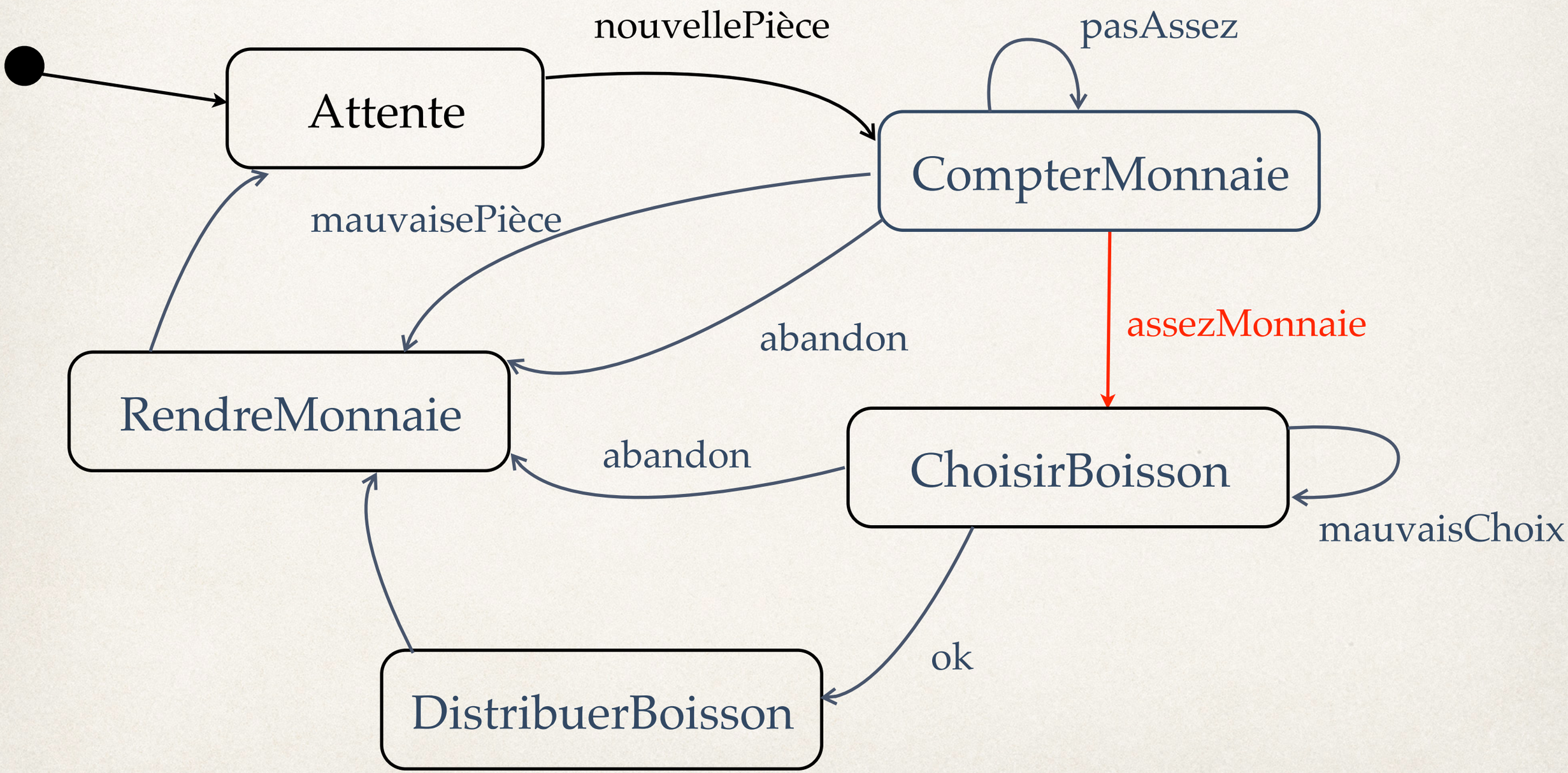
Exemple exécution



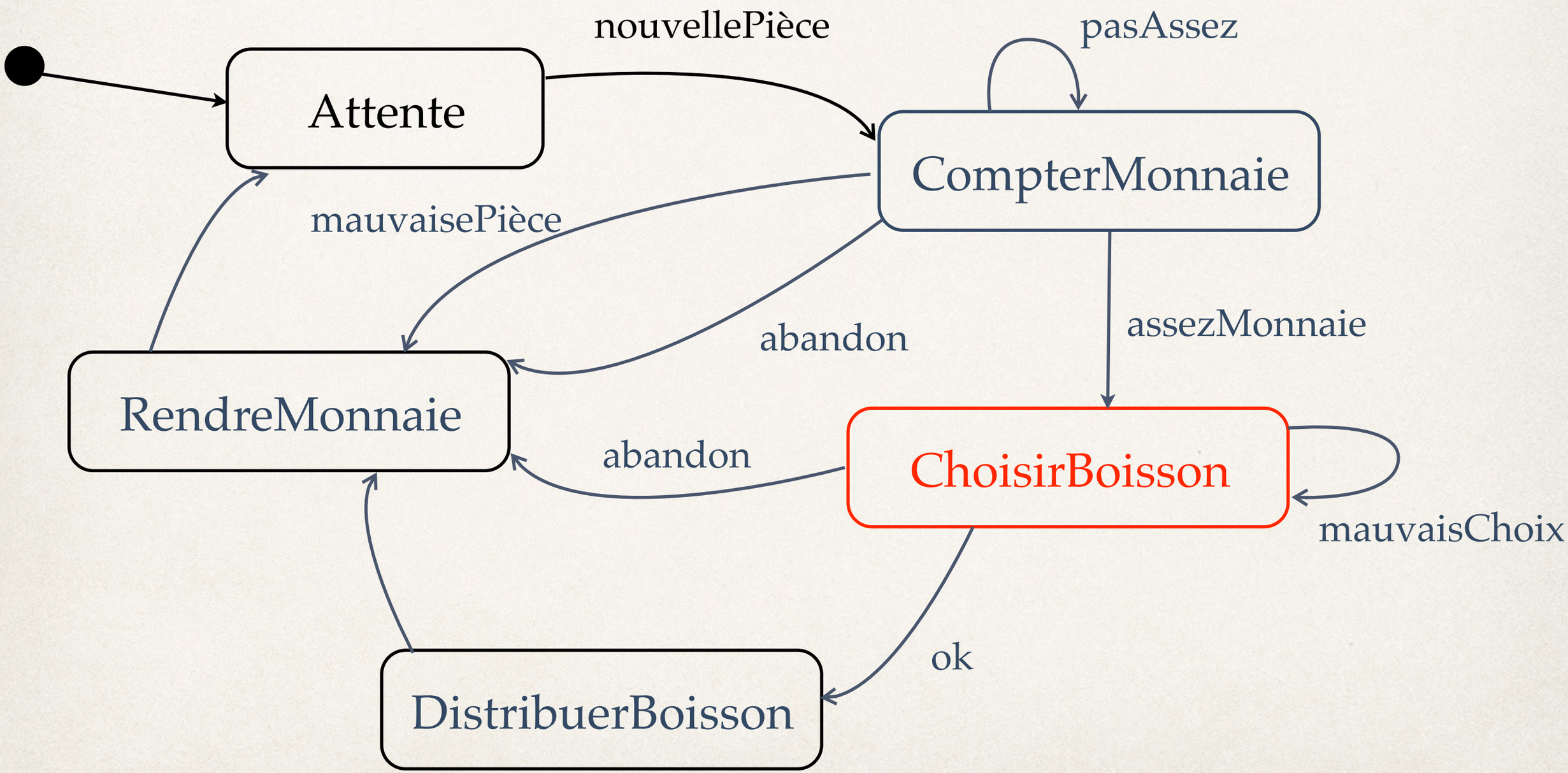
Exemple exécution



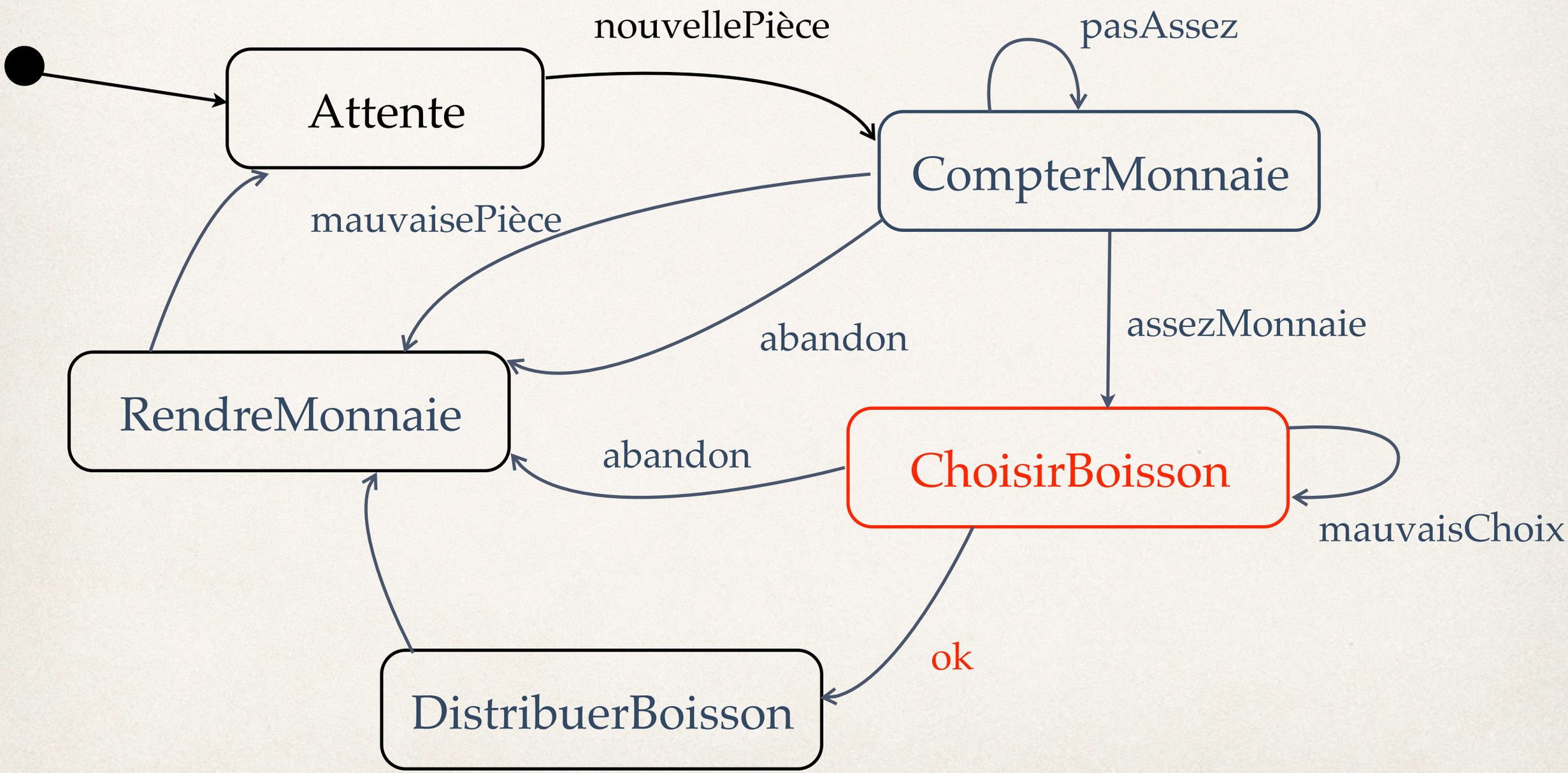
Exemple exécution



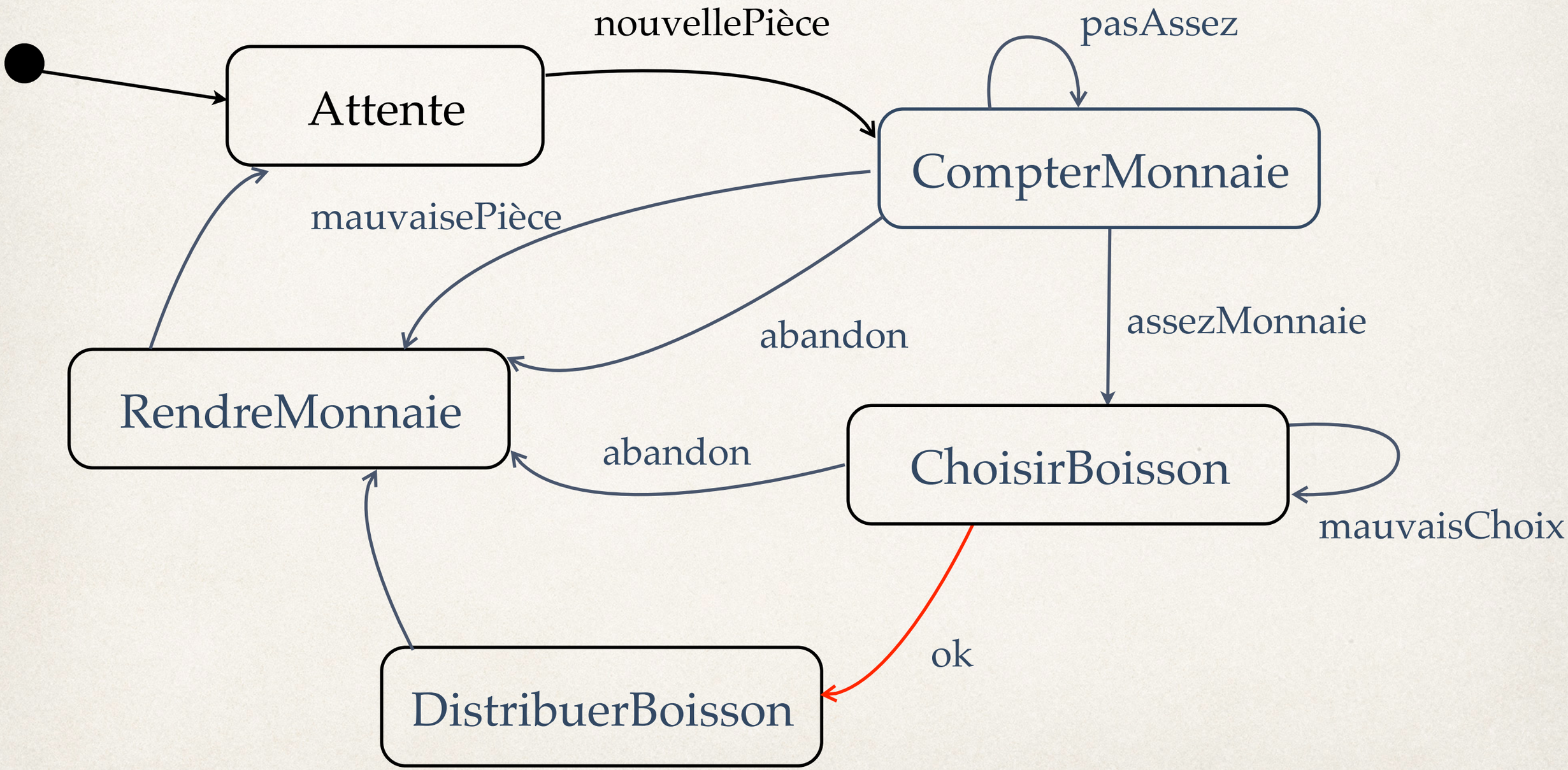
Exemple exécution



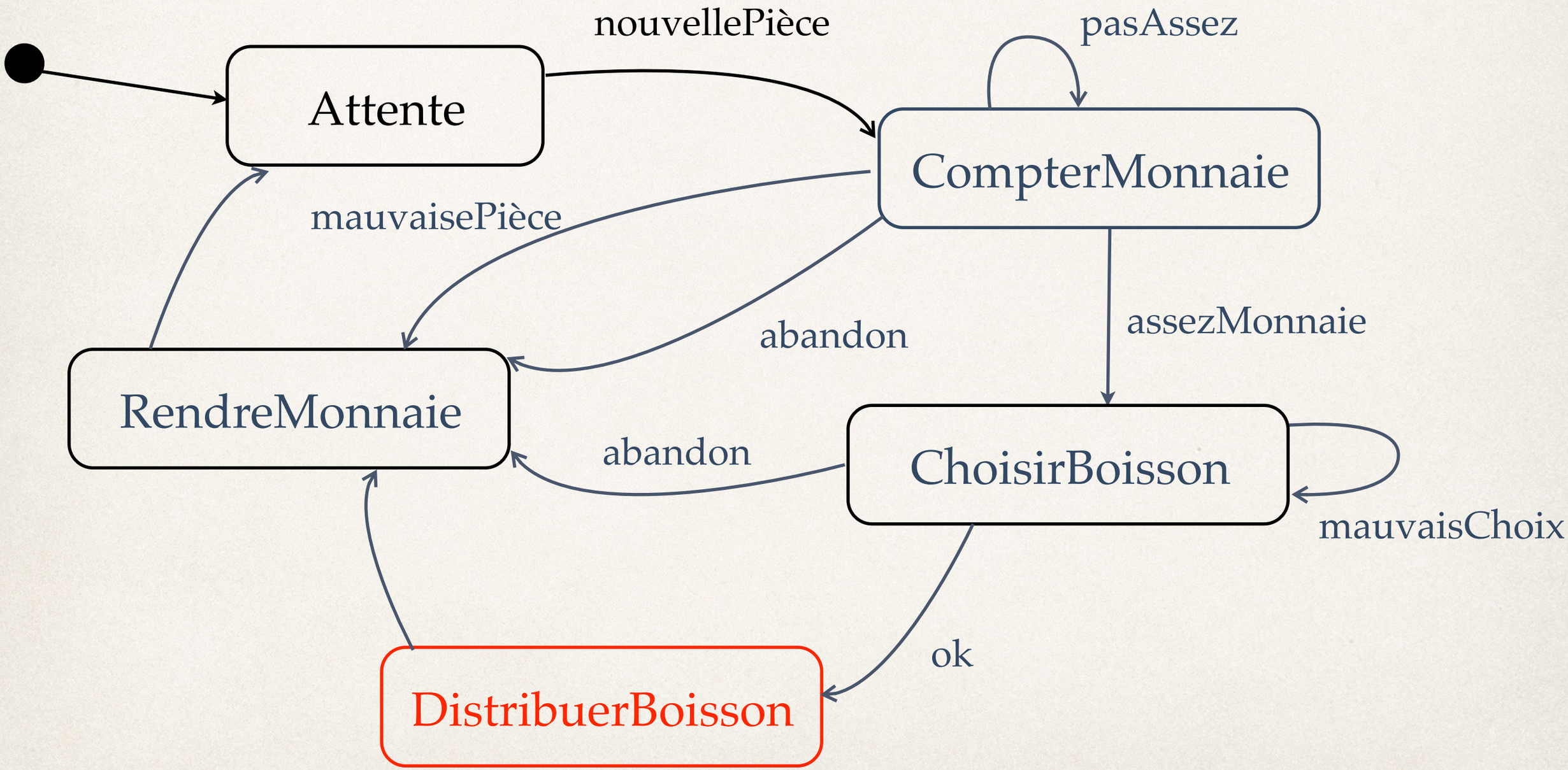
Exemple exécution



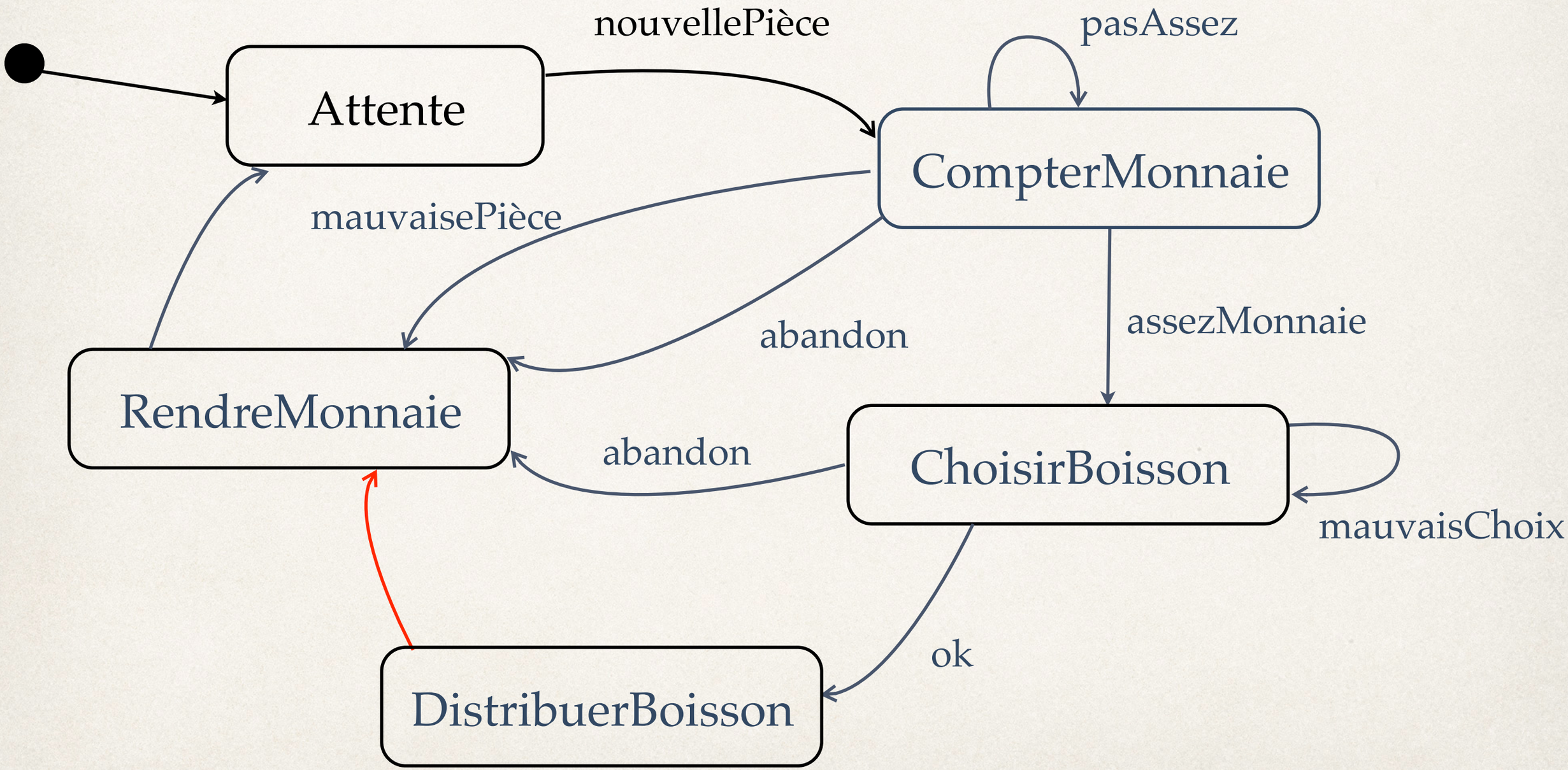
Exemple exécution



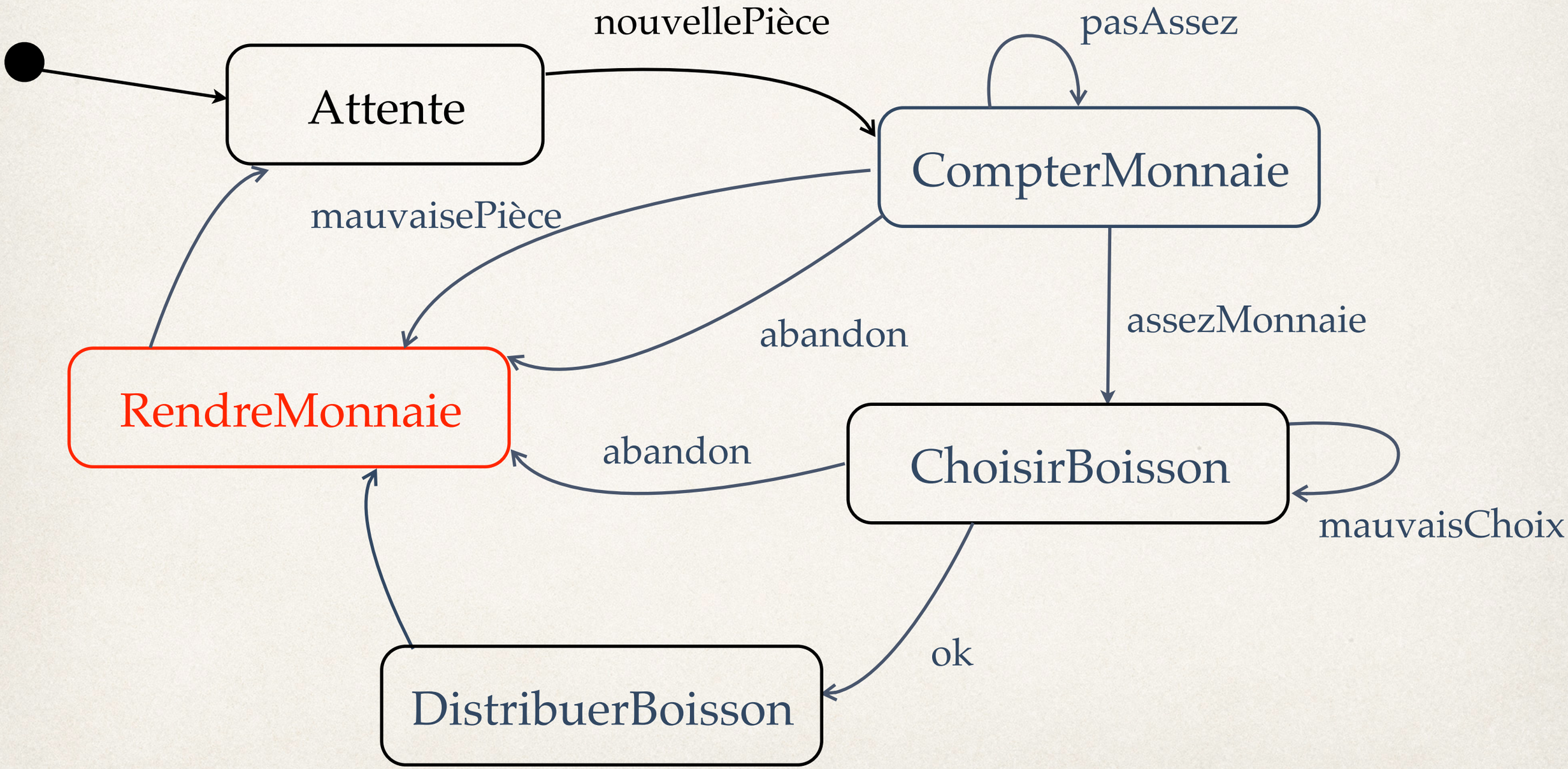
Exemple exécution



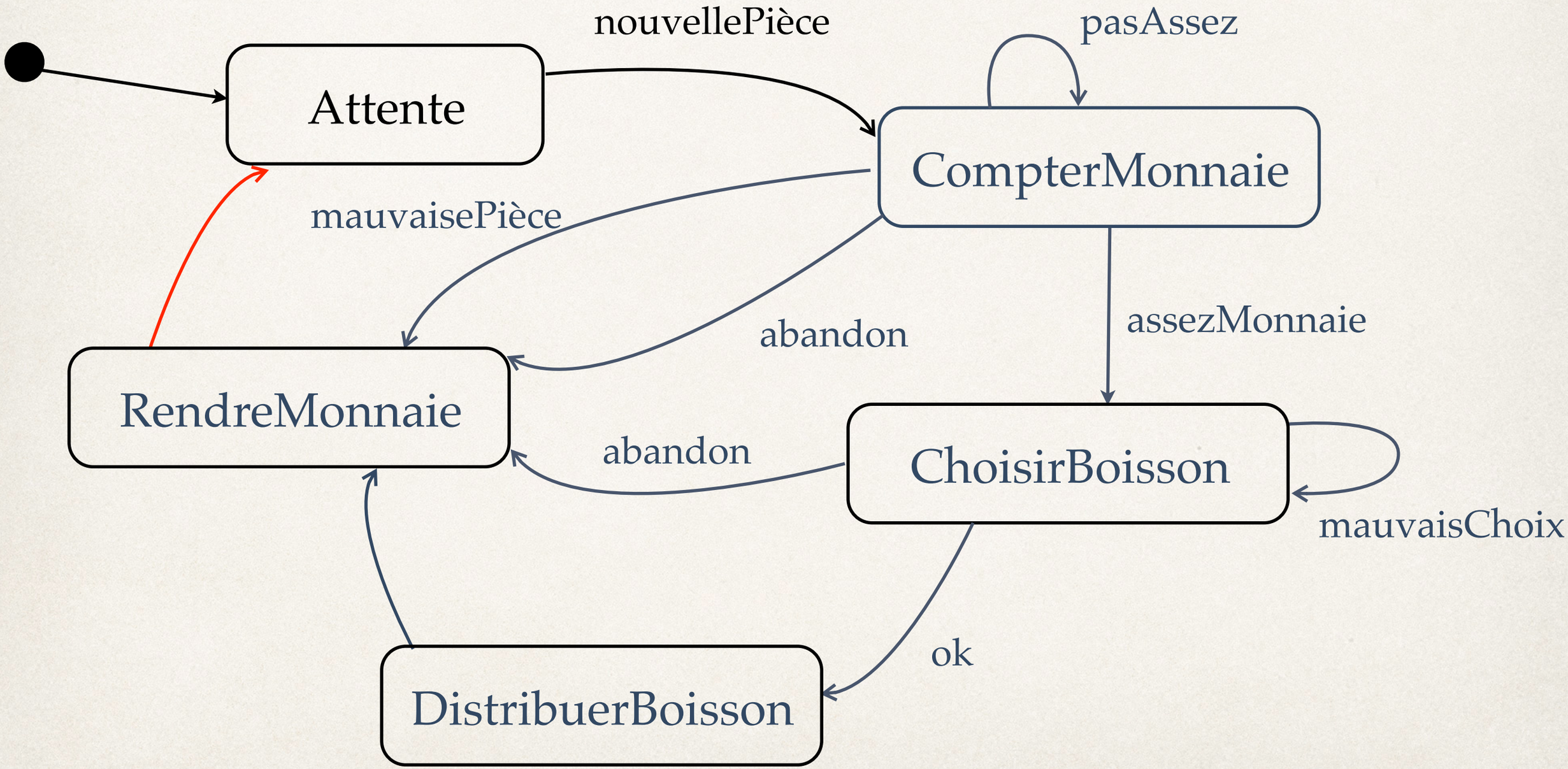
Exemple exécution



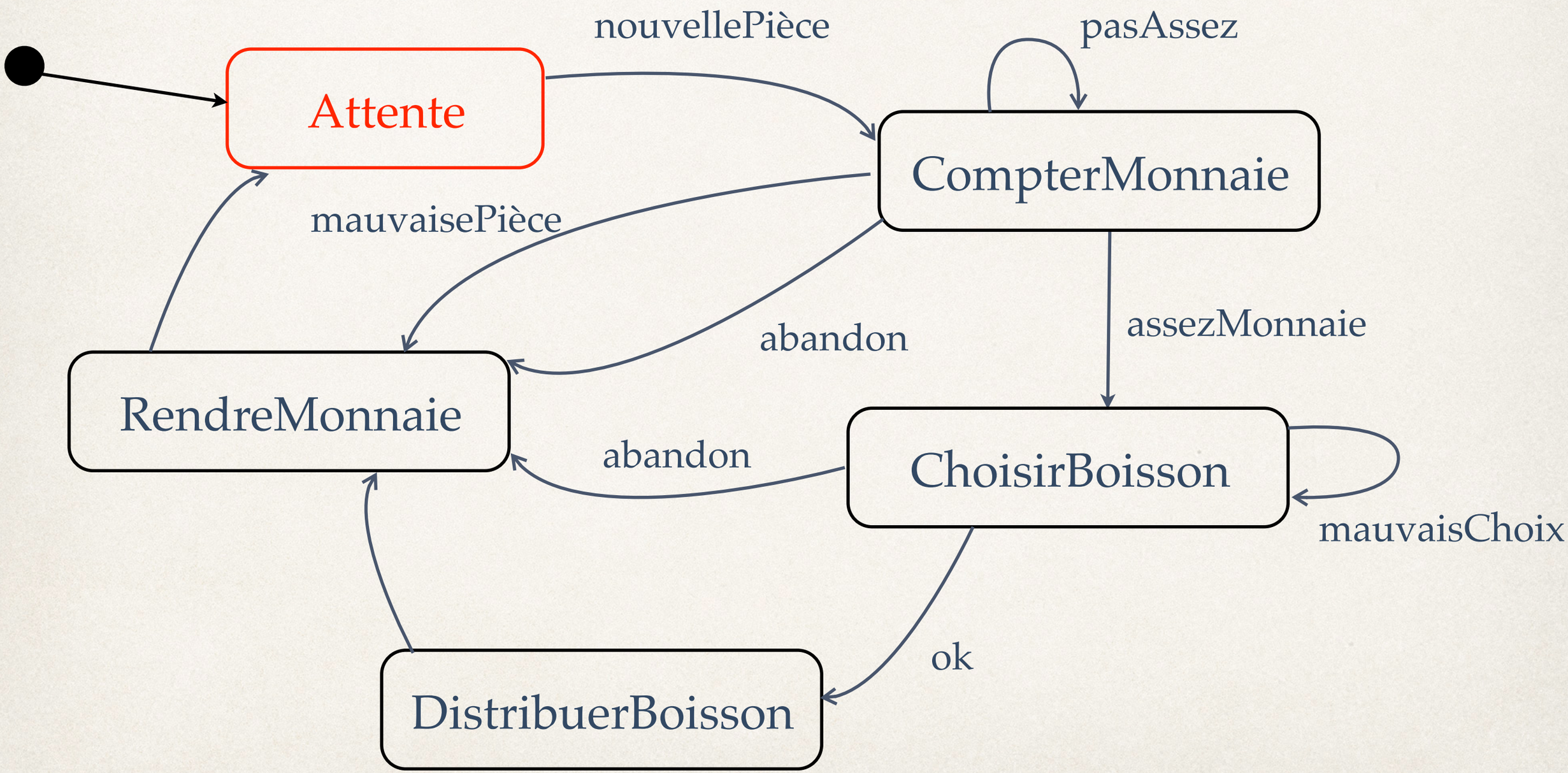
Exemple exécution



Exemple exécution



Exemple exécution

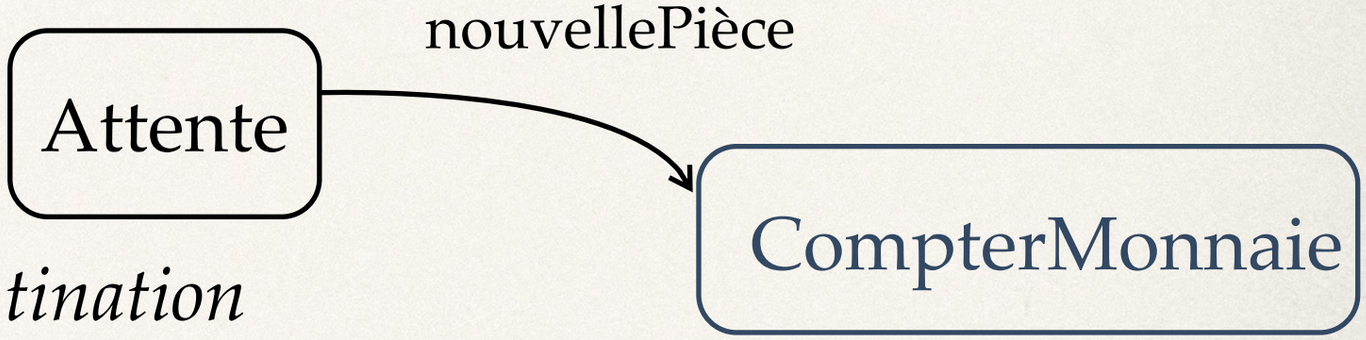


Etat

- décrit une situation de l'objet et du système
- une façon de garder de l'information sur le passé
- états
  - **initial** - l'exécution de la machine à états commence avec ça dans le cas de la machine à état d'un objet activé dans le constructeur
    max 1/machine à états
  - **final** - l'exécution de la machine à états se termine ici
    une fois atteint l'objet est détruit
    il peut y avoir plusieurs / machine à états
- Il peut avoir des actions qui lui sont associées

Transition

- Décrit un échange d'états



- A un état *source* et un état *destination*

- Peut être *déclenchée* par la **réception d'un événement**

- Peut être *conditionnée* par une **garde**

- Peut être *spontanée*, i.e. pas de condition, pas de garde elle est déclenchée par la terminaison de l'action effectué dans son état source

- Peut avoir une **action** qui lui est associée

Action

- Décrit un bout de fonctionnalité
  (affectation, envoi de signal, appel opération, etc.)

- Le moment d'exécution d'un action permet de les classifier en:

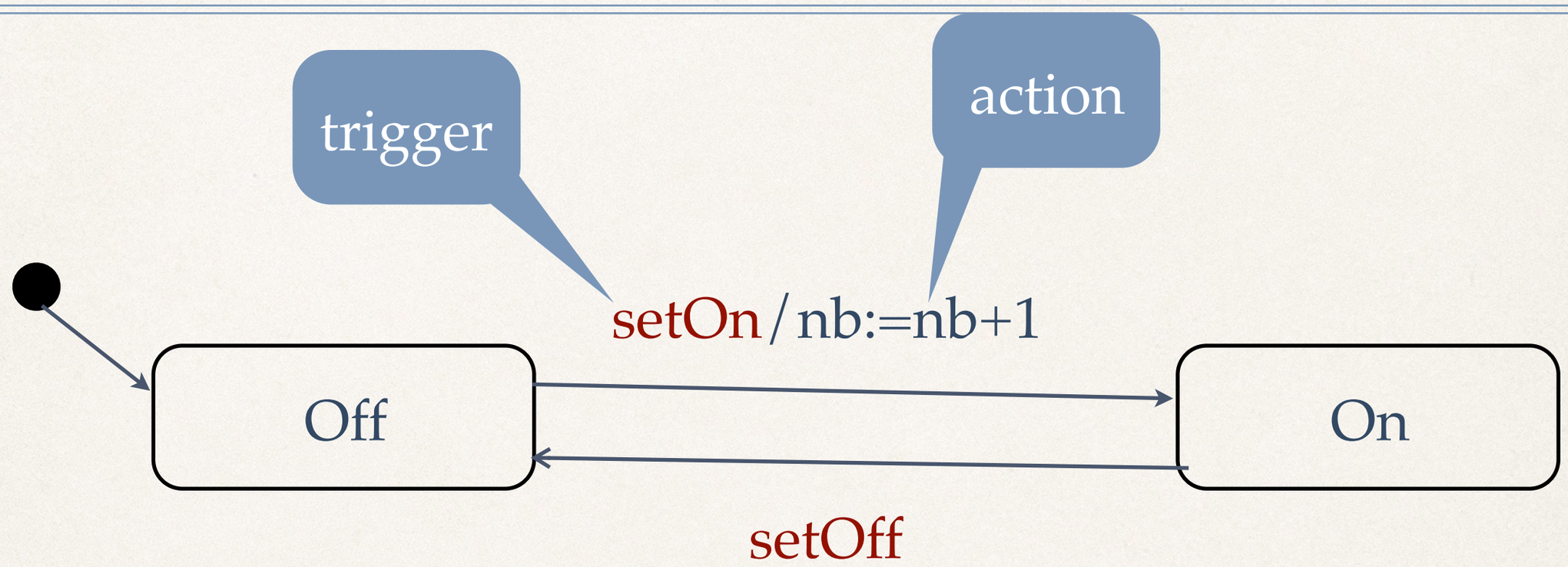
  - Entry action - action qui s'exécute à l'entrée dans un état

  - Exit action - action qui s'exécute juste avant la sortie d'un état

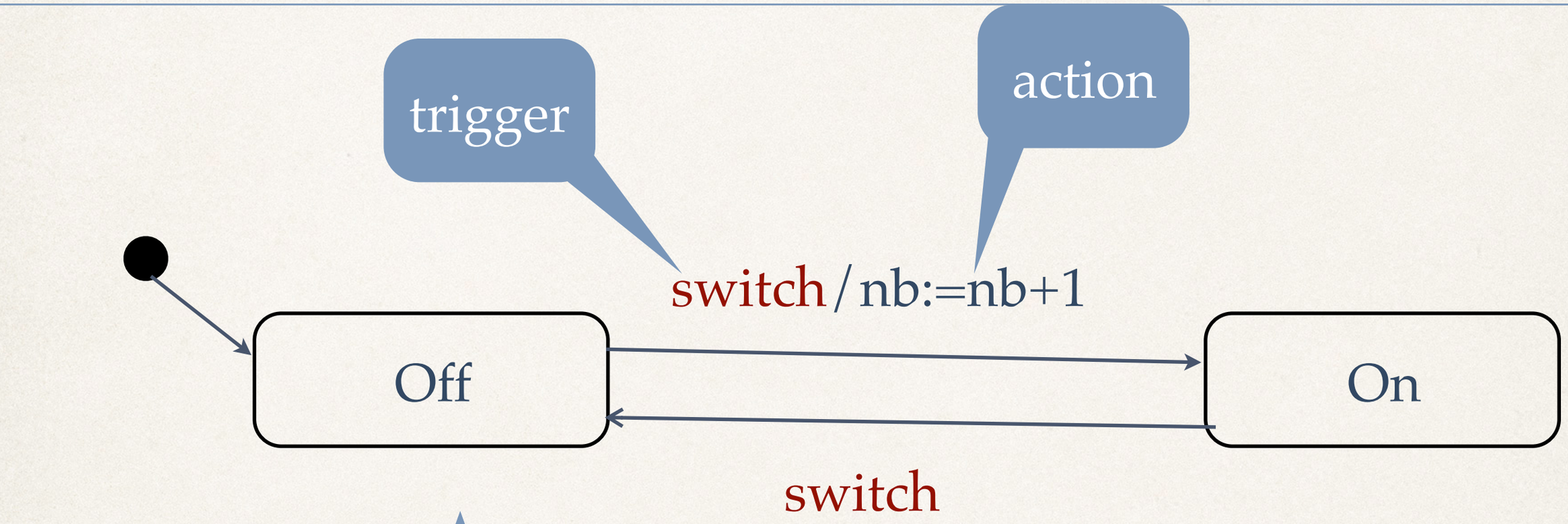
  - Do action - action qui s'exécute pendant la résidence dans un état

  - Action associée à une transition

Exemple simple

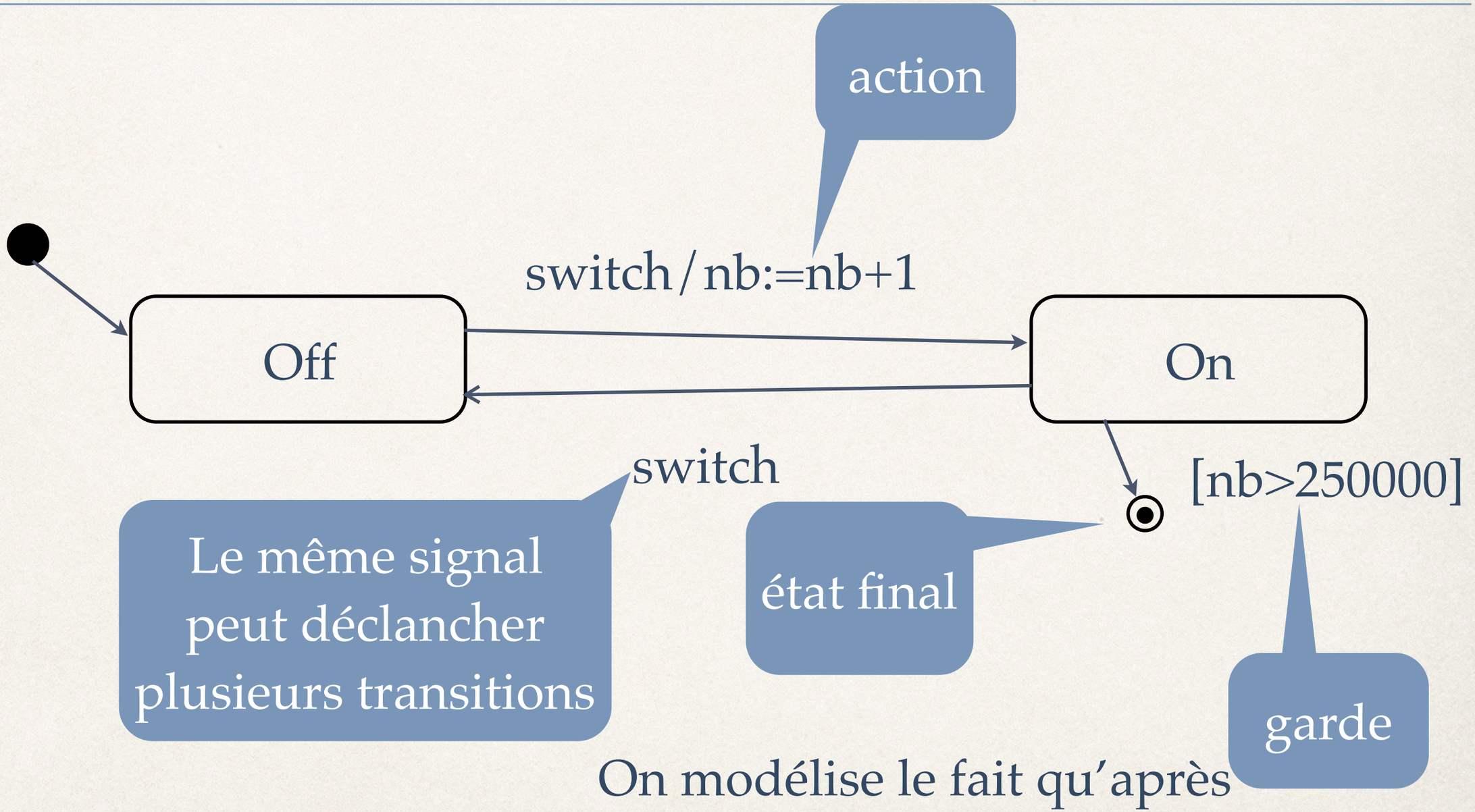


Variation - interrupteur à un bouton



le même signal apparaît sur les 2 transitions

Variation - interrupteur à un bouton



On modélise le fait qu'après 250000 allumages, l'interrupteur ne marche plus

Evénement

- quelque chose d'important qui se passe dans le système ou son environnement

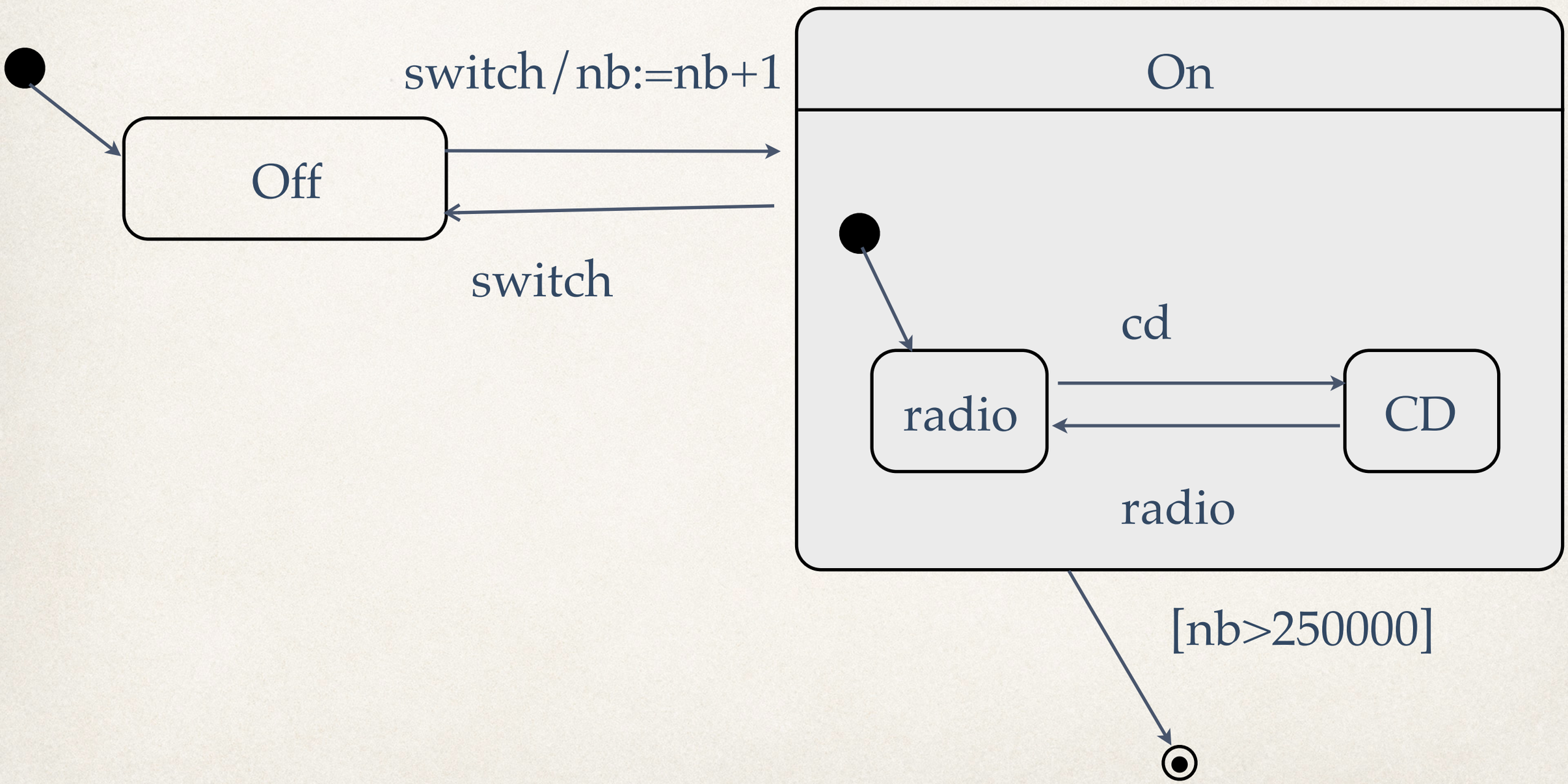
- utilisé comme trigger (déclencheur) des transitions

- types
  signal event - la réception d'un signal
  call event - la réception d'un appel (d'opération)
  time event - la réception d'un signal lié au temps
  change event - le fait qu'une variable ait changée de valeur
  ....

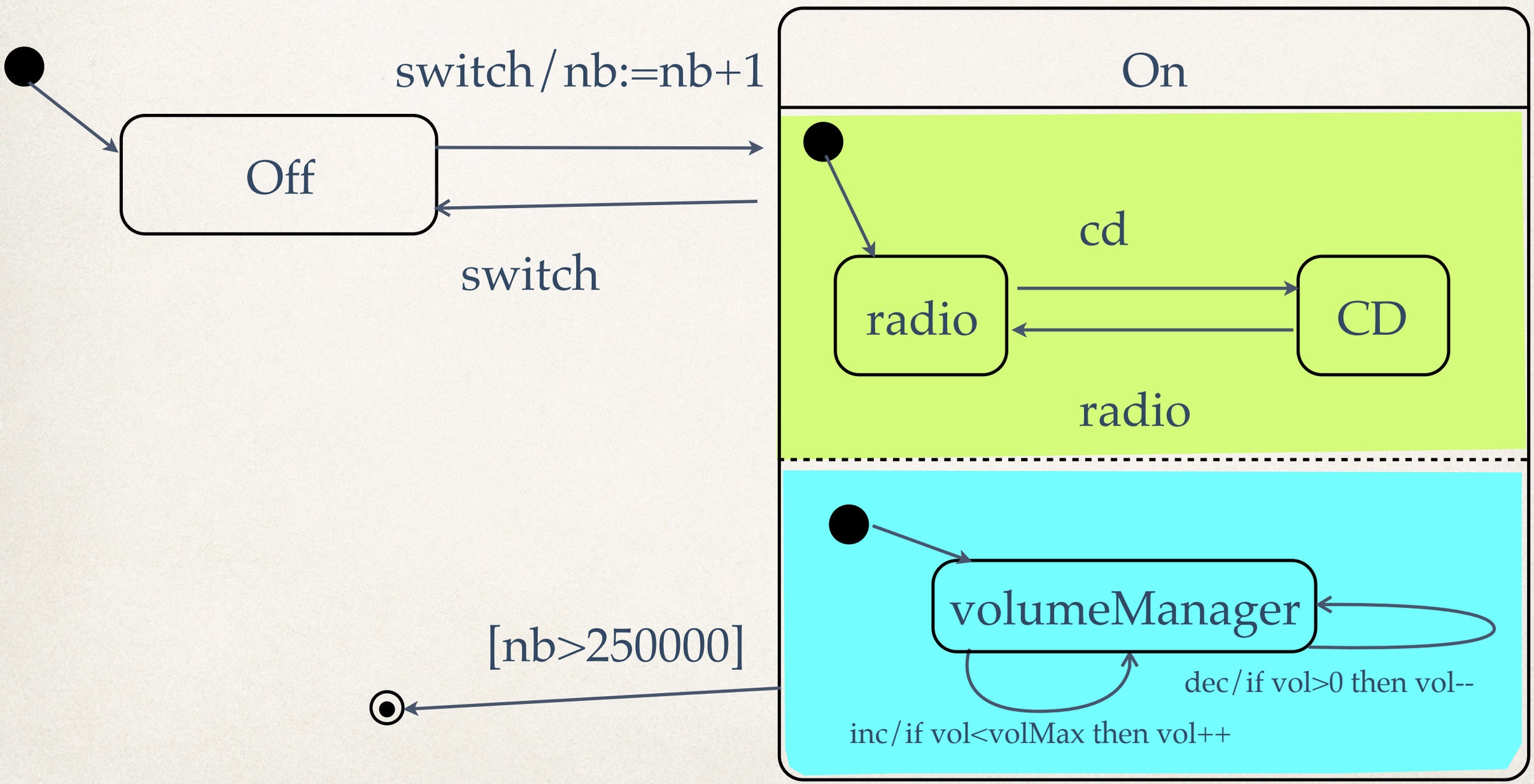
Etats hiérarchiques

- Avec tous les états au même niveau - la modélisation ne reflète pas les parties communes qui peuvent exister entre états
- Etats hiérarchiques - permettent de structurer les machines à états
  - Etat hiérarchique de type “or” - un état qui contient une sous-machine à états
  - Etat hiérarchique de type “and” - un état qui contient 2+ machines à états concurrentes

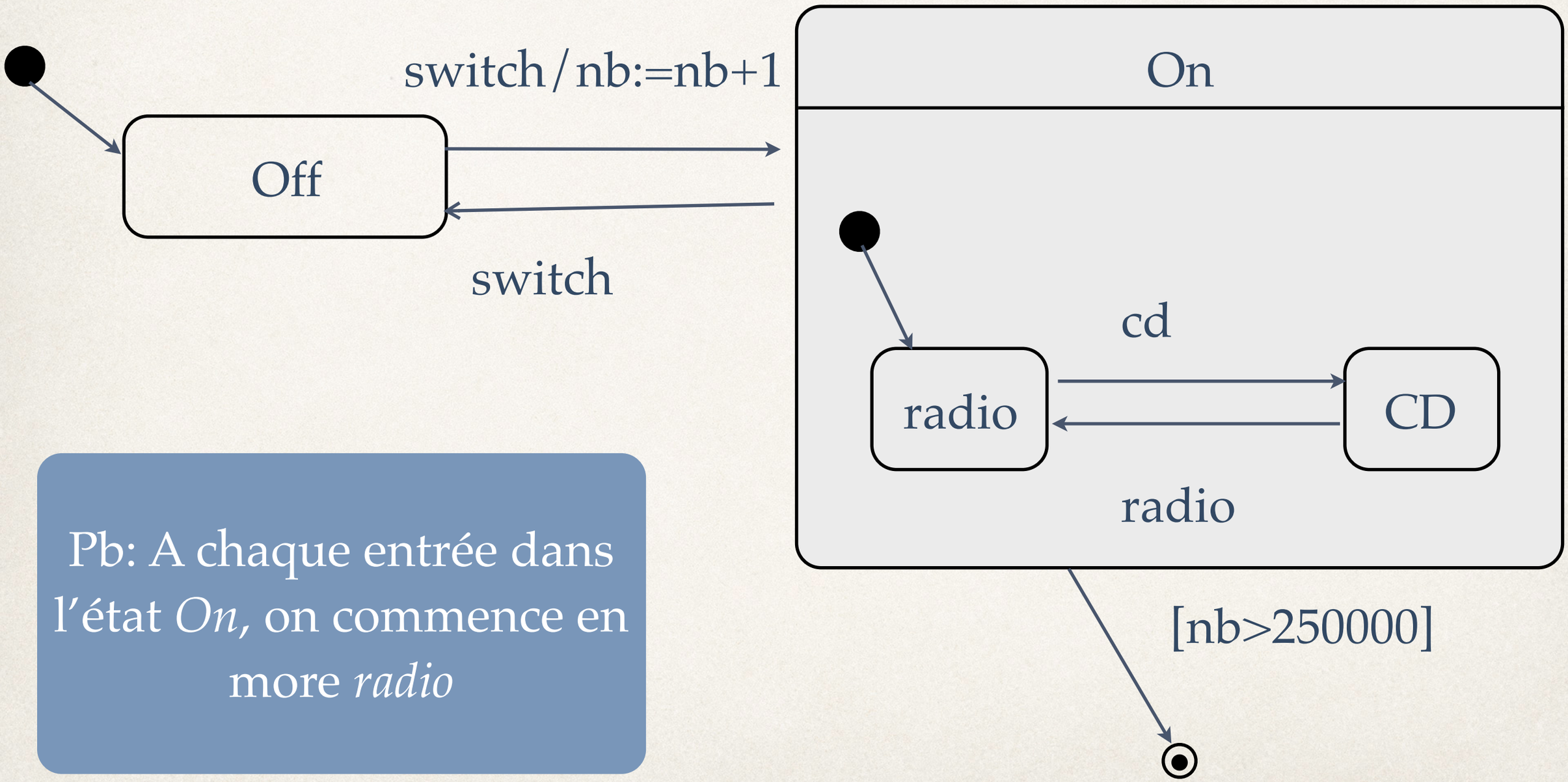
Exemple d'état composé (type *or*)



Exemple d'état composé parallèle (type *and*)

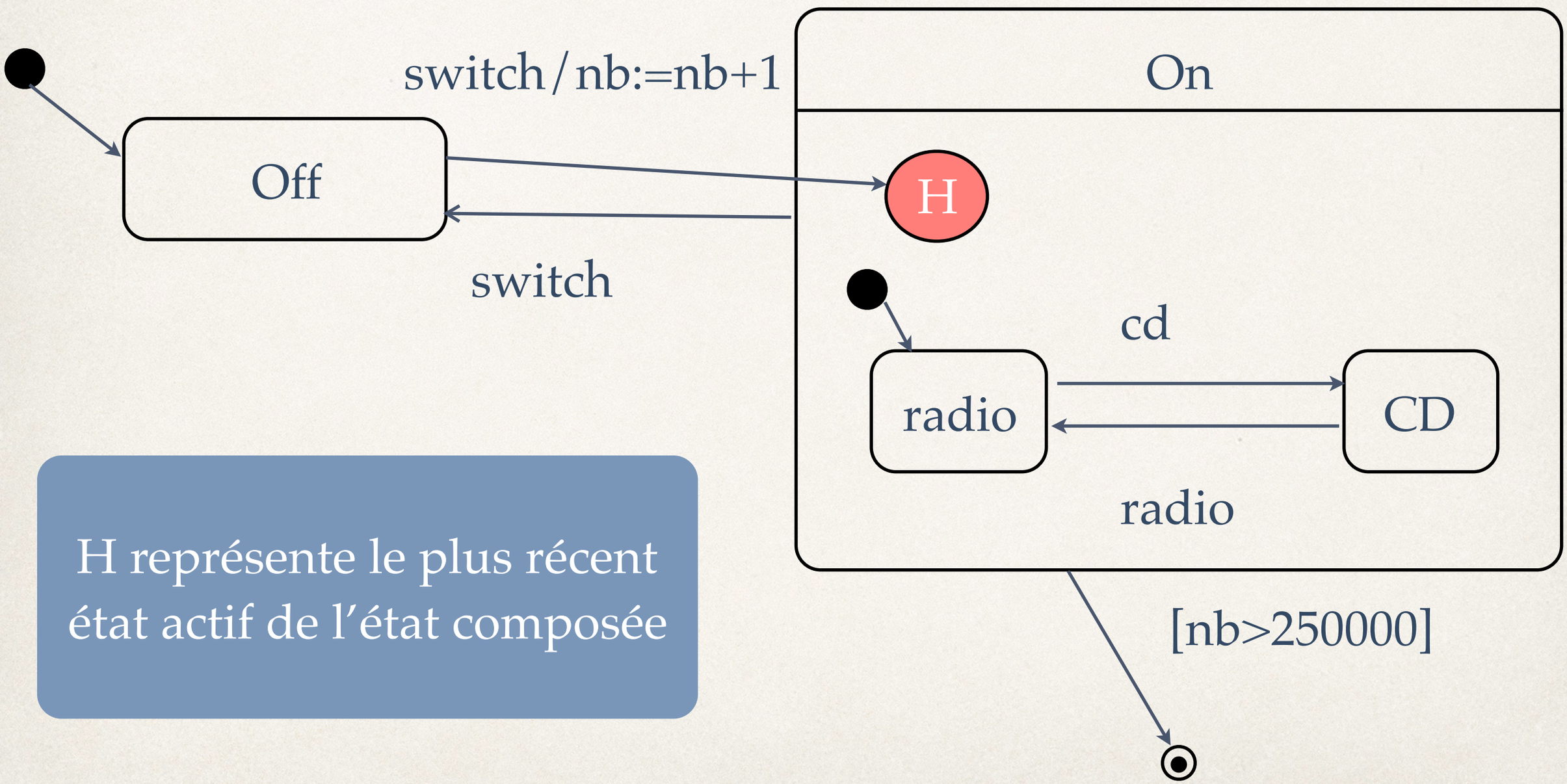


Amélioration possible



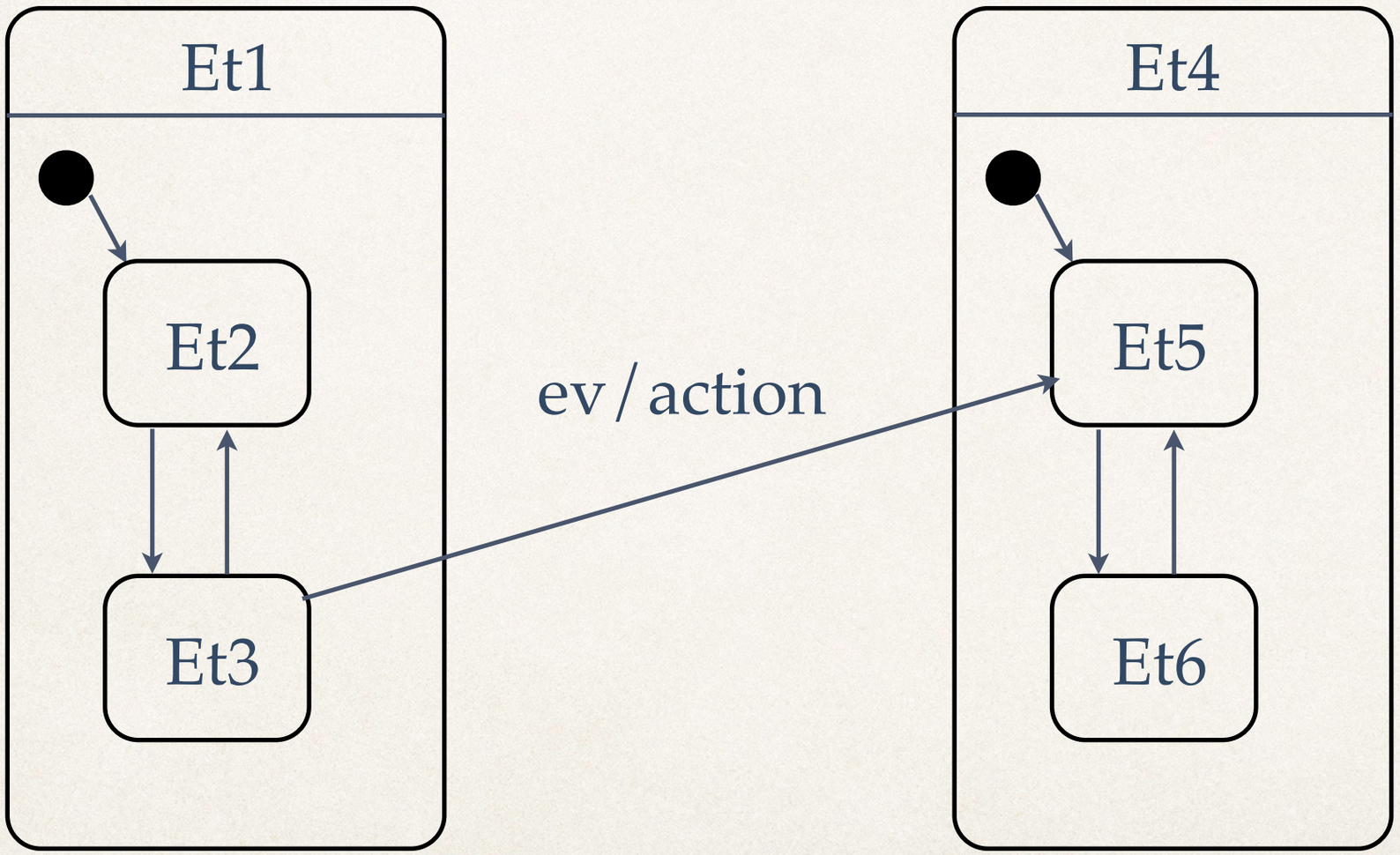
Pb: A chaque entrée dans l'état *On*, on commence en more *radio*

Etat histoire

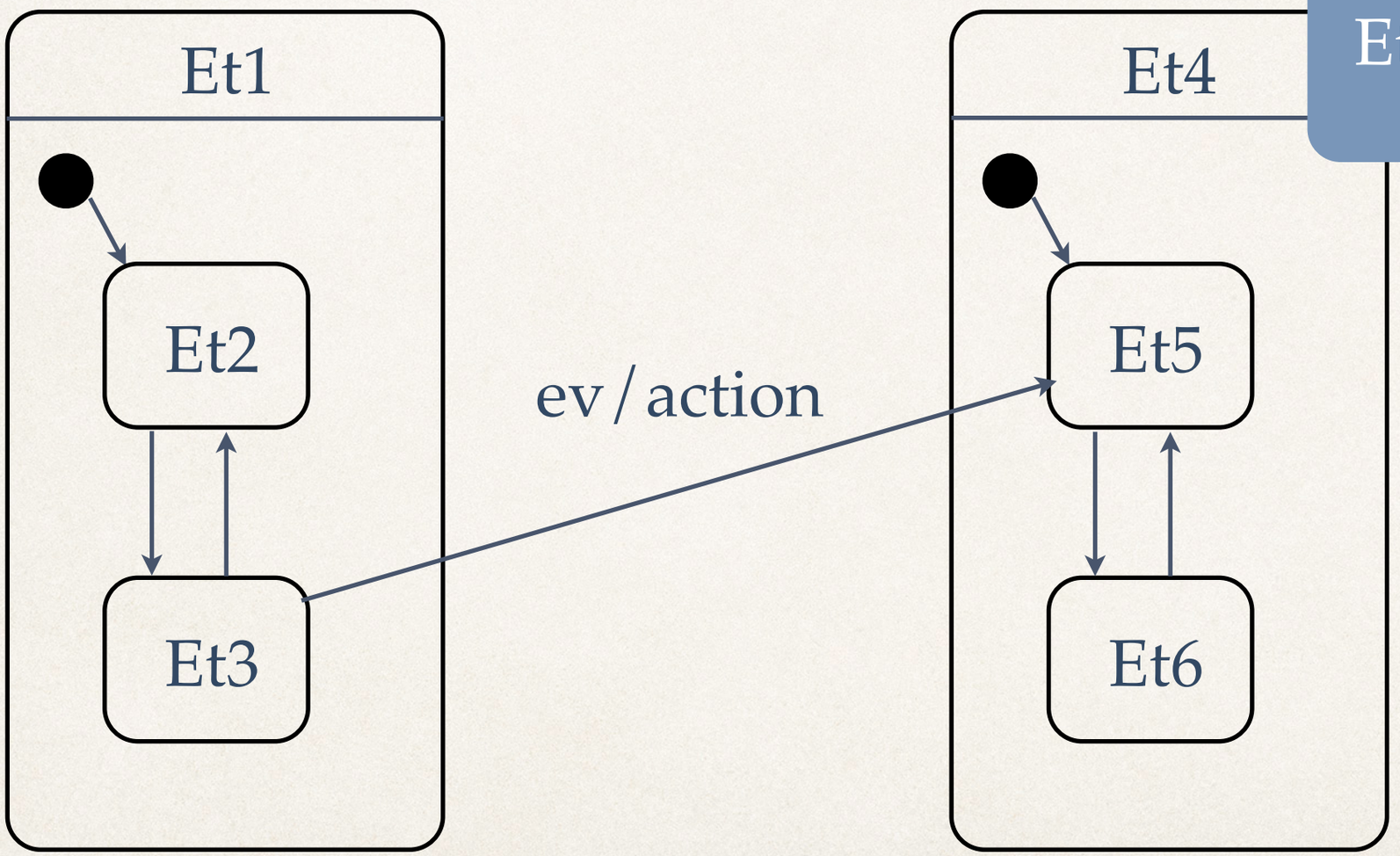


H représente le plus récent état actif de l'état composée

Changement d'état - un problème complexe



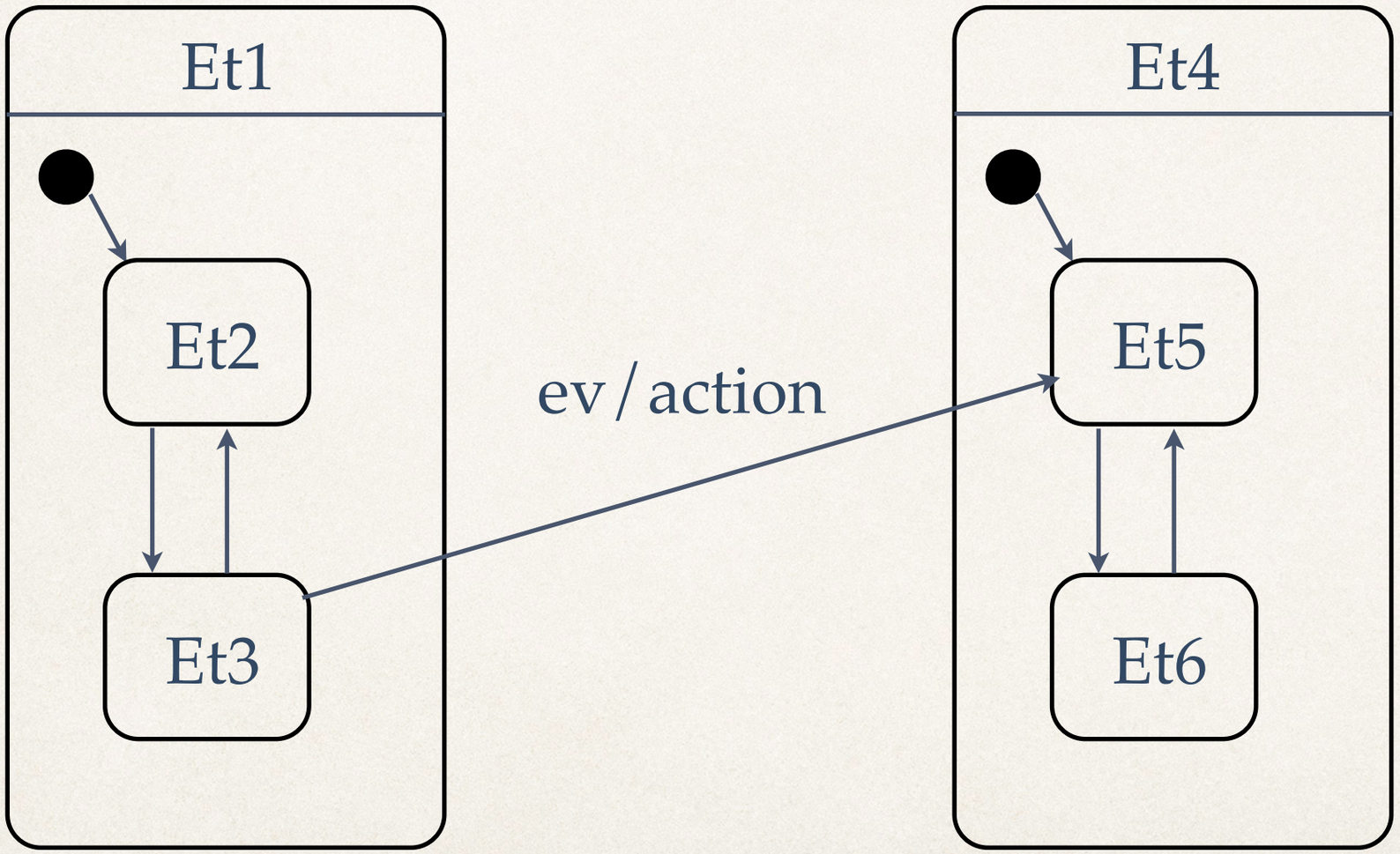
Changement d'état - un problème complexe



Est-ce que la transition Et3->Et4 est autorisée?

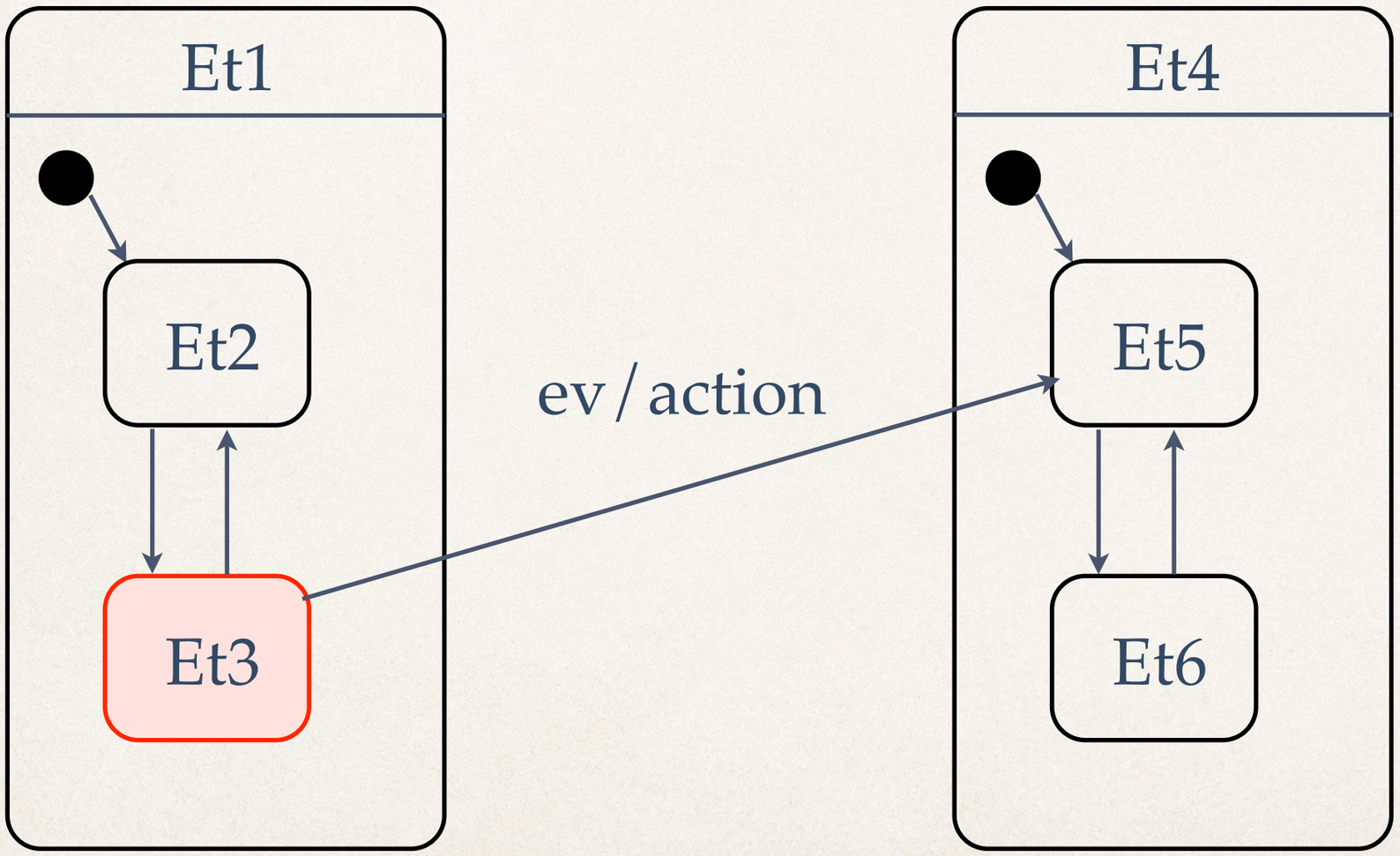
Oui

Changement d'état - un problème complexe



Lors de la transition Et3→Et4 l'enchaînement des actions est:

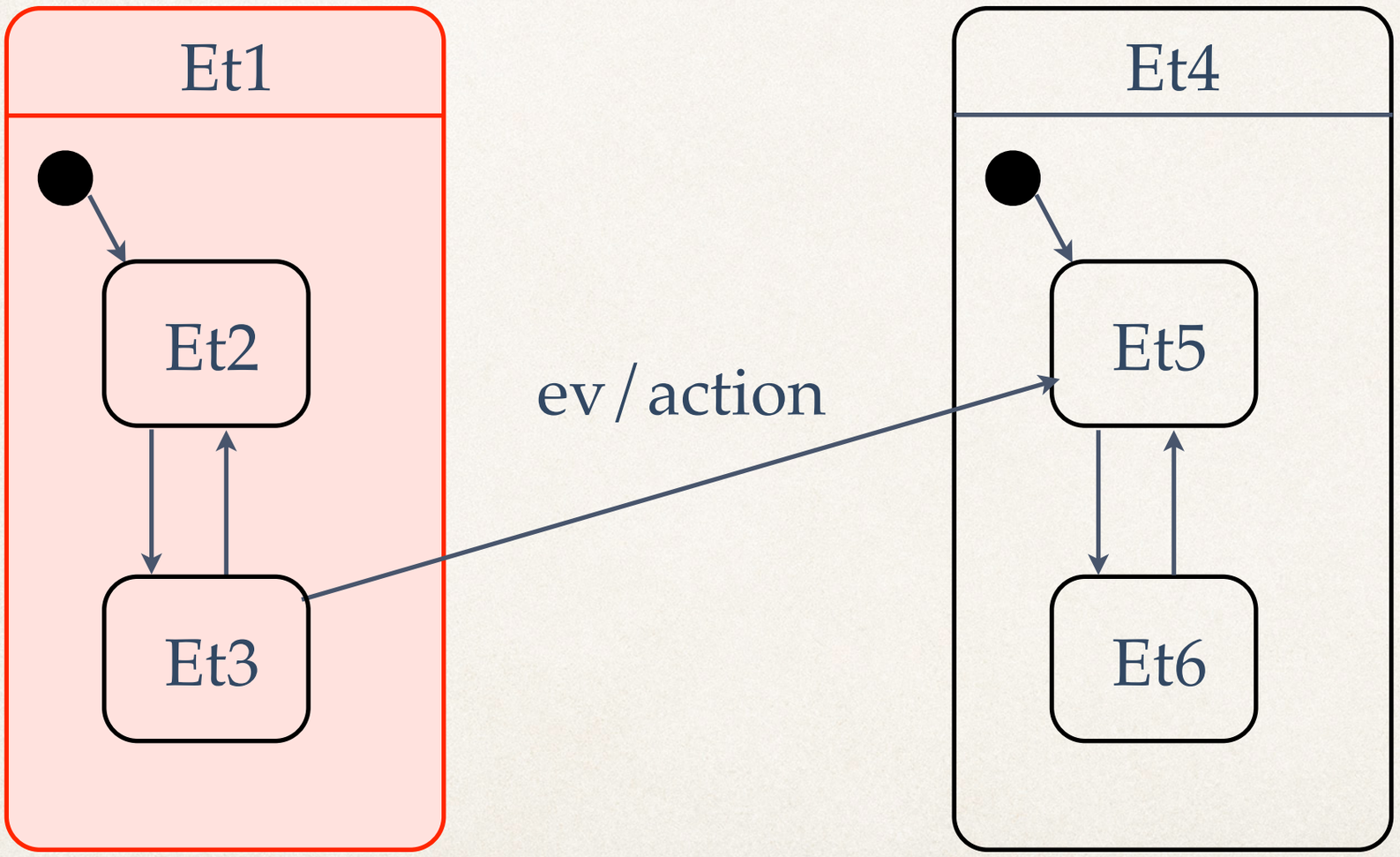
Changement d'état - enchaînement des actions



Lors de la transition Et3→Et4 l'enchaînement des actions est:

ExitAction Et3

Changement d'état - enchaînement des actions

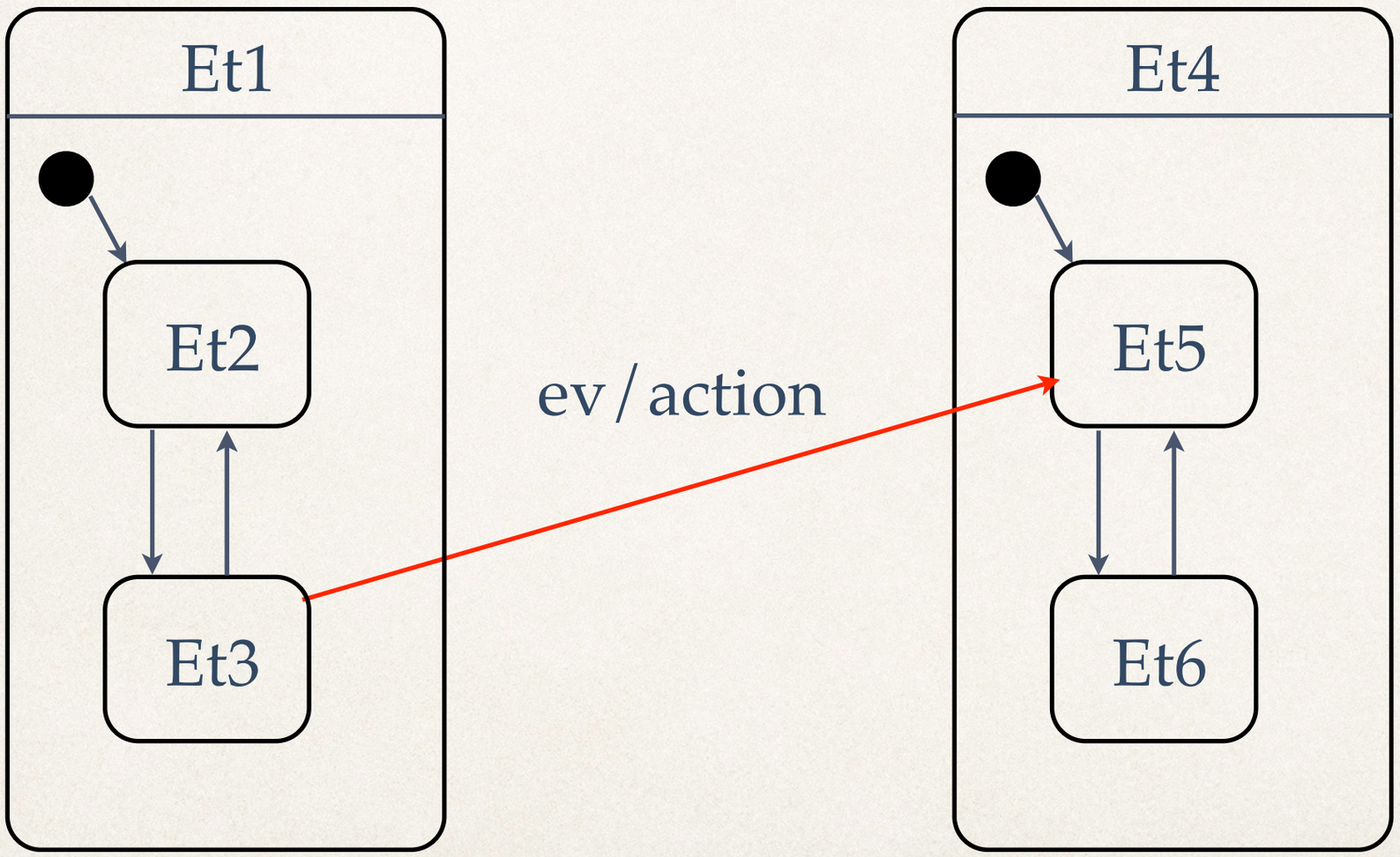


Lors de la transition Et3→Et4 l'enchaînement des actions est:

ExitAction Et3

ExitAction Et1

Changement d'état - enchaînement des actions



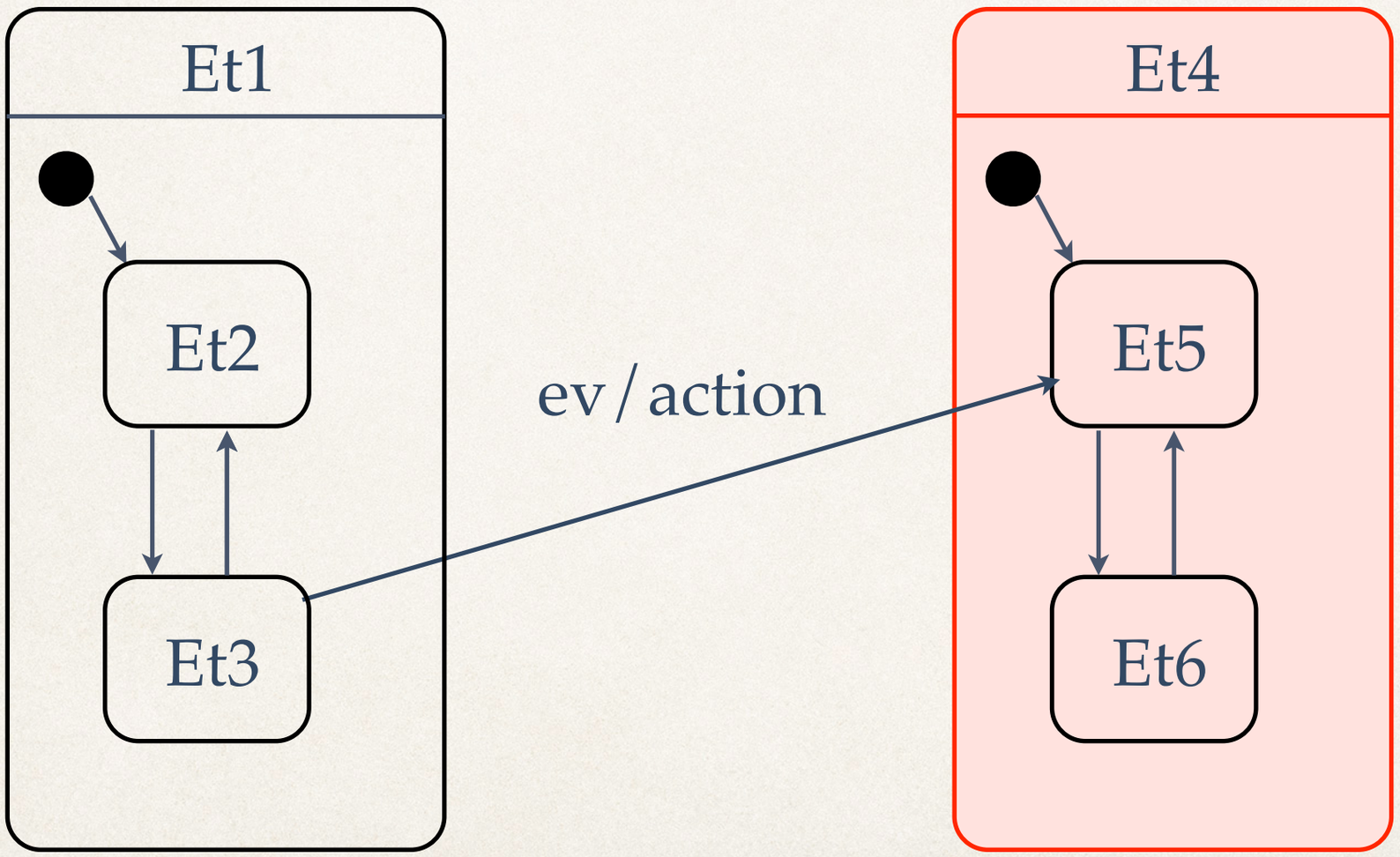
Lors de la transition Et3→Et4 l'enchaînement des actions est:

ExitAction Et3

ExitAction Et1

action

Changement d'état - enchaînement des actions



Lors de la transition Et3→Et4 l'enchaînement des actions est:

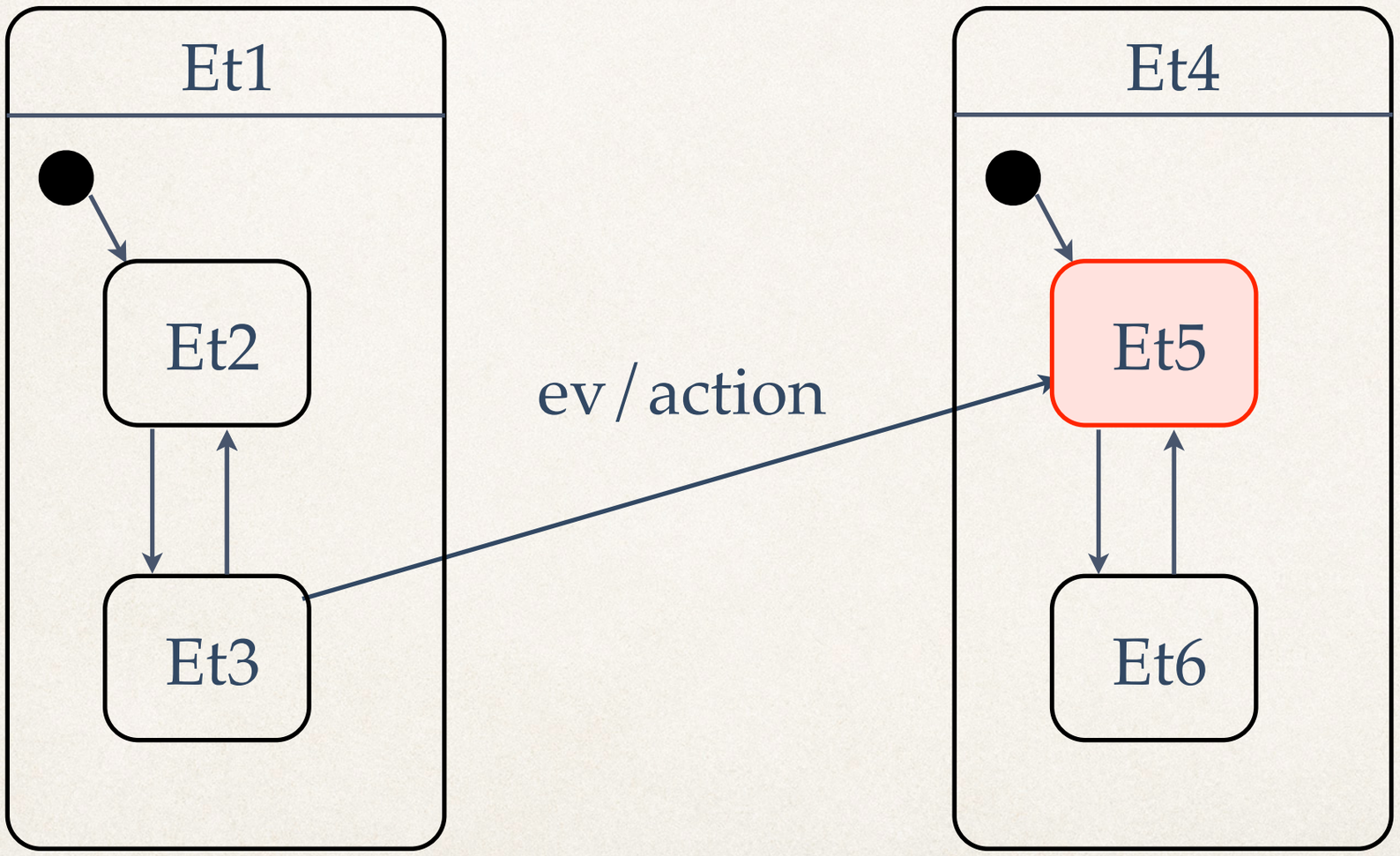
ExitAction Et3

ExitAction Et1

action

EntryAction Et4

Changement d'état - enchaînement des actions



Lors de la transition Et3→Et4 l'enchaînement des actions est:

ExitAction Et3

ExitAction Et1

action

EntryAction Et4

EntryAction Et5

Plan

- Principes de la communication en UML
- Description du comportement en UML
- Notions de base dans les machines à états
- **Utilisation des machines à états**

Utilisation des diagrammes états-transition

- En phase d'analyse
  - Description de la dynamique du systèmes vu de l'extérieur
  - Synthèse de scénarios des cas d'utilisation
  - Evénements = actions des acteurs
- En phase de conception
  - Description de la dynamique d'un objet (d'une classe)
  - Evénements = appels d'opérations, interactions avec l'environnement

Utilisation des machines à états?

- pour quels systèmes?
  pour des systèmes réactifs (comportement fortement dépendant de ce qui se passe dans son environnement)

- pourquoi les utiliser?
  Si on les utilise judicieusement:
    facile à lire
    donnent des résultats de vérification intéressants
    code généré plus efficace

- Attention la spécification du comportement avec des machines à états ne se prête pas à tout système (e.g. compilateur)

Conclusions

- La modélisation peut couvrir le comportement
- Les machines à états permettent de décrire le comportement des systèmes réactives
- Les modèles de comportement sont utilisés comme entrée pour des outils de validation et pour des générateurs de code