# C10 : Modélisation de la chaine d'information des systèmes
# C10-2 : Modélisation des systèmes à évènements discrets

Émilien DURIF

Lycée La Martinière Monplaisir Lyon
Classe de MPSI
14 Juin 2022

# Plan

# Plan

# Modélisation de la commande des systèmes

## Introduction

- Le chapitre précédent à permis de donner les bases de l'**étude des systèmes numériques** avec la manipulation des informations binaires.
- Ces informations sont **traitées** par des **cartes de commandes** qui sont généralement insérées dans des **micro-contrôleur**.
- Il faut maintenant faire le lien entre le **cahier des charges** et le **fonctionnement attendu du système** en mettant en relation toutes les informations issues de commandes ou de mesures numériques.
- Dans ce chapitre nous développerons les bases de lecture des **diagrammes comportementaux d'états et d'activités** nécessaires pour la commande et la simulation du fonctionnement d'un **système complexe**.

# Modélisation de la commande des systèmes

## Introduction

- Le chapitre précédent à permis de donner les bases de l'**étude des systèmes numériques** avec la manipulation des informations binaires.
- Ces informations sont **traitées** par des **cartes de commandes** qui sont généralement insérées dans des **micro-contrôleur**.
- Il faut maintenant faire le lien entre le **cahier des charges** et le **fonctionnement attendu du système** en mettant en relation toutes les informations issues de commandes ou de mesures numériques.
- Dans ce chapitre nous développerons les bases de lecture des **diagrammes comportementaux d'états et d'activités** nécessaires pour la commande et la simulation du fonctionnement d'un **système complexe**.

# Modélisation de la commande des systèmes

## Introduction

- Le chapitre précédent à permis de donner les bases de l'**étude des systèmes numériques** avec la manipulation des informations binaires.
- Ces informations sont **traitées** par des **cartes de commandes** qui sont généralement insérées dans des **micro-contrôleur**.
- Il faut maintenant faire le lien entre le **cahier des charges** et le **fonctionnement attendu du système** en mettant en relation toutes les informations issues de commandes ou de mesures numériques.
- Dans ce chapitre nous développerons les bases de lecture des **diagrammes comportementaux d'états et d'activités** nécessaires pour la commande et la simulation du fonctionnement d'un **système complexe**.

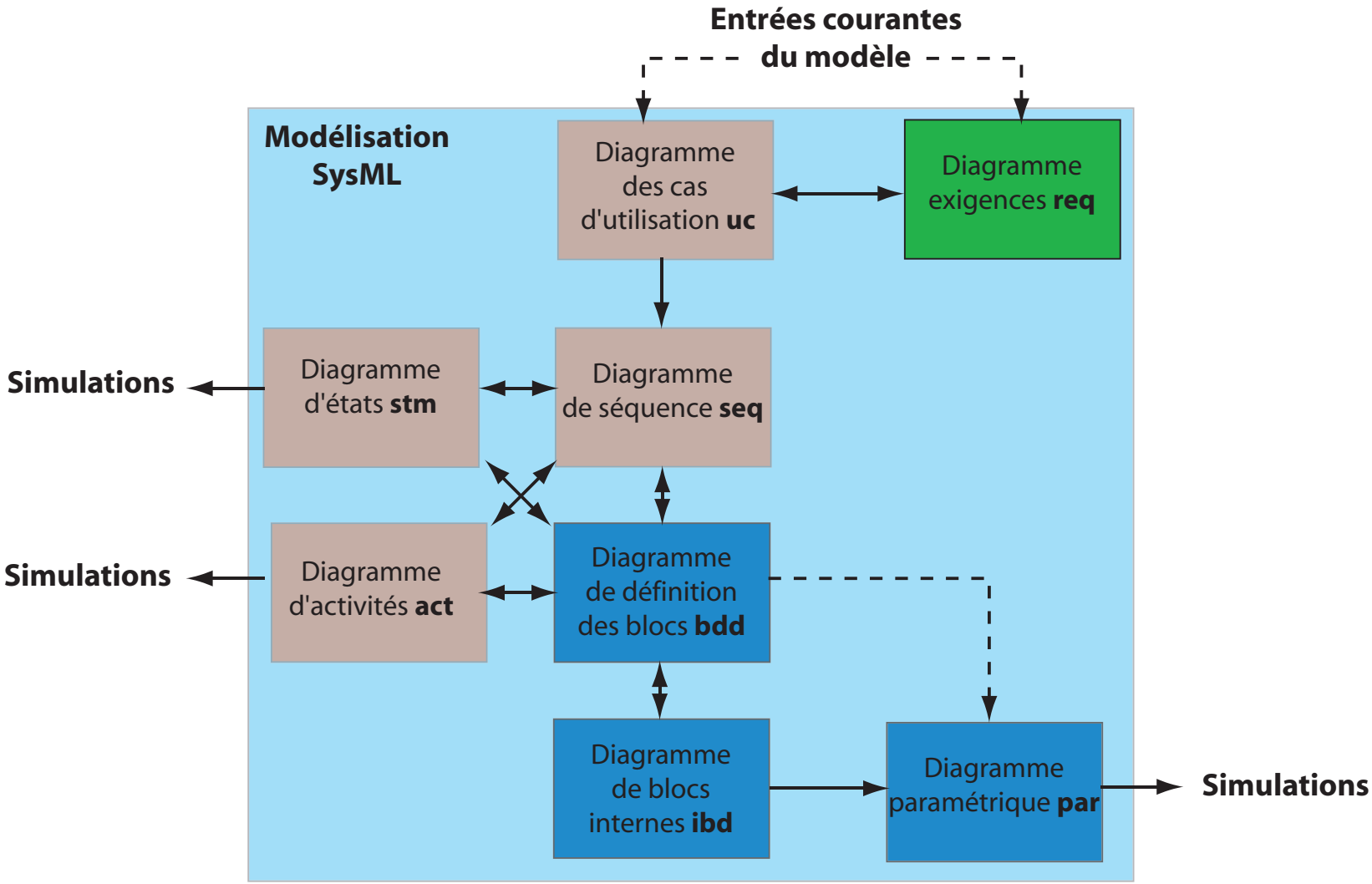
# Modélisation de la commande des systèmes

## Introduction

- Le chapitre précédent à permis de donner les bases de l'**étude des systèmes numériques** avec la manipulation des informations binaires.
- Ces informations sont **traitées** par des **cartes de commandes** qui sont généralement insérées dans des **micro-contrôleur**.
- Il faut maintenant faire le lien entre le **cahier des charges** et le **fonctionnement attendu du système** en mettant en relation toutes les informations issues de commandes ou de mesures numériques.
- Dans ce chapitre nous développerons les bases de lecture des **diagrammes comportementaux d'états et d'activités** nécessaires pour la commande et la simulation du fonctionnement d'un **système complexe**.

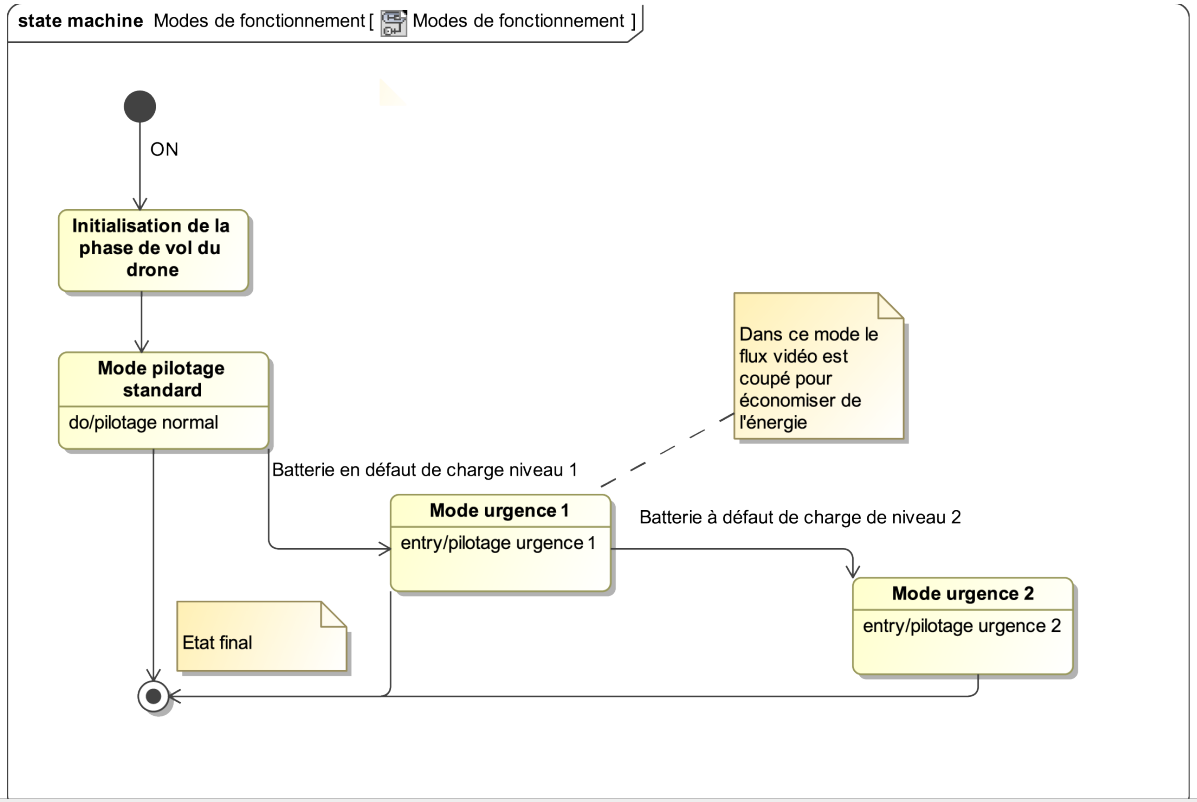
# Plan

# Modélisation à l'aide de l'outil SysML

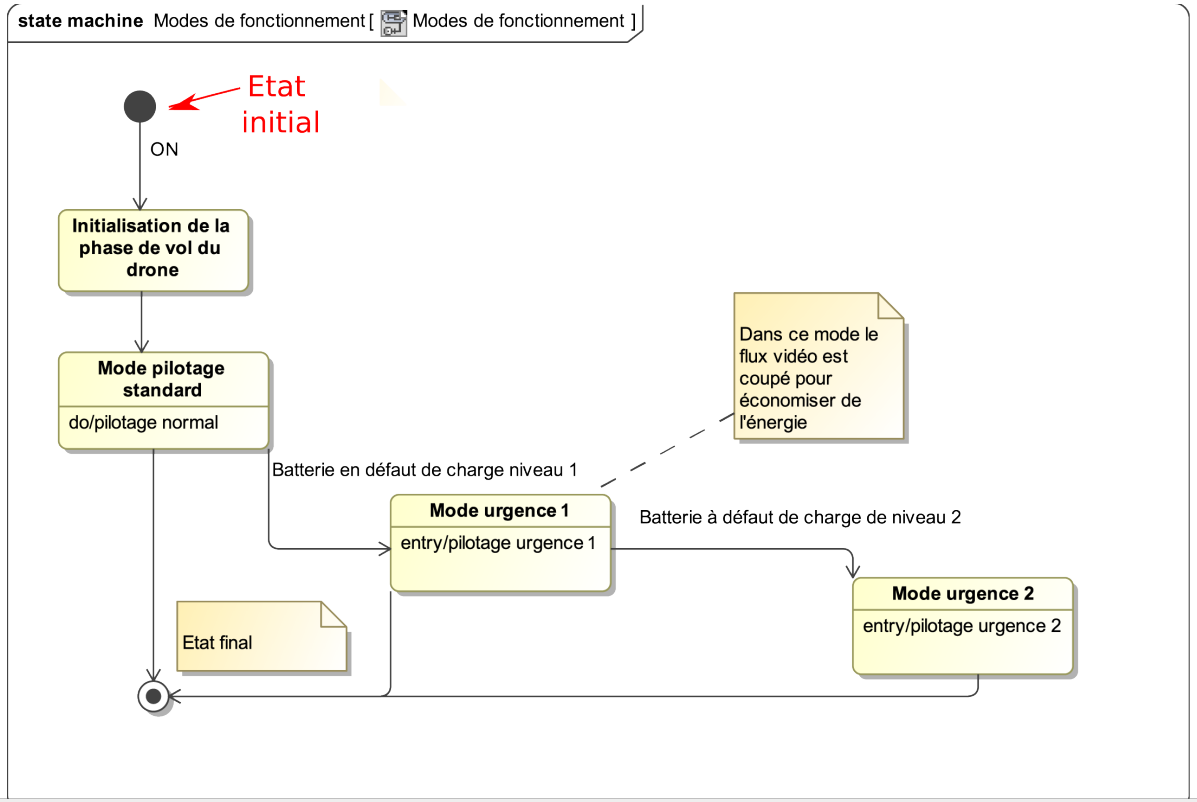


# Plan

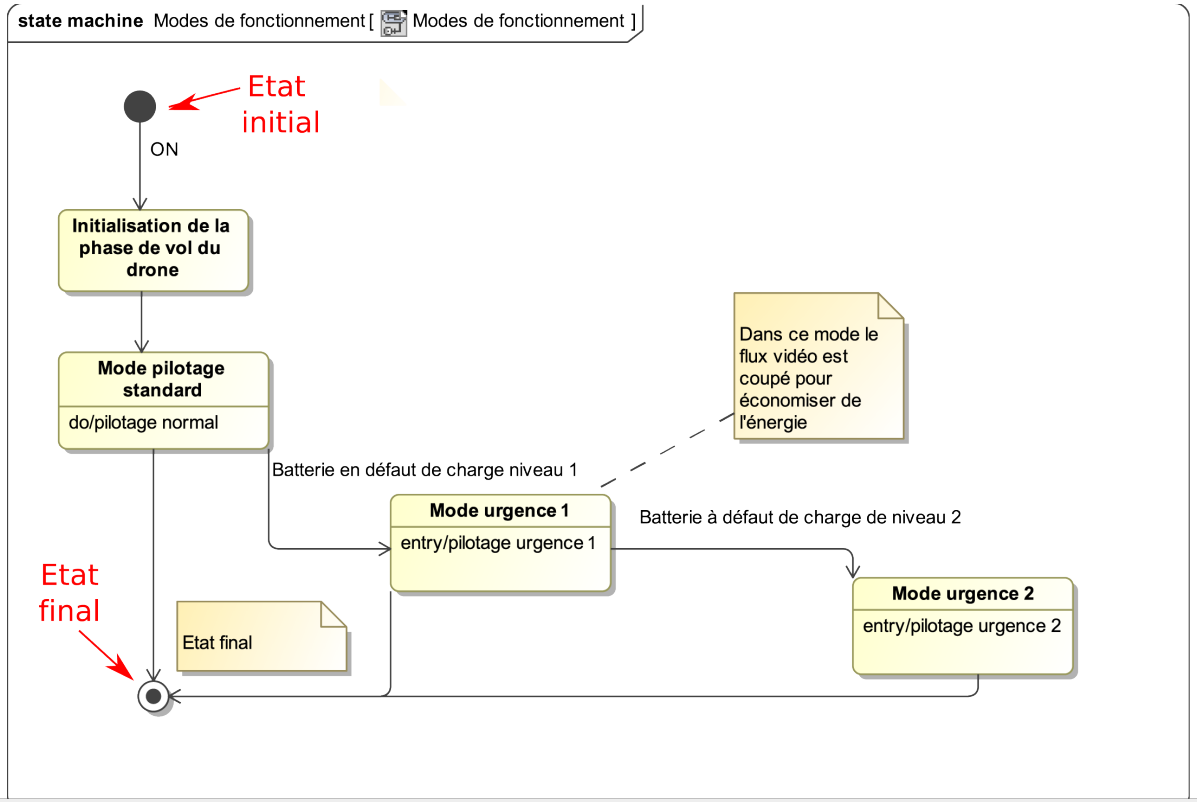
# Diagramme d'états



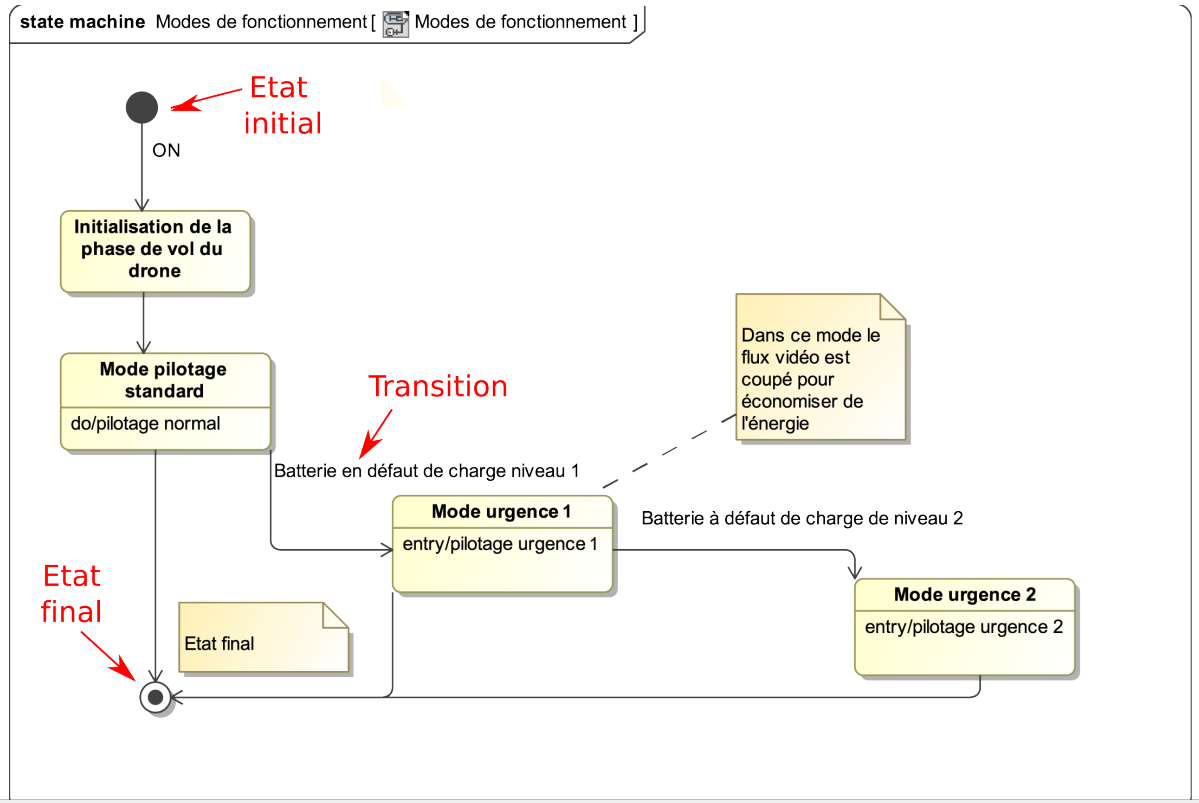
# Diagramme d'états



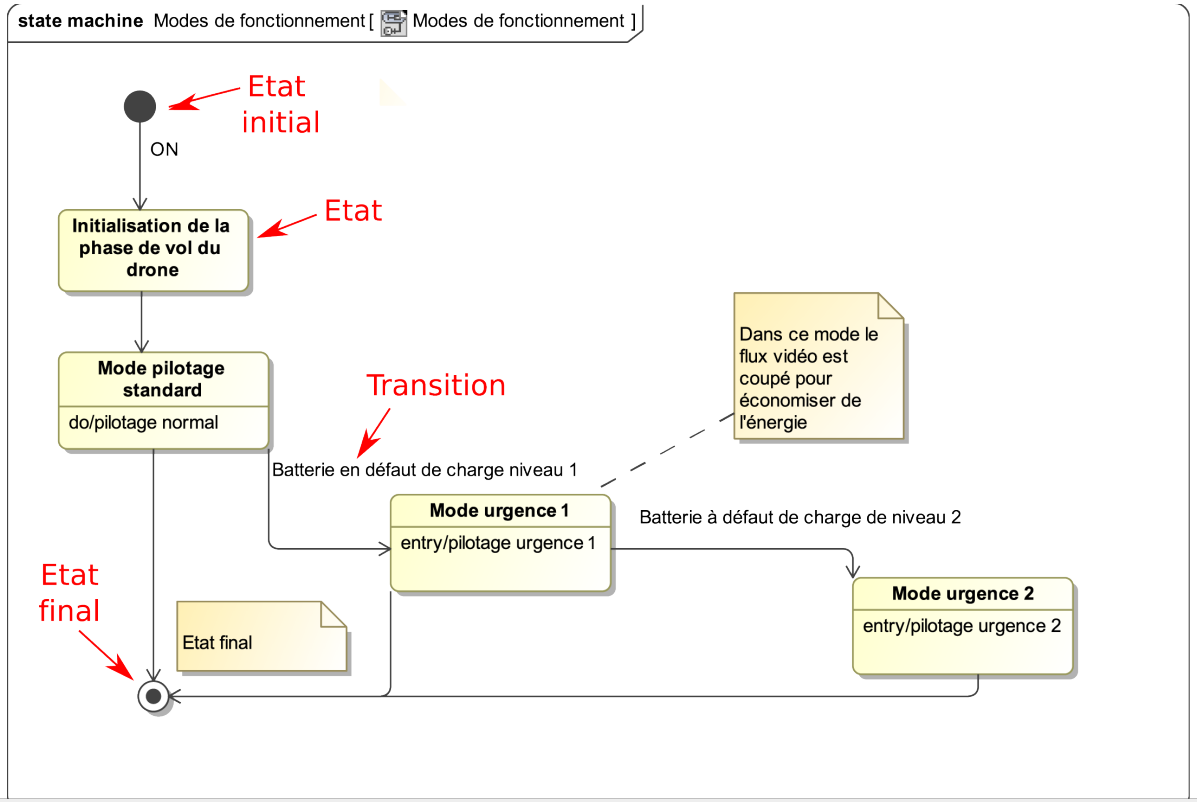
# Diagramme d'états



# Diagramme d'états



# Diagramme d'états



# Diagramme d'états

## Diagramme d'état

Le **diagramme d'états** est un diagramme comportemental appelé *State Machine Diagram* (**stm**) dans le langage SysML. Il est **rattaché à un bloc** (système, sous-système ou composant). Il permet de modéliser les différents états pris par le bloc en fonction d'évènements. Les éléments graphiques utilisés dans ce diagramme sont principalement :

- des rectangles aux coins arrondis pour les états,
- des flèches orientées de l'état de départ à l'état cible pour les transitions,
- des occurrences attachées à la transition spécifient les conditions de franchissement.

## Remarque

Chaque bloc ne conduit pas forcément à un diagramme d'états car cela n'est pas toujours possible.

# Diagramme d'états

## Diagramme d'état

Le **diagramme d'états** est un diagramme comportemental appelé *State Machine Diagram* (**stm**) dans le langage SysML. Il est **rattaché à un bloc** (système, sous-système ou composant). Il permet de modéliser les différents états pris par le bloc en fonction d'évènements. Les éléments graphiques utilisés dans ce diagramme sont principalement :

- des rectangles aux coins arrondis pour les états,
- des flèches orientées de l'état de départ à l'état cible pour les transitions,
- des occurrences attachées à la transition spécifient les conditions de franchissement.

## Remarque

Chaque bloc ne conduit pas forcément à un diagramme d'états car cela n'est pas toujours possible.

# Diagramme d'états

## Diagramme d'état

Le **diagramme d'états** est un diagramme comportemental appelé *State Machine Diagram* (**stm**) dans le langage SysML. Il est **rattaché à un bloc** (système, sous-système ou composant). Il permet de modéliser les différents états pris par le bloc en fonction d'évènements. Les éléments graphiques utilisés dans ce diagramme sont principalement :

- des rectangles aux coins arrondis pour les états,
- des flèches orientées de l'état de départ à l'état cible pour les transitions,
- des occurrences attachées à la transition spécifient les conditions de franchissement.

## Remarque

Chaque bloc ne conduit pas forcément à un diagramme d'états car cela n'est pas toujours possible.

# Diagramme d'états

## Diagramme d'état

Le **diagramme d'états** est un diagramme comportemental appelé *State Machine Diagram* (**stm**) dans le langage SysML. Il est **rattaché à un bloc** (système, sous-système ou composant). Il permet de modéliser les différents états pris par le bloc en fonction d'évènements. Les éléments graphiques utilisés dans ce diagramme sont principalement :

- des rectangles aux coins arrondis pour les états,
- des flèches orientées de l'état de départ à l'état cible pour les transitions,
- des occurrences attachées à la transition spécifient les conditions de franchissement.

## Remarque

Chaque bloc ne conduit pas forcément à un diagramme d'états car cela n'est pas toujours possible.

# Plan

# Diagramme d'états : états

## États

Un **état** représente une situation d'une durée finie durant laquelle un système ou sous-système (on parlera alors d'**objet**) :

- exécute une activité,
- satisfait à une certaine condition,
- ou bien est en attente d'un évènement.

## Remarque : règle de syntaxe

Un état peut être partitionné en plusieurs compartiments séparés par une ligne horizontale. Le premier compartiment contient le nom de l'état et les autres peuvent recevoir des transitions internes, ou des sous-états quand il s'agit d'un état composite. Dans le cas d'un état simple, on peut omettre toute barre de séparation.

# Diagramme d'états : états

## États

Un **état** représente une situation d'une durée finie durant laquelle un système ou sous-système (on parlera alors d'**objet**) :

- exécute une activité,
- satisfait à une certaine condition,
- ou bien est en attente d'un évènement.

## Remarque : règle de syntaxe

Un état peut être partitionné en plusieurs compartiments séparés par une ligne horizontale. Le premier compartiment contient le nom de l'état et les autres peuvent recevoir des transitions internes, ou des sous-états quand il s'agit d'un état composite. Dans le cas d'un état simple, on peut omettre toute barre de séparation.

# Diagramme d'états : états

## États

Un **état** représente une situation d'une durée finie durant laquelle un système ou sous-système (on parlera alors d'**objet**) :

- exécute une activité,
- satisfait à une certaine condition,
- ou bien est en attente d'un évènement.

## Remarque : règle de syntaxe

Un état peut être partitionné en plusieurs compartiments séparés par une ligne horizontale. Le premier compartiment contient le nom de l'état et les autres peuvent recevoir des transitions internes, ou des sous-états quand il s'agit d'un état composite. Dans le cas d'un état simple, on peut omettre toute barre de séparation.

# Diagramme d'états : états

## États

Un **état** représente une situation d'une durée finie durant laquelle un système ou sous-système (on parlera alors d'**objet**) :

- exécute une activité,
- satisfait à une certaine condition,
- ou bien est en attente d'un évènement.

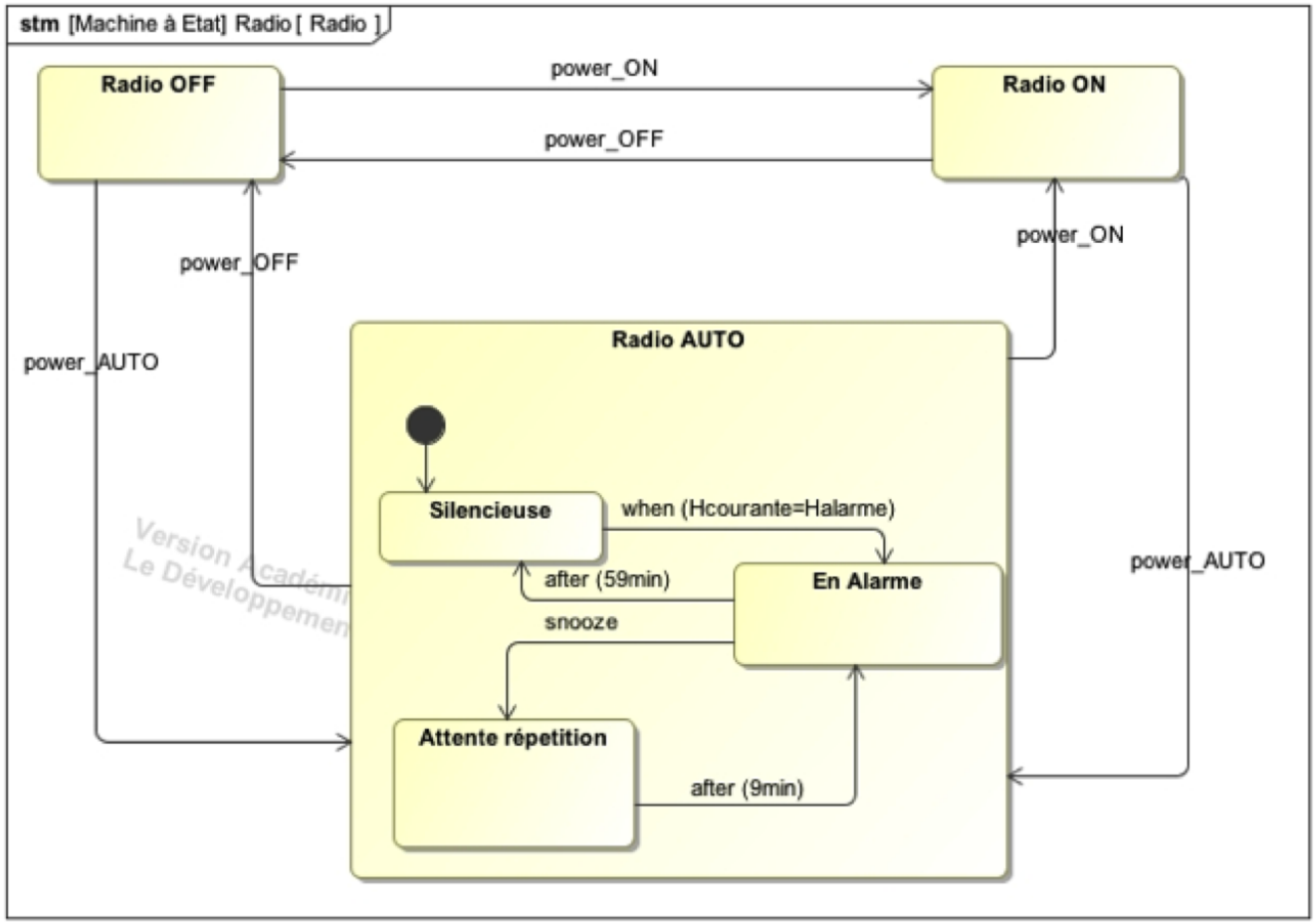
## Remarque : règle de syntaxe

Un état peut être partitionné en plusieurs compartiments séparés par une ligne horizontale. Le premier compartiment contient le nom de l'état et les autres peuvent recevoir des transitions internes, ou des sous-états quand il s'agit d'un état composite. Dans le cas d'un état simple, on peut omettre toute barre de séparation.

# Diagramme d'états : états composites

## État composite

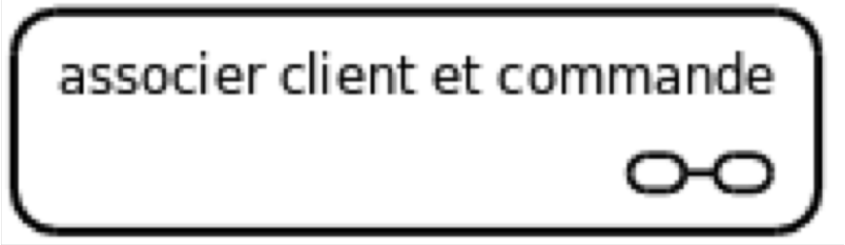
Un **état composite** peut contenir ou envelopper des sous-états.



# Diagramme d'états : états composites

**Remarque**

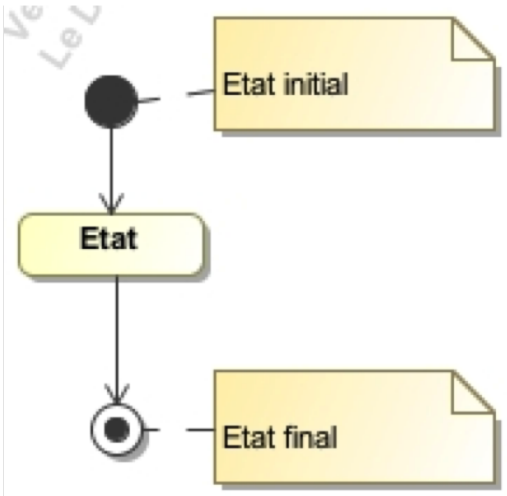
On peut représenter en abrégé un état composite. Ceci permet d'indiquer qu'un état est composite et que sa définition est donnée sur un autre diagramme.



# État initial, état final

## État initial, état final

- Un **état initial** est un pseudo état qui indique l'état de départ par défaut, lorsque le diagramme d'états ou l'état enveloppant est invoqué.
- Un **état final** est un pseudo état qui indique que le diagramme d'états-transitions, ou l'état enveloppant est terminé.

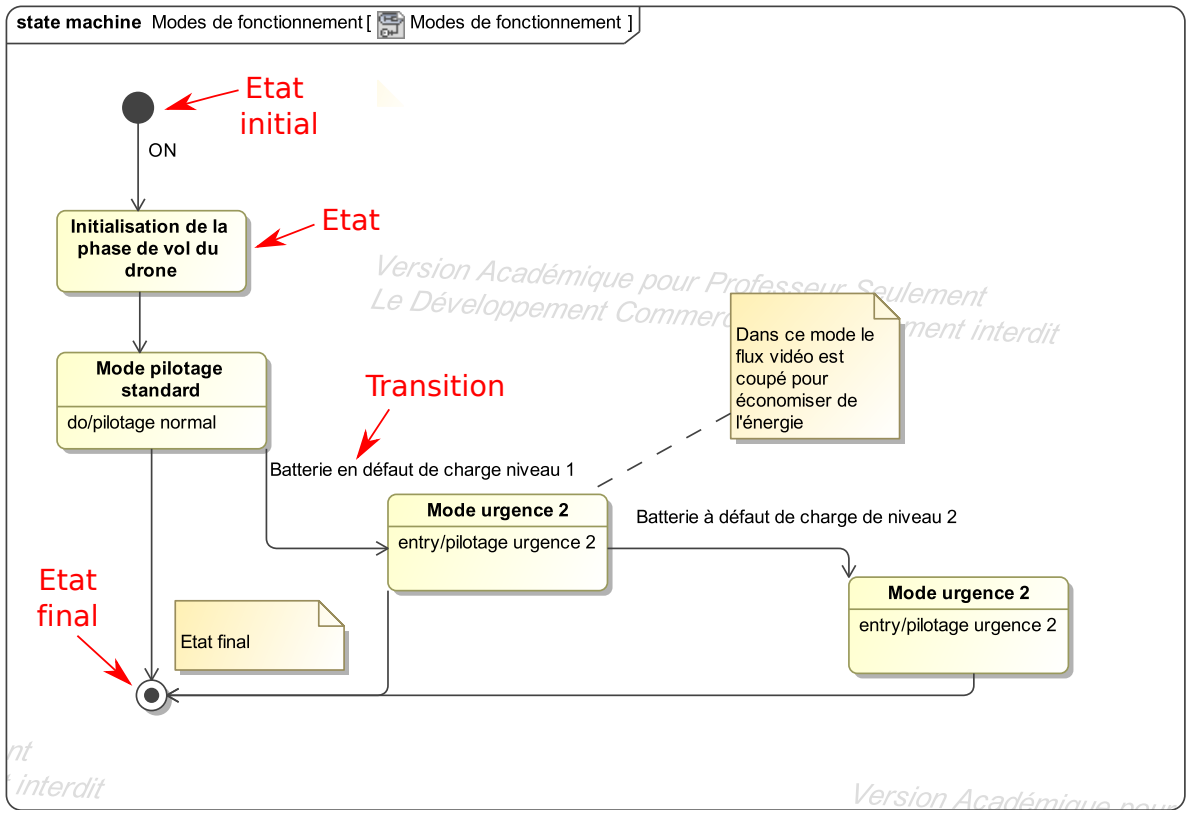


# Plan

# Transition

## Définition

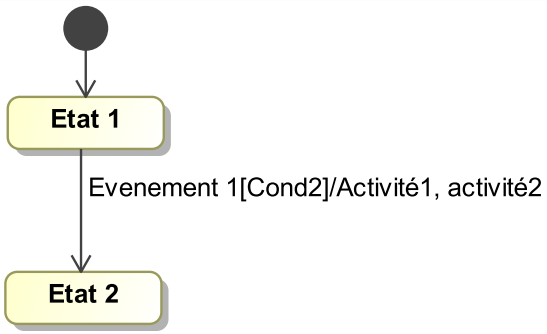
Une **transition** représente le passage d'un état à un autre.



# Transition externe

## Transition externe

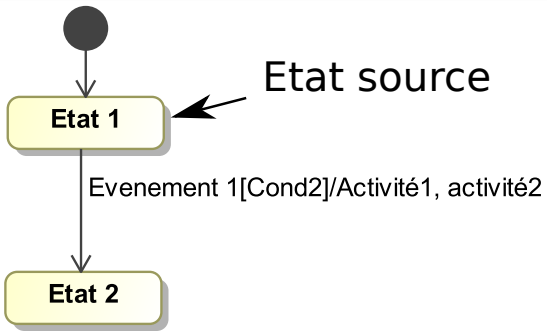
- Une **transition externe** décrit la réaction d'un objet lorsqu'un évènement se produit. En règle général une transition possède :
  - un état source : *Etat 1*,
  - un état cible : *Etat 2*,
  - un évènement déclencheur : *Evenement 1*,
  - une condition de garde : *Cond2*,
  - un effet : *activité 1* puis *activité 2* ;
- **Syntaxe :** évènement [garde] /effet.
- Elle est représentée par une ligne terminée par une flèche. Elle relie un état source à un état cible. Une transition ne peut être franchie que si son état source est actif.



# Transition externe

## Transition externe

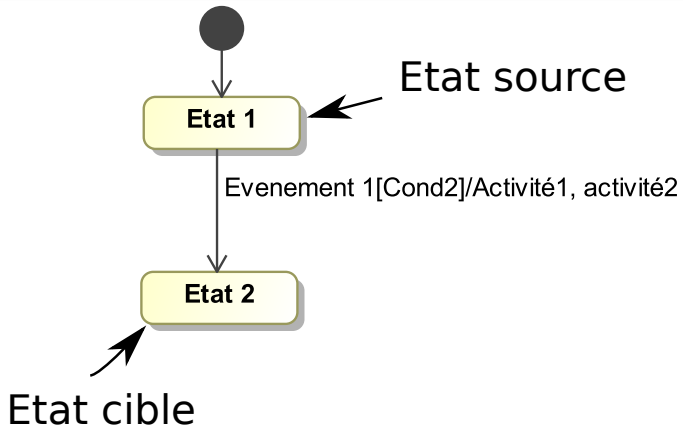
- Une **transition externe** décrit la réaction d'un objet lorsqu'un évènement se produit. En règle général une transition possède :
  - un état source : *Etat 1*,
  - un état cible : *Etat 2*,
  - un évènement déclencheur : *Evenement 1*,
  - une condition de garde : *Cond2*,
  - un effet : *activité 1* puis *activité 2* ;
- **Syntaxe :** évènement [garde] /effet.
- Elle est représentée par une ligne terminée par une flèche. Elle relie un état source à un état cible. Une transition ne peut être franchie que si son état source est actif.



# Transition externe

## Transition externe

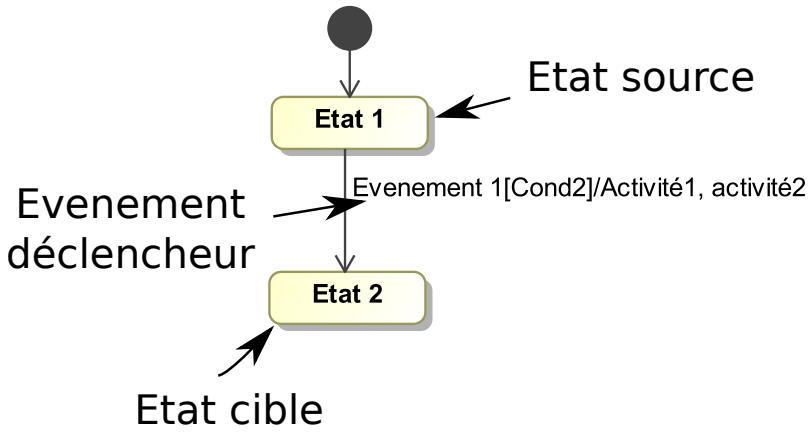
- Une **transition externe** décrit la réaction d'un objet lorsqu'un évènement se produit. En règle général une transition possède :
  - un état source : *Etat 1*,
  - un état cible : *Etat 2*,
  - un évènement déclencheur : *Evenement 1*,
  - une condition de garde : *Cond2*,
  - un effet : *activité 1* puis *activité 2* ;
- **Syntaxe :** évènement [garde] /effet.
- Elle est représentée par une ligne terminée par une flèche. Elle relie un état source à un état cible. Une transition ne peut être franchie que si son état source est actif.



# Transition externe

## Transition externe

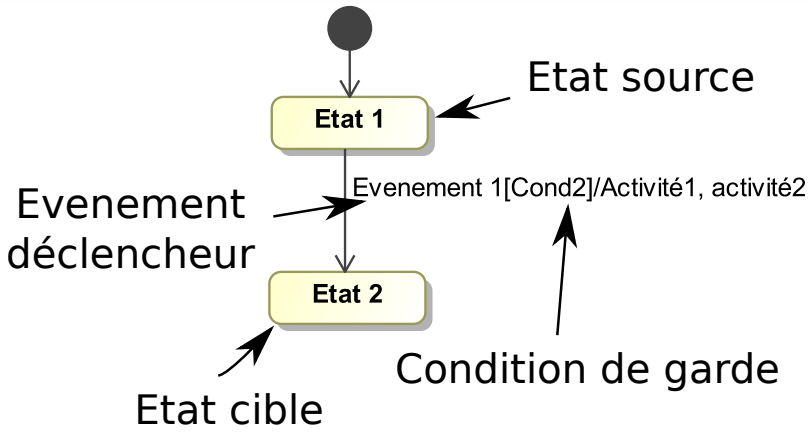
- Une **transition externe** décrit la réaction d'un objet lorsqu'un évènement se produit. En règle général une transition possède :
  - un état source : *Etat 1*,
  - un état cible : *Etat 2*,
  - un évènement déclencheur : *Evenement 1*,
  - une condition de garde : *Cond2*,
  - un effet : *activité 1* puis *activité 2* ;
- **Syntaxe :** évènement [garde] /effet.
- Elle est représentée par une ligne terminée par une flèche. Elle relie un état source à un état cible. Une transition ne peut être franchie que si son état source est actif.



# Transition externe

## Transition externe

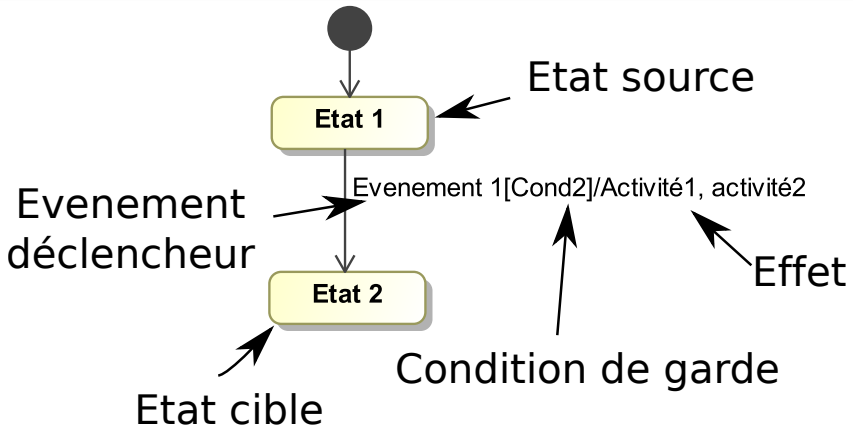
- Une **transition externe** décrit la réaction d'un objet lorsqu'un évènement se produit. En règle général une transition possède :
  - un état source : *Etat 1*,
  - un état cible : *Etat 2*,
  - un évènement déclencheur : *Evenement 1*,
  - une condition de garde : *Cond2*,
  - un effet : *activité 1* puis *activité 2* ;
- **Syntaxe :** évènement [garde] /effet.
- Elle est représentée par une ligne terminée par une flèche. Elle relie un état source à un état cible. Une transition ne peut être franchie que si son état source est actif.



# Transition externe

## Transition externe

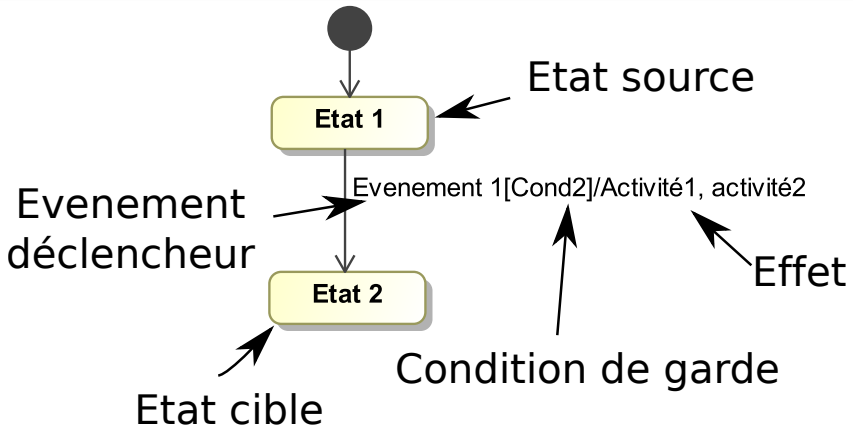
- Une **transition externe** décrit la réaction d'un objet lorsqu'un évènement se produit. En règle général une transition possède :
  - un état source : *Etat 1*,
  - un état cible : *Etat 2*,
  - un évènement déclencheur : *Evenement 1*,
  - une condition de garde : *Cond2*,
  - un effet : *activité 1* puis *activité 2* ;
- **Syntaxe :** évènement [garde] /effet.
- Elle est représentée par une ligne terminée par une flèche. Elle relie un état source à un état cible. Une transition ne peut être franchie que si son état source est actif.



# Transition externe

## Transition externe

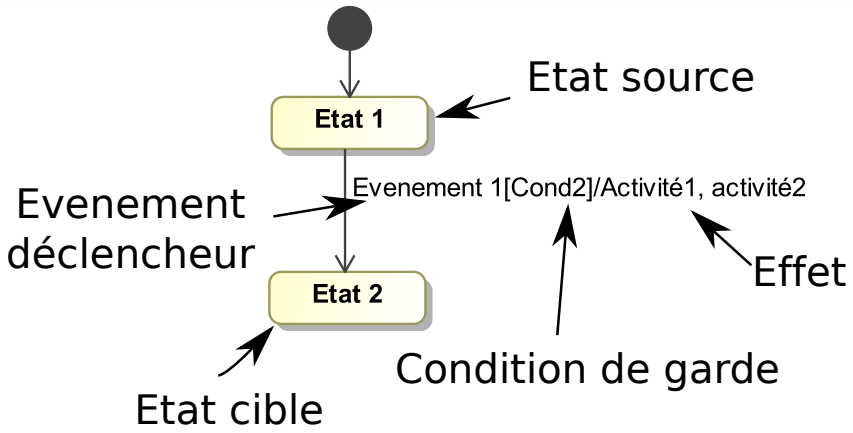
- Une **transition externe** décrit la réaction d'un objet lorsqu'un évènement se produit. En règle général une transition possède :
  - un état source : *Etat 1*,
  - un état cible : *Etat 2*,
  - un évènement déclencheur : *Evenement 1*,
  - une condition de garde : *Cond2*,
  - un effet : *activité 1* puis *activité 2* ;
- **Syntaxe :** évènement [garde] /effet.
- Elle est représentée par une ligne terminée par une flèche. Elle relie un état source à un état cible. Une transition ne peut être franchie que si son état source est actif.



# Transition externe

## Transition externe

- Une **transition externe** décrit la réaction d'un objet lorsqu'un évènement se produit. En règle général une transition possède :
  - un état source : *Etat 1*,
  - un état cible : *Etat 2*,
  - un évènement déclencheur : *Evenement 1*,
  - une condition de garde : *Cond2*,
  - un effet : *activité 1* puis *activité 2* ;
- **Syntaxe :** évènement [garde] /effet.
- Elle est représentée par une ligne terminée par une flèche. Elle relie un état source à un état cible. Une transition ne peut être franchie que si son état source est actif.

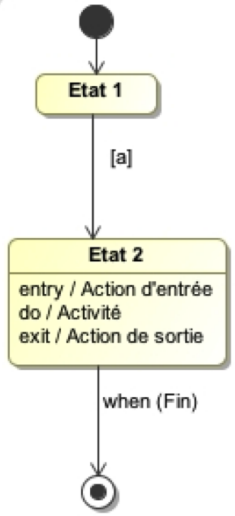


# Transition interne

## Transition interne

Une transition interne est notée à l'intérieur d'un état. Elle est précédée de l'un des mots clé suivants :

- **entry/action d'entrée** : entraine la réalisation d'une action à l'entrée de l'état,
- **do/Activité** : entraine la réalisation d'une activité lorsque l'état est actif,
- **On évènement/activité** : entraine la réalisation d'une activité lorsqu'un un évènement se réalise,
- **exit/Action de sortie** : entraine la réalisation d'une action à la sortie de l'état.

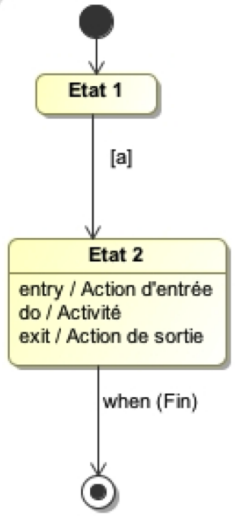


# Transition interne

## Transition interne

Une transition interne est notée à l'intérieur d'un état. Elle est précédée de l'un des mots clé suivants :

- **entry/action d'entrée** : entraine la réalisation d'une action à l'entrée de l'état,
- **do/Activité** : entraine la réalisation d'une activité lorsque l'état est actif,
- **On évènement/activité** : entraine la réalisation d'une activité lorsqu'un un évènement se réalise,
- **exit/Action de sortie** : entraine la réalisation d'une action à la sortie de l'état.

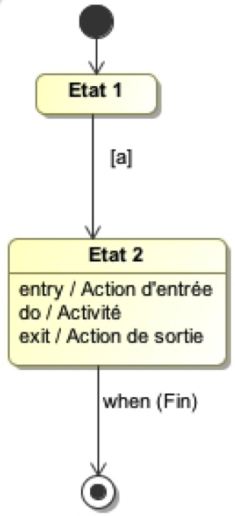


# Transition interne

## Transition interne

Une transition interne est notée à l'intérieur d'un état. Elle est précédée de l'un des mots clé suivants :

- **entry/action d'entrée** : entraine la réalisation d'une action à l'entrée de l'état,
- **do/Activité** : entraine la réalisation d'une activité lorsque l'état est actif,
- **On évènement/activité** : entraine la réalisation d'une activité lorsqu'un un évènement se réalise,
- **exit/Action de sortie** : entraine la réalisation d'une action à la sortie de l'état.

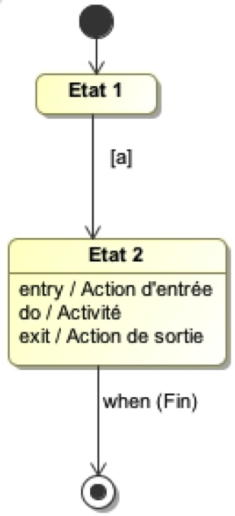


# Transition interne

## Transition interne

Une transition interne est notée à l'intérieur d'un état. Elle est précédée de l'un des mots clé suivants :

- **entry/action d'entrée** : entraine la réalisation d'une action à l'entrée de l'état,
- **do/Activité** : entraine la réalisation d'une activité lorsque l'état est actif,
- **On évènement/activité** : entraine la réalisation d'une activité lorsqu'un un évènement se réalise,
- **exit/Action de sortie** : entraine la réalisation d'une action à la sortie de l'état.

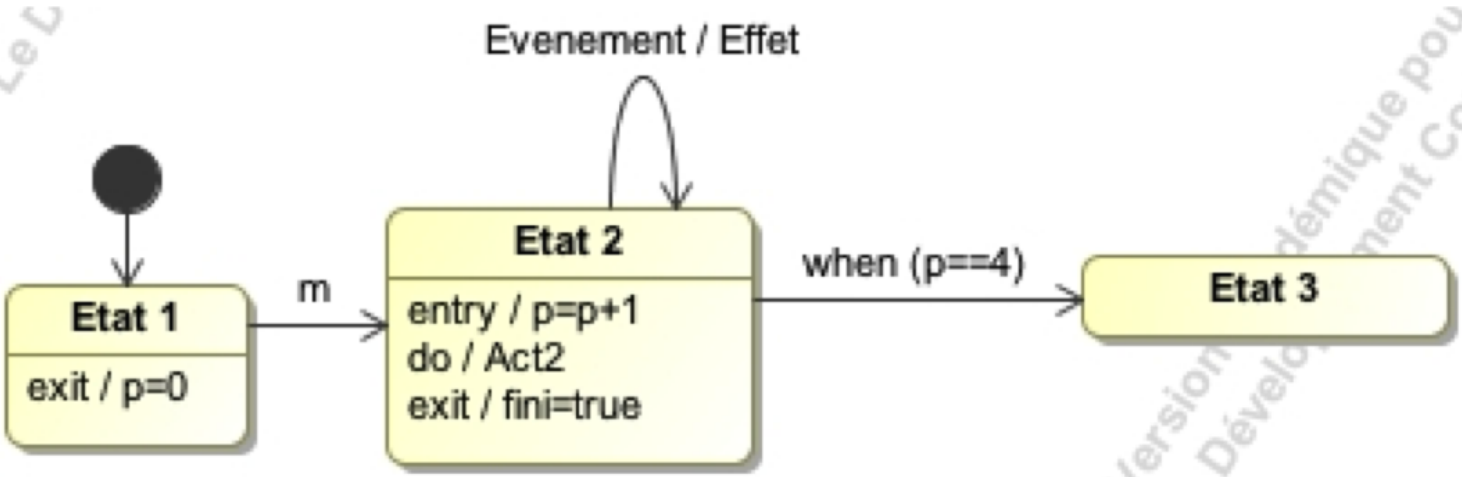


# Transition propre

## Transition propre

Une transition propre concerne un état propre. Elle relie donc un état à lui-même.

- Dans l'exemple ci-dessous, lorsque **Etat2** est actif, si **Evenement** se produit, alors **Effet** (activité de courte durée) est déclenchée puis **Etat2** est de nouveau actif.
- Chaque fois que **Evenement** se produit, la réactivation de **Etat2** fait que **p** est incrémenté.
- Le fait de sortir et d'entrer dans **Etat2** peut avoir des effets néfastes comme l'interruption puis le redémarrage de **Act2**.

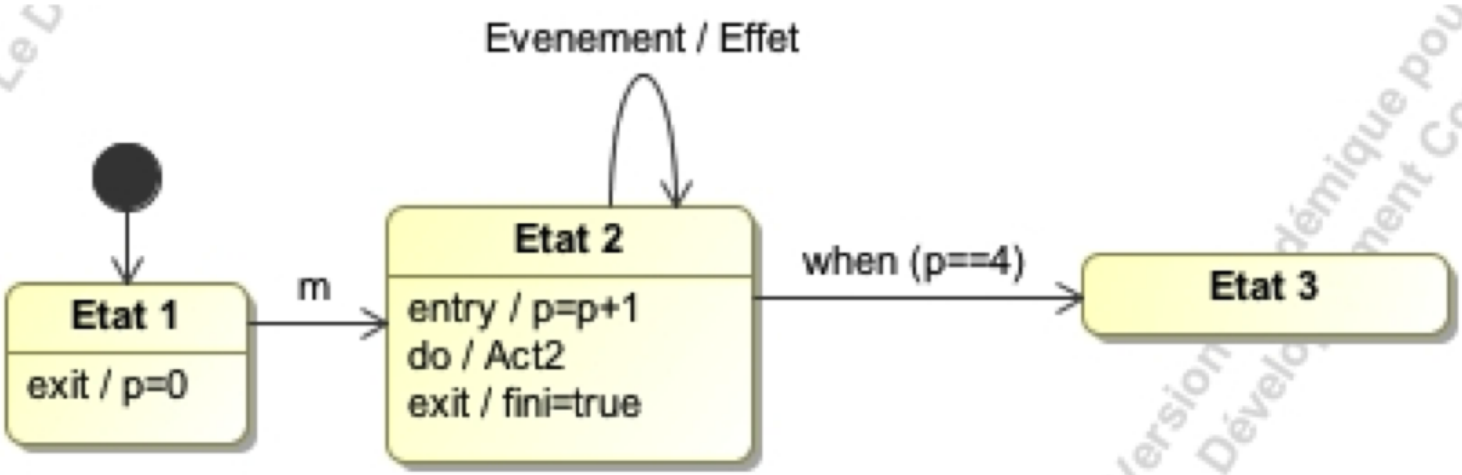


# Transition propre

## Transition propre

Une transition propre concerne un état propre. Elle relie donc un état à lui-même.

- Dans l'exemple ci-dessous, lorsque **Etat2** est actif, si **Evenement** se produit, alors **Effet** (activité de courte durée) est déclenchée puis **Etat2** est de nouveau actif.
- Chaque fois que **Evenement** se produit, la réactivation de **Etat2** fait que **p** est incrémenté.
- Le fait de sortir et d'entrer dans **Etat2** peut avoir des effets néfastes comme l'interruption puis le redémarrage de **Act2**.

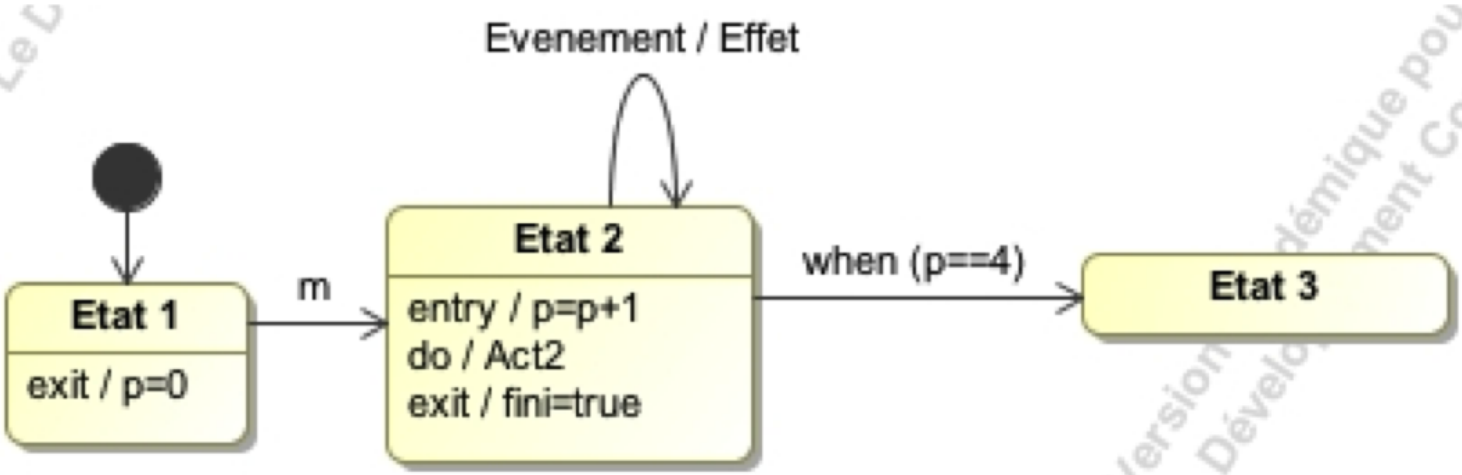


# Transition propre

## Transition propre

Une transition propre concerne un état propre. Elle relie donc un état à lui-même.

- Dans l'exemple ci-dessous, lorsque **Etat2** est actif, si **Evenement** se produit, alors **Effet** (activité de courte durée) est déclenchée puis **Etat2** est de nouveau actif.
- Chaque fois que **Evenement** se produit, la réactivation de **Etat2** fait que **p** est incrémenté.
- Le fait de sortir et d'entrer dans **Etat2** peut avoir des effets néfastes comme l'interruption puis le redémarrage de **Act2**.



# Événement

## Évènement

Un **événement** se produit pendant l'exécution d'un système.

- Un événement se produit à un instant précis et est dépourvu de durée.
- Quand un évènement est reçu, une transition peut être déclenchée et faire basculer l'objet dans un nouvel état.
- On peut diviser les événements en plusieurs types explicites et implicites : signal, appel, changement et temporel.

# Événement

## Types d'évènements

- Un **événement de type signal** : appui sur un bouton.
- Un **événement de type changement** : **when** (< **condition_booléenne** >) Un évènement de changement est évalué continuellement jusqu'à ce qu'il devienne vrai et c'est à ce moment là que la transition se déclenche.
- Les **événement temporels** sont générés par le passage du temps.
  - Un événement temporel spécifié de manière relative : **after** (< **durée** >),
  - Un événement temporel spécifié de manière absolue : **when** (< **date** >) ou **at** (< **date** >) dans le logiciel MagicDraw.

# Événement

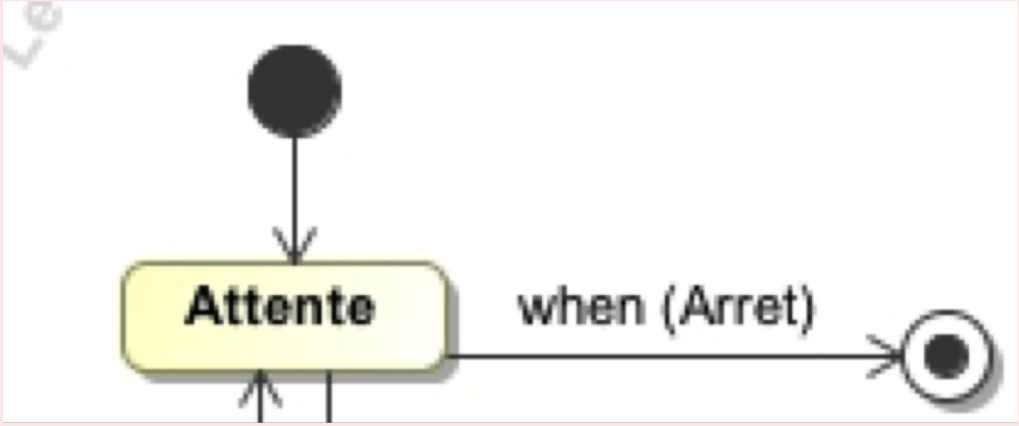
## Types d'évènements

- Un **événement de type signal** : appui sur un bouton.
- Un **événement de type changement** : **when** (< **condition_booléenne** >) Un évènement de changement est évalué continuellement jusqu'à ce qu'il devienne vrai et c'est à ce moment là que la transition se déclenche.
- Les **événement temporels** sont générés par le passage du temps.
  - Un événement temporel spécifié de manière relative : **after** (< **durée** >),
  - Un événement temporel spécifié de manière absolue : **when** (< **date** >) ou **at** (< **date** >) dans le logiciel MagicDraw.

# Événement

## Types d'évènements

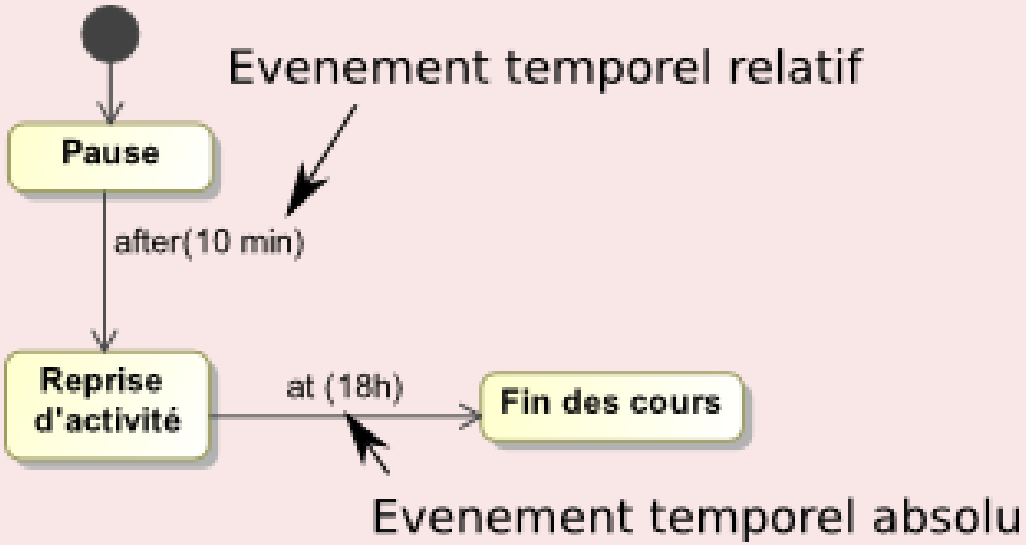
- Un **événement de type signal** : appui sur un bouton.
- Un **événement de type changement** : **when** (< **condition_booléenne** >) Un évènement de changement est évalué continuellement jusqu'à ce qu'il devienne vrai et c'est à ce moment là que la transition se déclenche.
- Les **événement temporels** sont générés par le passage du temps.
  - Un événement temporel spécifié de manière relative : **after** (< **durée** >),
  - Un événement temporel spécifié de manière absolue : **when** (< **date** >) ou **at** (< **date** >) dans le logiciel MagicDraw.



# Événement

## Types d'évènements

- Un **événement de type signal** : appui sur un bouton.
- Un **événement de type changement** : **when** (< **condition_booléenne** >) Un évènement de changement est évalué continuellement jusqu'à ce qu'il devienne vrai et c'est à ce moment là que la transition se déclenche.
- Les **événement temporels** sont générés par le passage du temps.
  - Un événement temporel spécifié de manière relative : **after** (< **durée** >),
  - Un événement temporel spécifié de manière absolue : **when** (< **date** >) ou **at** (< **date** >) dans le logiciel MagicDraw.

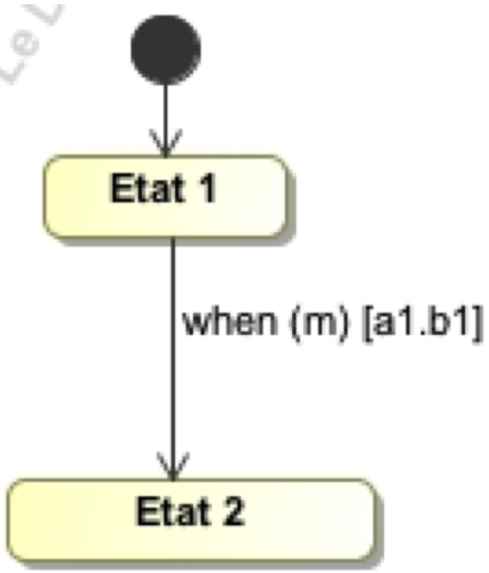


# Condition de garde

## Condition de garde

Une **condition de garde** est représentée entre crochet : [*garde*]
Il s'agit d'une expression logique sur les attributs de l'objet associés au diagramme d'états ainsi que sur les paramètres de l'événement déclencheur.

- La condition est évaluée uniquement lorsque l'événement déclencheur se produit.
- Si l'expression est fausse à ce moment là, la transition ne se déclenche pas.
- Si elle est vrai, la transition se déclenche et ses effets se produisent.

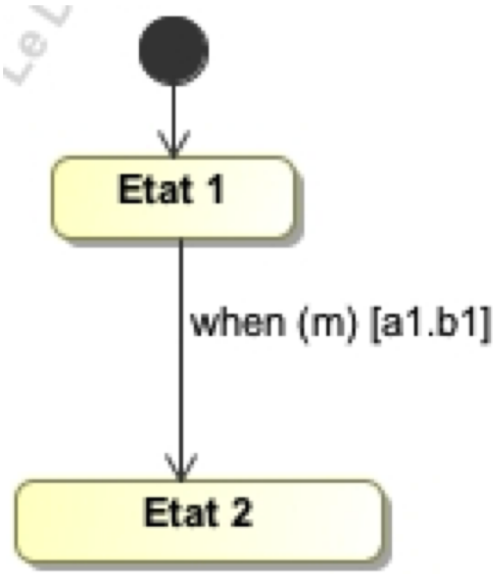


# Condition de garde

## Condition de garde

Une **condition de garde** est représentée entre crochet : [*garde*]
Il s'agit d'une expression logique sur les attributs de l'objet associés au diagramme d'états ainsi que sur les paramètres de l'événement déclencheur.

- La condition est évaluée uniquement lorsque l'événement déclencheur se produit.
- Si l'expression est fausse à ce moment là, la transition ne se déclenche pas.
- Si elle est vrai, la transition se déclenche et ses effets se produisent.

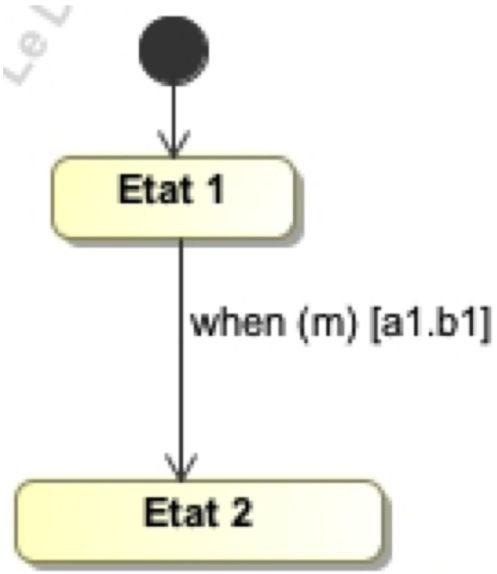


# Condition de garde

## Condition de garde

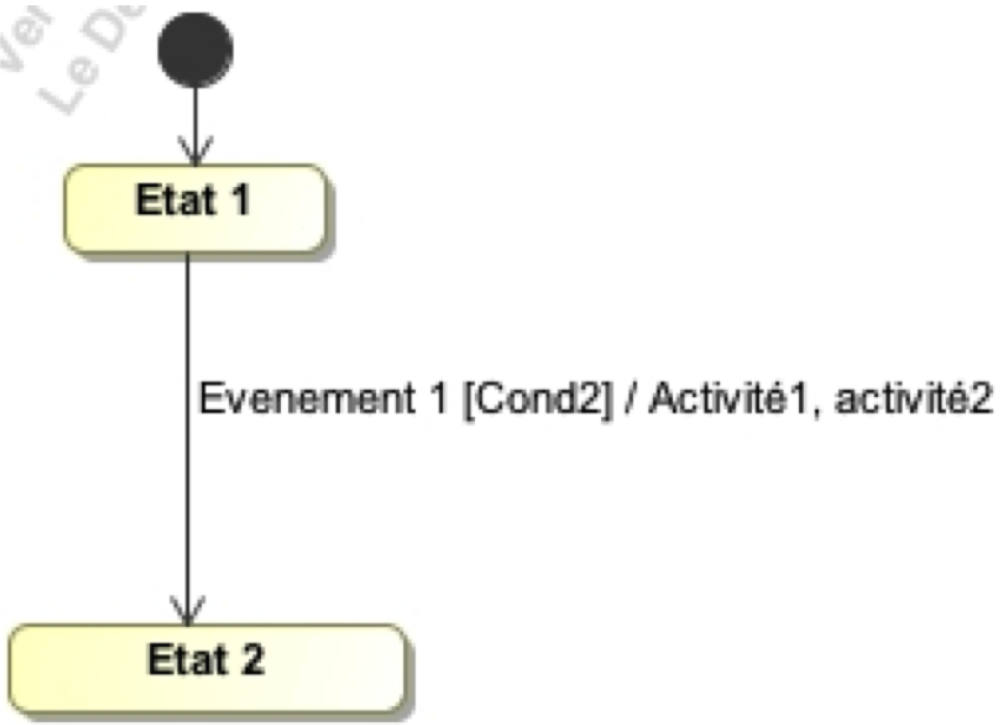
Une **condition de garde** est représentée entre crochet : [*garde*]
Il s'agit d'une expression logique sur les attributs de l'objet associés au diagramme d'états ainsi que sur les paramètres de l'événement déclencheur.

- La condition est évaluée uniquement lorsque l'événement déclencheur se produit.
- Si l'expression est fausse à ce moment là, la transition ne se déclenche pas.
- Si elle est vrai, la transition se déclenche et ses effets se produisent.



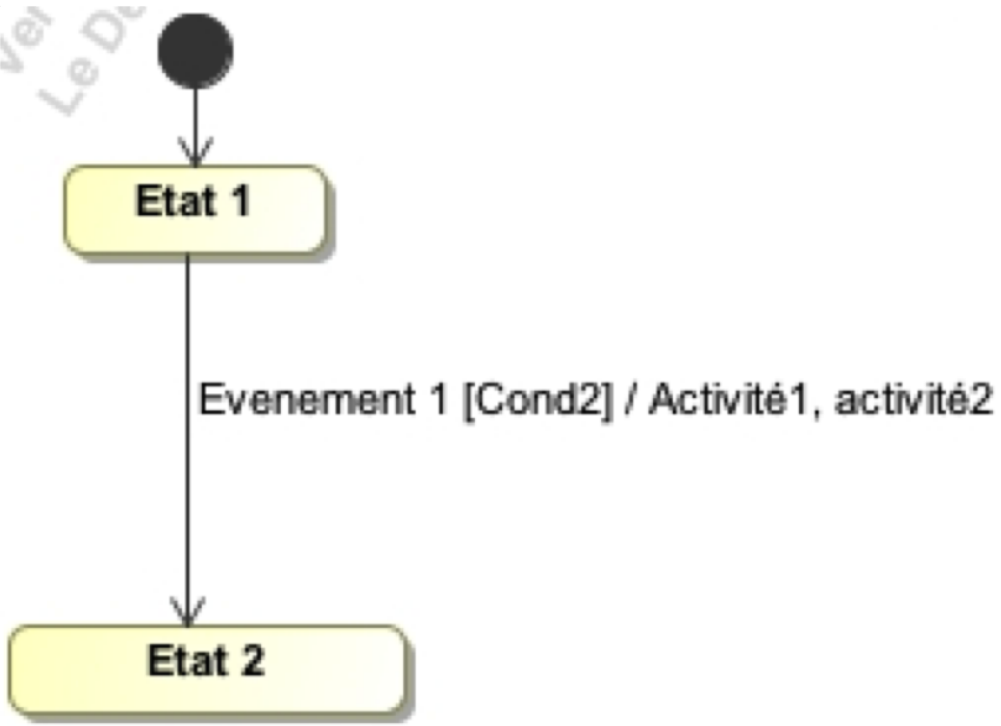
# Effet d'une transition

- Lorsqu'une transition se déclenche, son effet spécifié par "/**activité**" dans la syntaxe, s'exécute. Il s'agit généralement d'une activité qui peut être :
  - une opération primitive comme une instruction d'assignation ;
  - l'envoi d'un signal ;
  - l'appel d'une opération ;
  - une liste d'activités, etc.
- La façon de spécifier l'activité à réaliser est laissée libre (langage naturel ou pseudo-code). Lorsque l'exécution de l'effet est terminée, l'état cible de la transition devient actif.



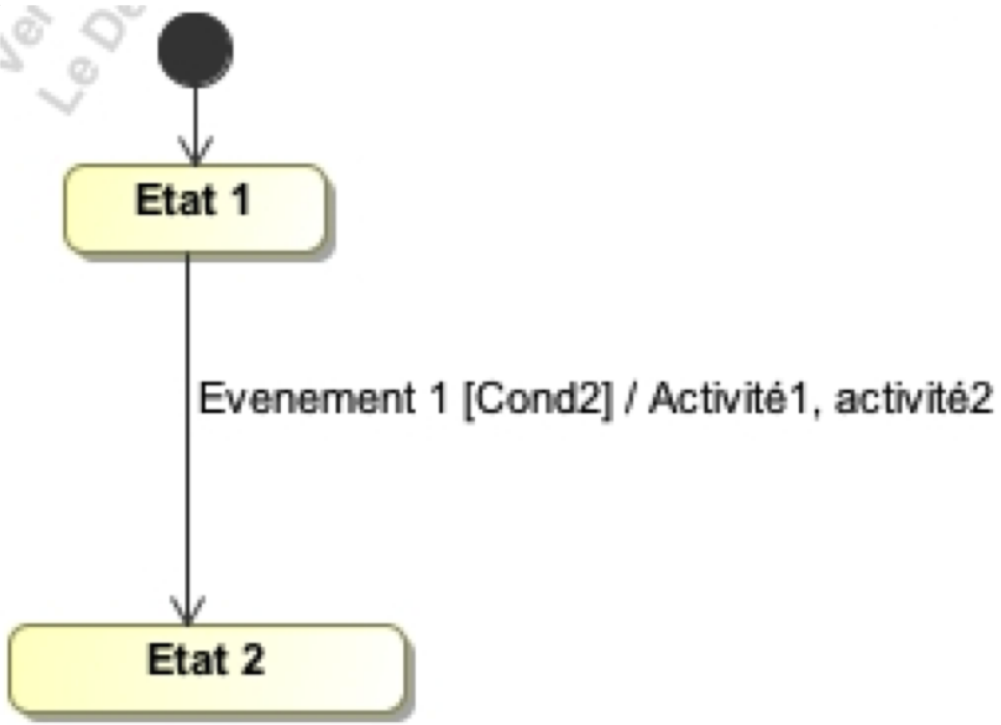
# Effet d'une transition

- Lorsqu'une transition se déclenche, son effet spécifié par "/**activité**" dans la syntaxe, s'exécute. Il s'agit généralement d'une activité qui peut être :
  - une opération primitive comme une instruction d'assignation ;
  - l'envoi d'un signal ;
  - l'appel d'une opération ;
  - une liste d'activités, etc.
- La façon de spécifier l'activité à réaliser est laissée libre (langage naturel ou pseudo-code). Lorsque l'exécution de l'effet est terminée, l'état cible de la transition devient actif.



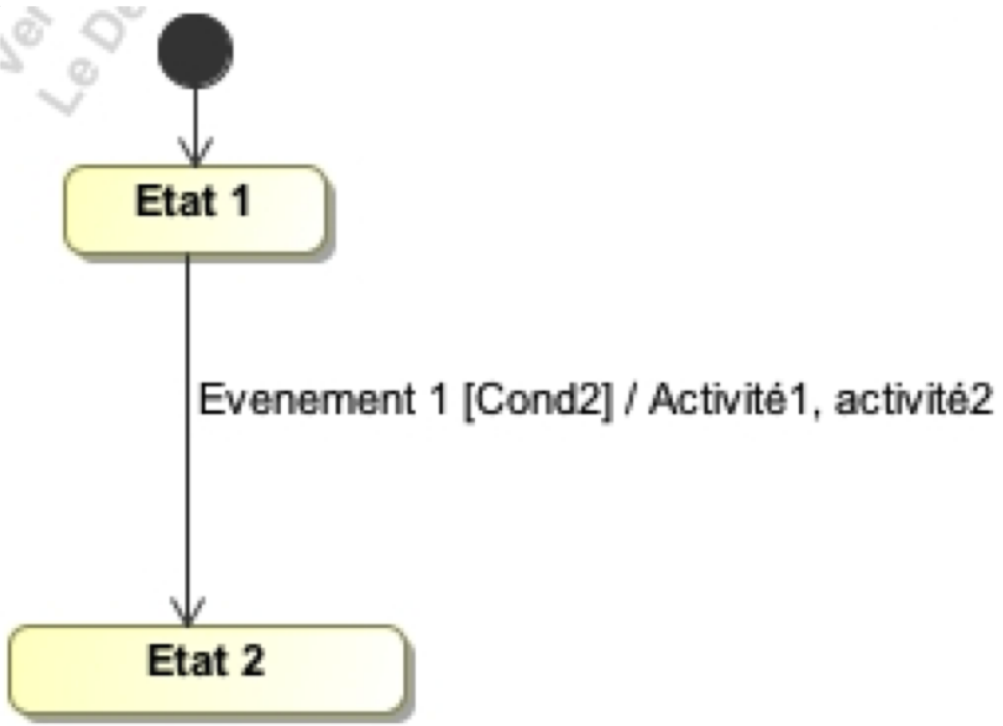
# Effet d'une transition

- Lorsqu'une transition se déclenche, son effet spécifié par "/**activité**" dans la syntaxe, s'exécute. Il s'agit généralement d'une activité qui peut être :
  - une opération primitive comme une instruction d'assignation ;
  - l'envoi d'un signal ;
  - l'appel d'une opération ;
  - une liste d'activités, etc.
- La façon de spécifier l'activité à réaliser est laissée libre (langage naturel ou pseudo-code). Lorsque l'exécution de l'effet est terminée, l'état cible de la transition devient actif.



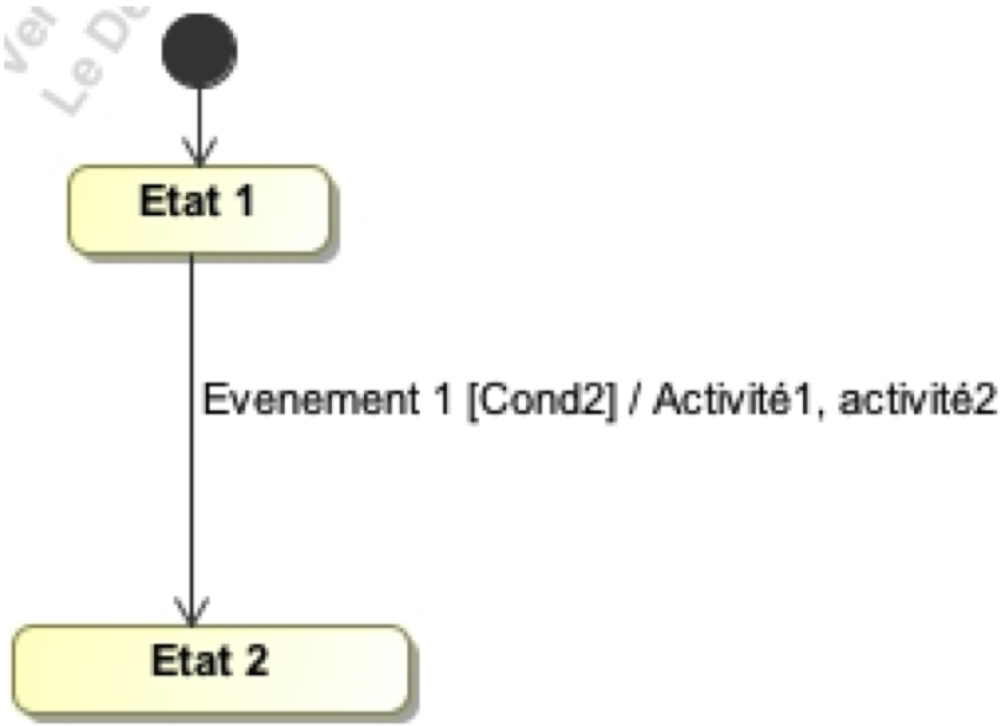
# Effet d'une transition

- Lorsqu'une transition se déclenche, son effet spécifié par "/**activité**" dans la syntaxe, s'exécute. Il s'agit généralement d'une activité qui peut être :
  - une opération primitive comme une instruction d'assignation ;
  - l'envoi d'un signal ;
  - l'appel d'une opération ;
  - une liste d'activités, etc.
- La façon de spécifier l'activité à réaliser est laissée libre (langage naturel ou pseudo-code). Lorsque l'exécution de l'effet est terminée, l'état cible de la transition devient actif.



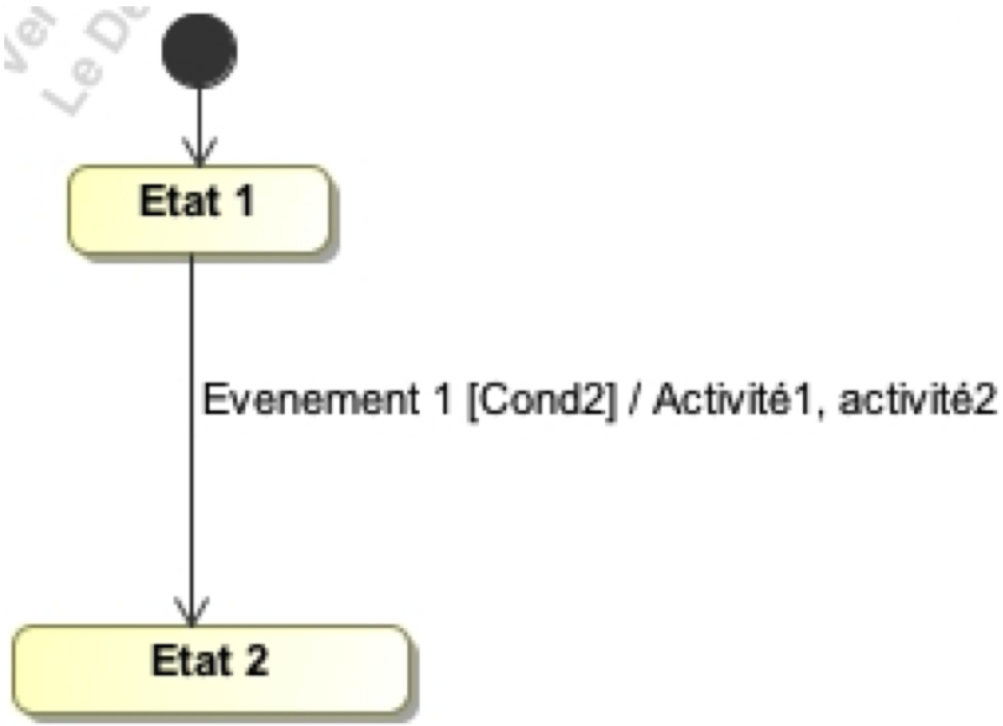
# Effet d'une transition

- Lorsqu'une transition se déclenche, son effet spécifié par "/**activité**" dans la syntaxe, s'exécute. Il s'agit généralement d'une activité qui peut être :
  - une opération primitive comme une instruction d'assignation ;
  - l'envoi d'un signal ;
  - l'appel d'une opération ;
  - une liste d'activités, etc.
- La façon de spécifier l'activité à réaliser est laissée libre (langage naturel ou pseudo-code). Lorsque l'exécution de l'effet est terminée, l'état cible de la transition devient actif.



# Effet d'une transition

- Lorsqu'une transition se déclenche, son effet spécifié par "/**activité**" dans la syntaxe, s'exécute. Il s'agit généralement d'une activité qui peut être :
  - une opération primitive comme une instruction d'assignation ;
  - l'envoi d'un signal ;
  - l'appel d'une opération ;
  - une liste d'activités, etc.
- La façon de spécifier l'activité à réaliser est laissée libre (langage naturel ou pseudo-code). Lorsque l'exécution de l'effet est terminée, l'état cible de la transition devient actif.

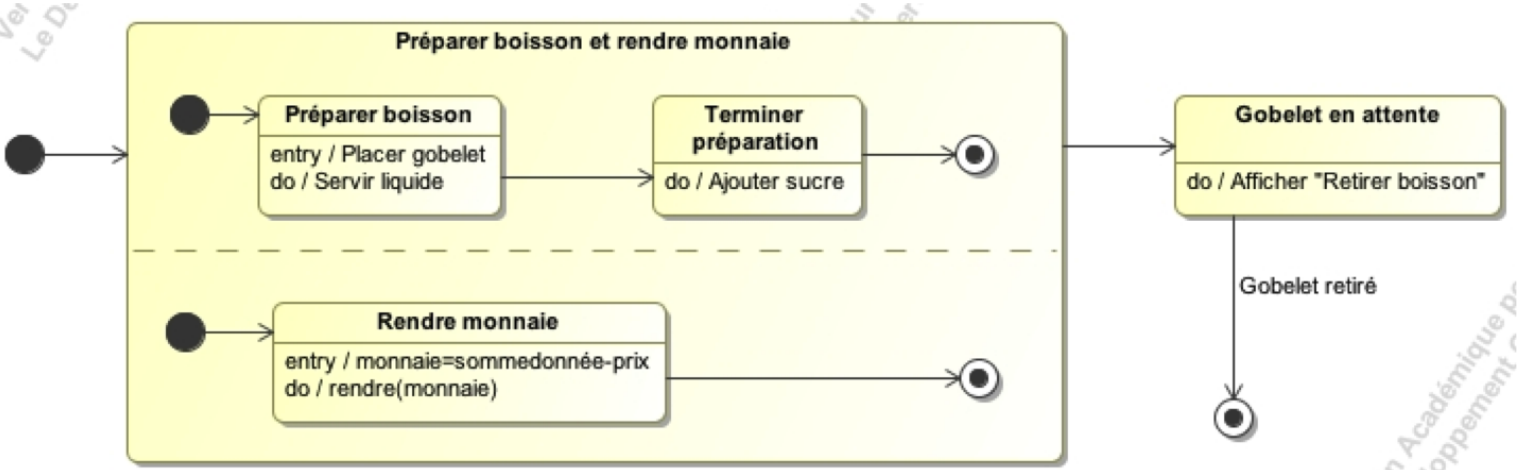


# Plan

# États orthogonaux

## États orthogonaux

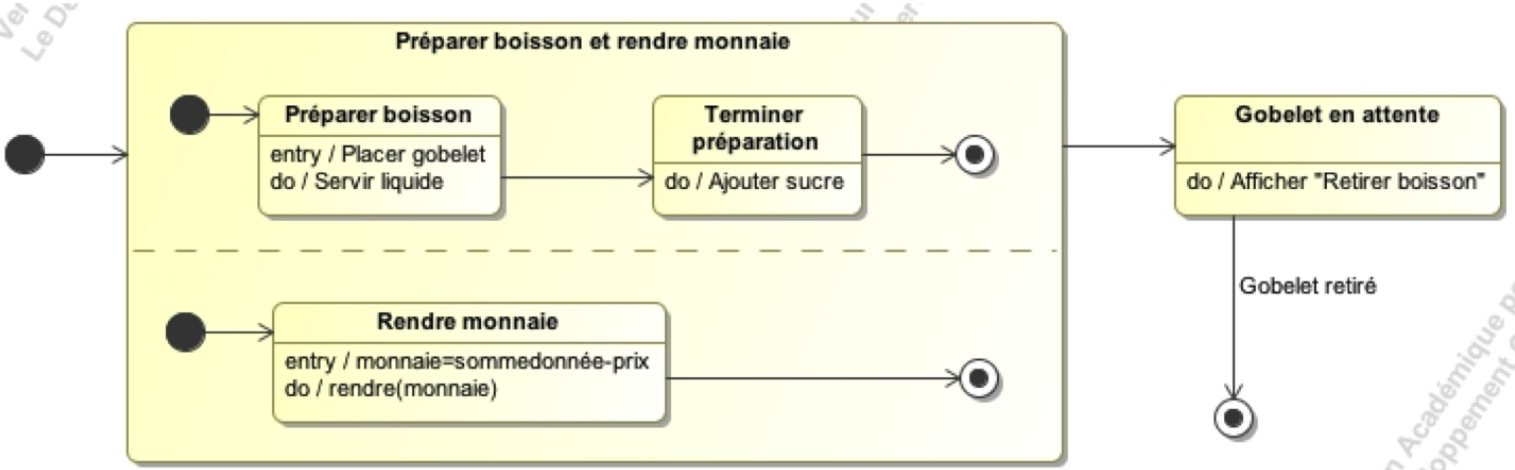
- Un **état orthogonal** est un état composite comportant plus d'une région représentant chacune d'elle un flot d'exécution. Ces états permettent de décrire efficacement les mécanismes concurrents.
- Chaque région peut posséder un état initial et final. Une transition qui atteint la bordure d'un état composite orthogonal est équivalente à une transition qui atteint les états initiaux de toutes ses régions concurrents.
- Toutes les régions concurrentes d'un état composite orthogonal doivent atteindre leur état final pour que l'état composite soit considéré comme terminé.



# États orthogonaux

## États orthogonaux

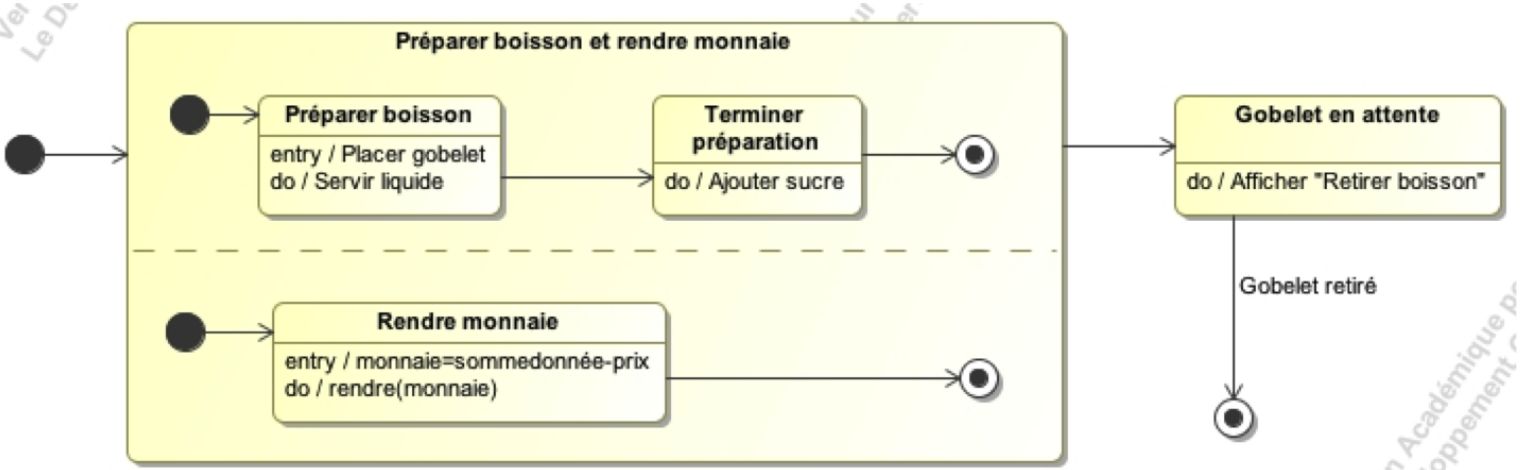
- Un **état orthogonal** est un état composite comportant plus d'une région représentant chacune d'elle un flot d'exécution. Ces états permettent de décrire efficacement les mécanismes concurrents.
- Chaque région peut posséder un état initial et final. Une transition qui atteint la bordure d'un état composite orthogonal est équivalente à une transition qui atteint les états initiaux de toutes ses régions concurrents.
- Toutes les régions concurrentes d'un état composite orthogonal doivent atteindre leur état final pour que l'état composite soit considéré comme terminé.



# États orthogonaux

## États orthogonaux

- Un **état orthogonal** est un état composite comportant plus d'une région représentant chacune d'elle un flot d'exécution. Ces états permettent de décrire efficacement les mécanismes concurrents.
- Chaque région peut posséder un état initial et final. Une transition qui atteint la bordure d'un état composite orthogonal est équivalente à une transition qui atteint les états initiaux de toutes ses régions concurrents.
- Toutes les régions concurrentes d'un état composite orthogonal doivent atteindre leur état final pour que l'état composite soit considéré comme terminé.

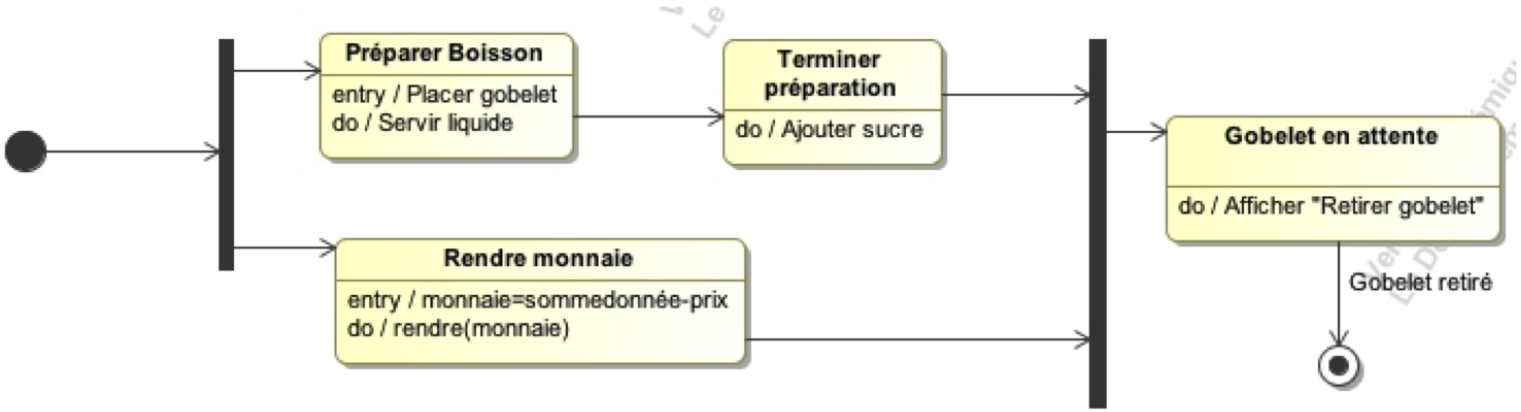


# Transitions complexes ou concurrentes

## Transitions complexes ou concurrentes

Les **transitions complexes ou concurrentes** sont représentées par une barre épaisse et peuvent éventuellement être nommées.

- Dès le franchissement de la bifurcation les états immédiatement suivants sont activés.
- L'union ne peut être franchies que lorsque toutes les activités des états immédiatement précédents sont terminées.

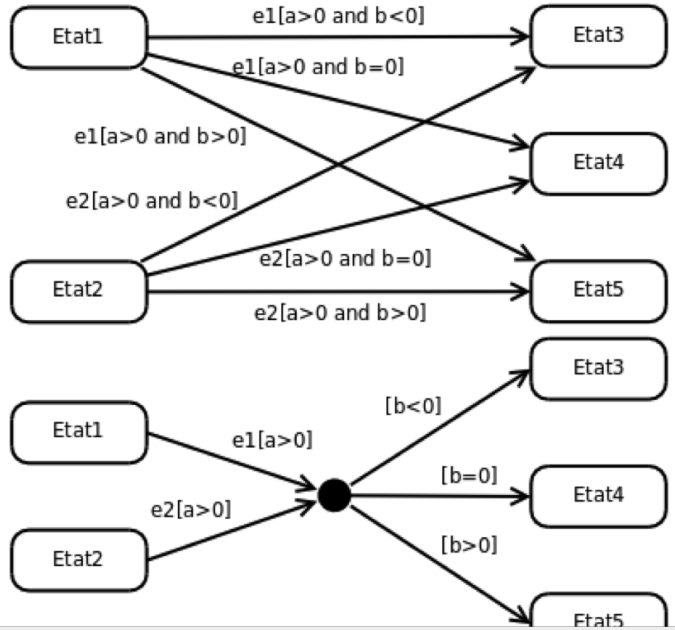


# Plan

# Point de jonction

## Point de jonction

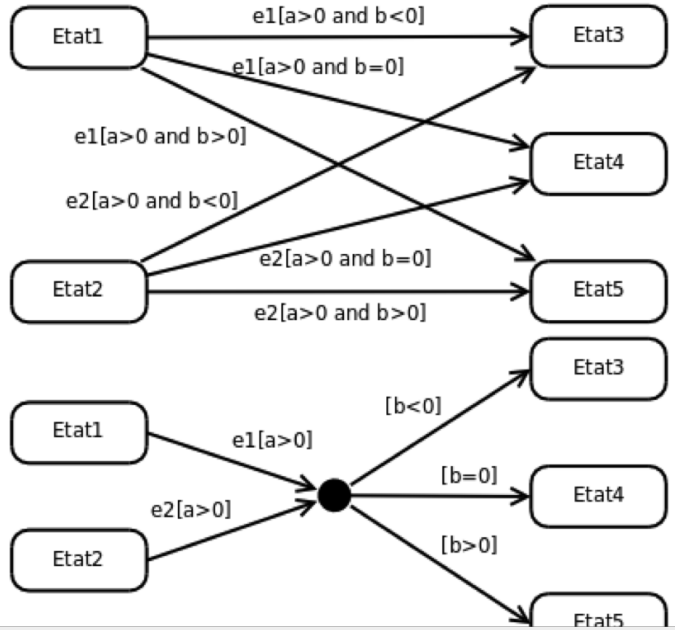
- Un **point de jonction** permet de partager des segments de transition.
- Un point de jonction peut avoir plusieurs segments de transition entrante et plusieurs segments de transition sortante. Par contre il ne peut avoir d'activité interne ni des transitions sortantes dotées de déclencheurs d'événements.
- Il ne s'agit pas d'un état qui peut être actif au cours d'un laps de temps fini.
- Toutes les gardes le long de ce chemin doivent s'évaluer à vrai dès le franchissement du premier segment.



# Point de jonction

## Point de jonction

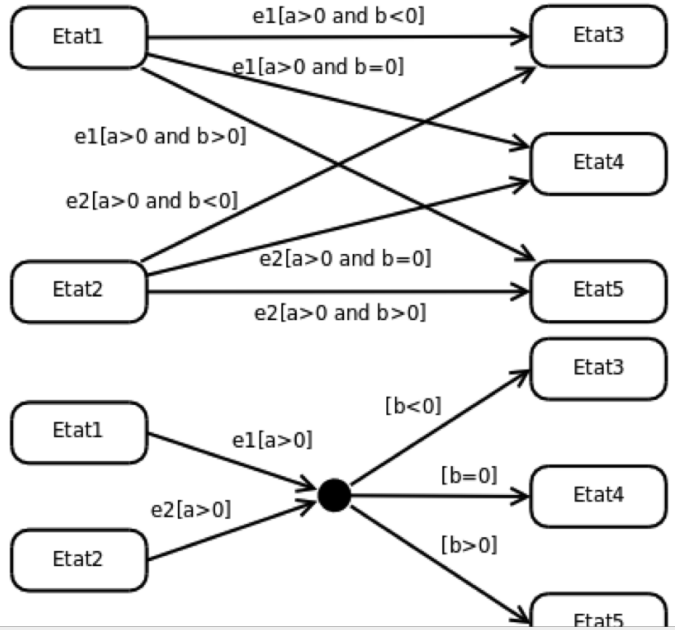
- Un **point de jonction** permet de partager des segments de transition.
- Un point de jonction peut avoir plusieurs segments de transition entrante et plusieurs segments de transition sortante. Par contre il ne peut avoir d'activité interne ni des transitions sortantes dotées de déclencheurs d'événements.
- Il ne s'agit pas d'un état qui peut être actif au cours d'un laps de temps fini.
- Toutes les gardes le long de ce chemin doivent s'évaluer à vrai dès le franchissement du premier segment.



# Point de jonction

## Point de jonction

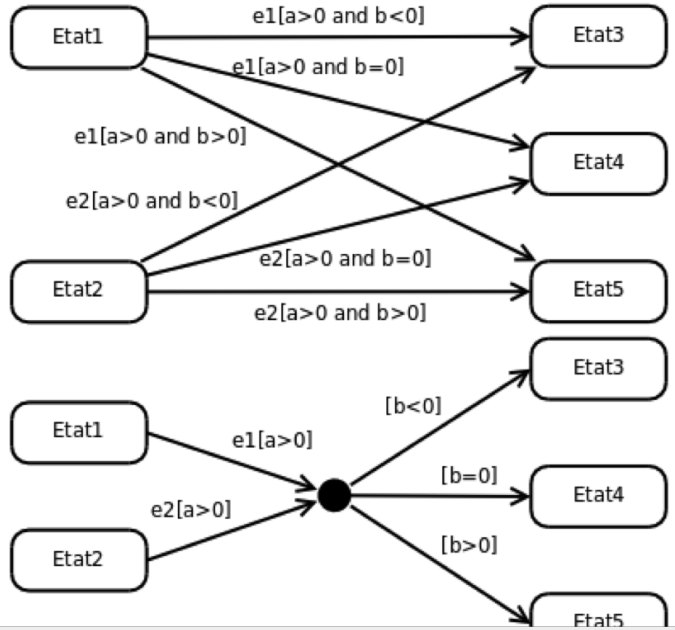
- Un **point de jonction** permet de partager des segments de transition.
- Un point de jonction peut avoir plusieurs segments de transition entrante et plusieurs segments de transition sortante. Par contre il ne peut avoir d'activité interne ni des transitions sortantes dotées de déclencheurs d'événements.
- Il ne s'agit pas d'un état qui peut être actif au cours d'un laps de temps fini.
- Toutes les gardes le long de ce chemin doivent s'évaluer à vrai dès le franchissement du premier segment.



# Point de jonction

## Point de jonction

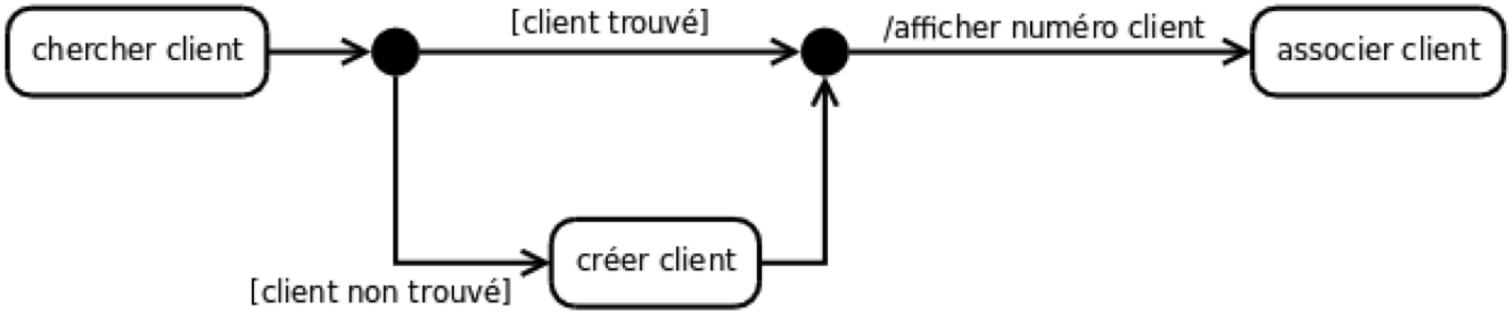
- Un **point de jonction** permet de partager des segments de transition.
- Un point de jonction peut avoir plusieurs segments de transition entrante et plusieurs segments de transition sortante. Par contre il ne peut avoir d'activité interne ni des transitions sortantes dotées de déclencheurs d'événements.
- Il ne s'agit pas d'un état qui peut être actif au cours d'un laps de temps fini.
- Toutes les gardes le long de ce chemin doivent s'évaluer à vrai dès le franchissement du premier segment.



# Point de jonction

## Point de jonction

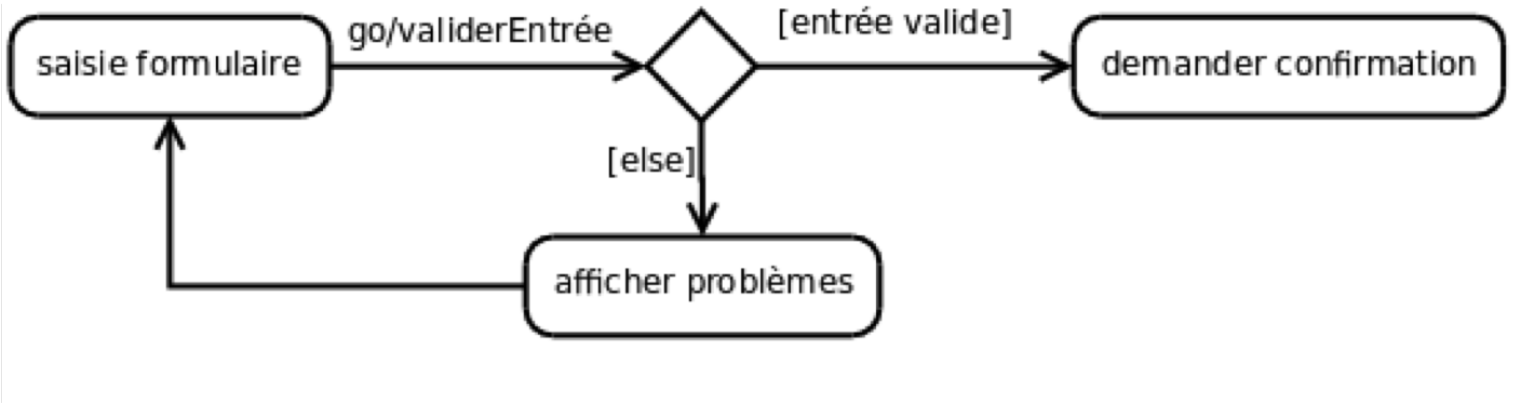
- Un **point de jonction** permet de partager des segments de transition.
- Un point de jonction peut avoir plusieurs segments de transition entrante et plusieurs segments de transition sortante. Par contre il ne peut avoir d'activité interne ni des transitions sortantes dotées de déclencheurs d'événements.
- Il ne s'agit pas d'un état qui peut être actif au cours d'un laps de temps fini.
- Toutes les gardes le long de ce chemin doivent s'évaluer à vrai dès le franchissement du premier segment.



# Point de décision

## Point de décision

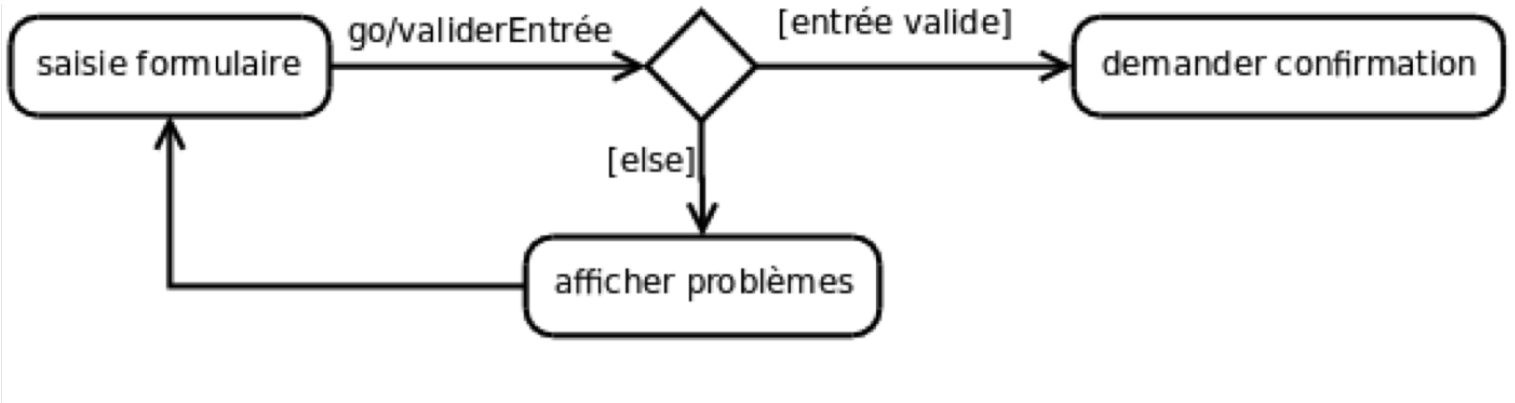
- Un **point de décision** possède une entrée et au moins deux sorties.
- Contrairement à un point de jonction, les gardes situées après le point de décision sont évaluées au moment où il est atteint.
- Cela permet de baser le choix sur des résultat obtenus en franchissant le segment avant le point de choix. Si, quand le point de décision est atteint, aucun segment en aval n'est franchissable, c'est que le modèle est mal formé.
- Condition exclusive avec la garde : **[else]**, sur un des segments en aval d'un point de choix. ce segment n'est franchissable que si les gardes des autres segments sont toutes fausses.



# Point de décision

## Point de décision

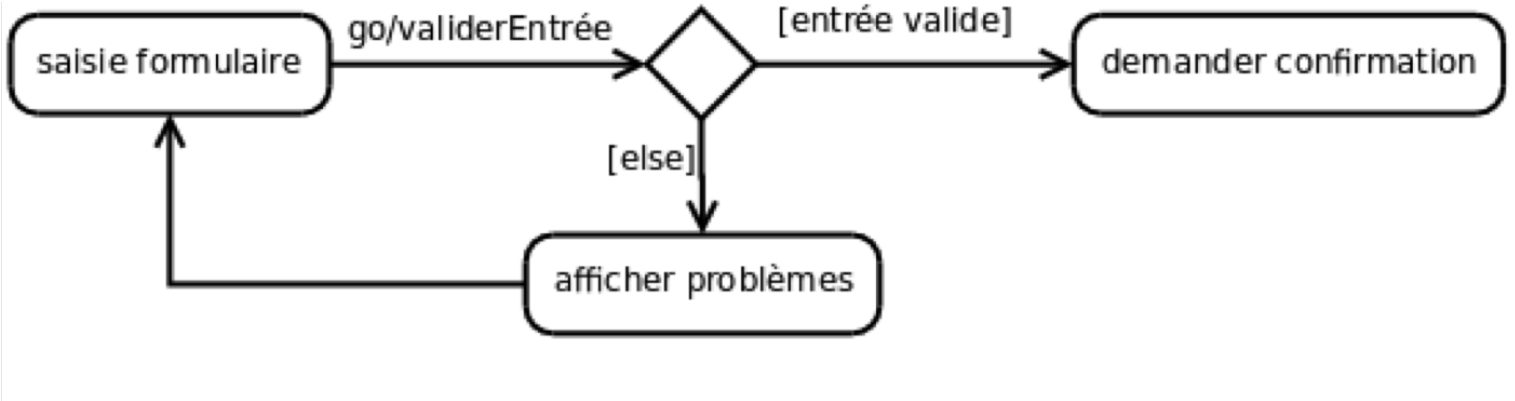
- Un **point de décision** possède une entrée et au moins deux sorties.
- Contrairement à un point de jonction, les gardes situées après le point de décision sont évaluées au moment où il est atteint.
- Cela permet de baser le choix sur des résultat obtenus en franchissant le segment avant le point de choix. Si, quand le point de décision est atteint, aucun segment en aval n'est franchissable, c'est que le modèle est mal formé.
- Condition exclusive avec la garde : **[else]**, sur un des segments en aval d'un point de choix. ce segment n'est franchissable que si les gardes des autres segments sont toutes fausses.



# Point de décision

## Point de décision

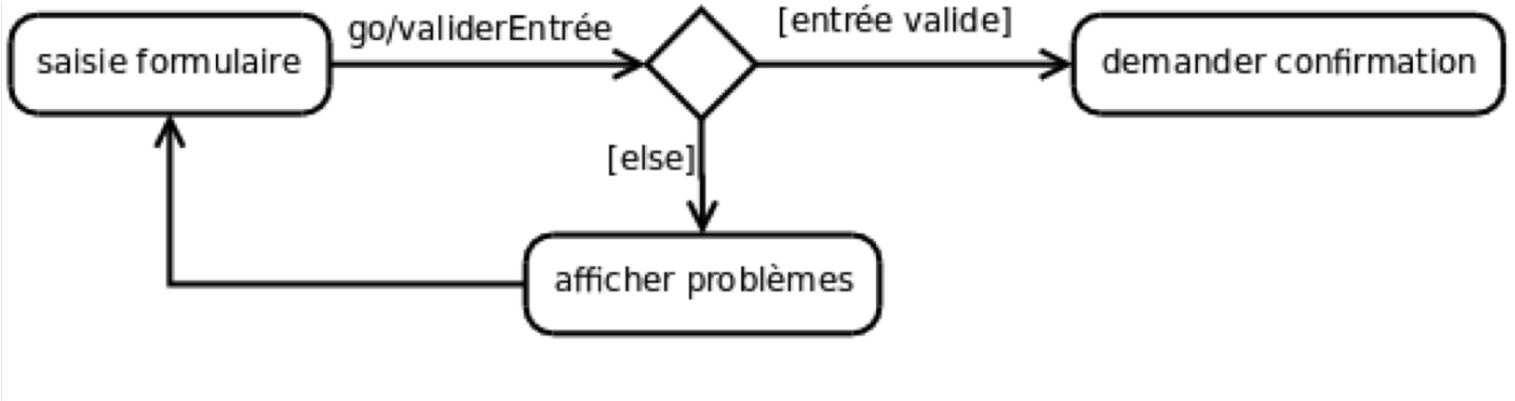
- Un **point de décision** possède une entrée et au moins deux sorties.
- Contrairement à un point de jonction, les gardes situées après le point de décision sont évaluées au moment où il est atteint.
- Cela permet de baser le choix sur des résultat obtenus en franchissant le segment avant le point de choix. Si, quand le point de décision est atteint, aucun segment en aval n'est franchissable, c'est que le modèle est mal formé.
- Condition exclusive avec la garde : **[else]**, sur un des segments en aval d'un point de choix. ce segment n'est franchissable que si les gardes des autres segments sont toutes fausses.



# Point de décision

## Point de décision

- Un **point de décision** possède une entrée et au moins deux sorties.
- Contrairement à un point de jonction, les gardes situées après le point de décision sont évaluées au moment où il est atteint.
- Cela permet de baser le choix sur des résultat obtenus en franchissant le segment avant le point de choix. Si, quand le point de décision est atteint, aucun segment en aval n'est franchissable, c'est que le modèle est mal formé.
- Condition exclusive avec la garde : **[else]**, sur un des segments en aval d'un point de choix. ce segment n'est franchissable que si les gardes des autres segments sont toutes fausses.



# Plan

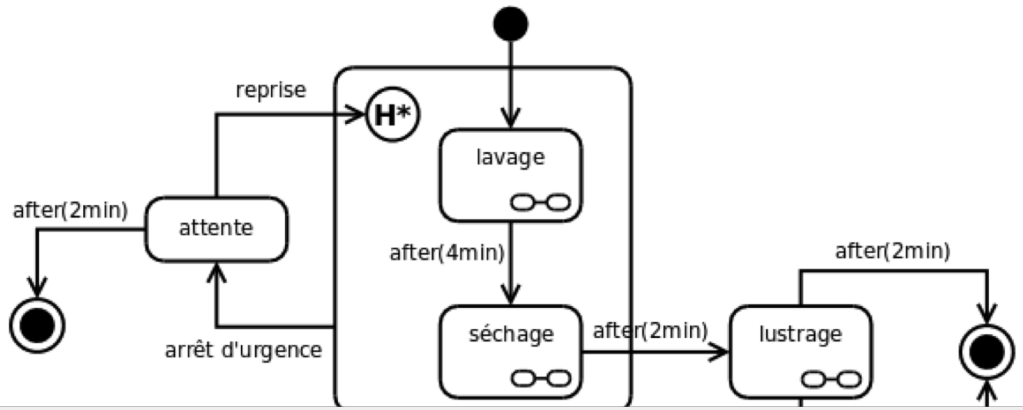
# État historique

## État historique

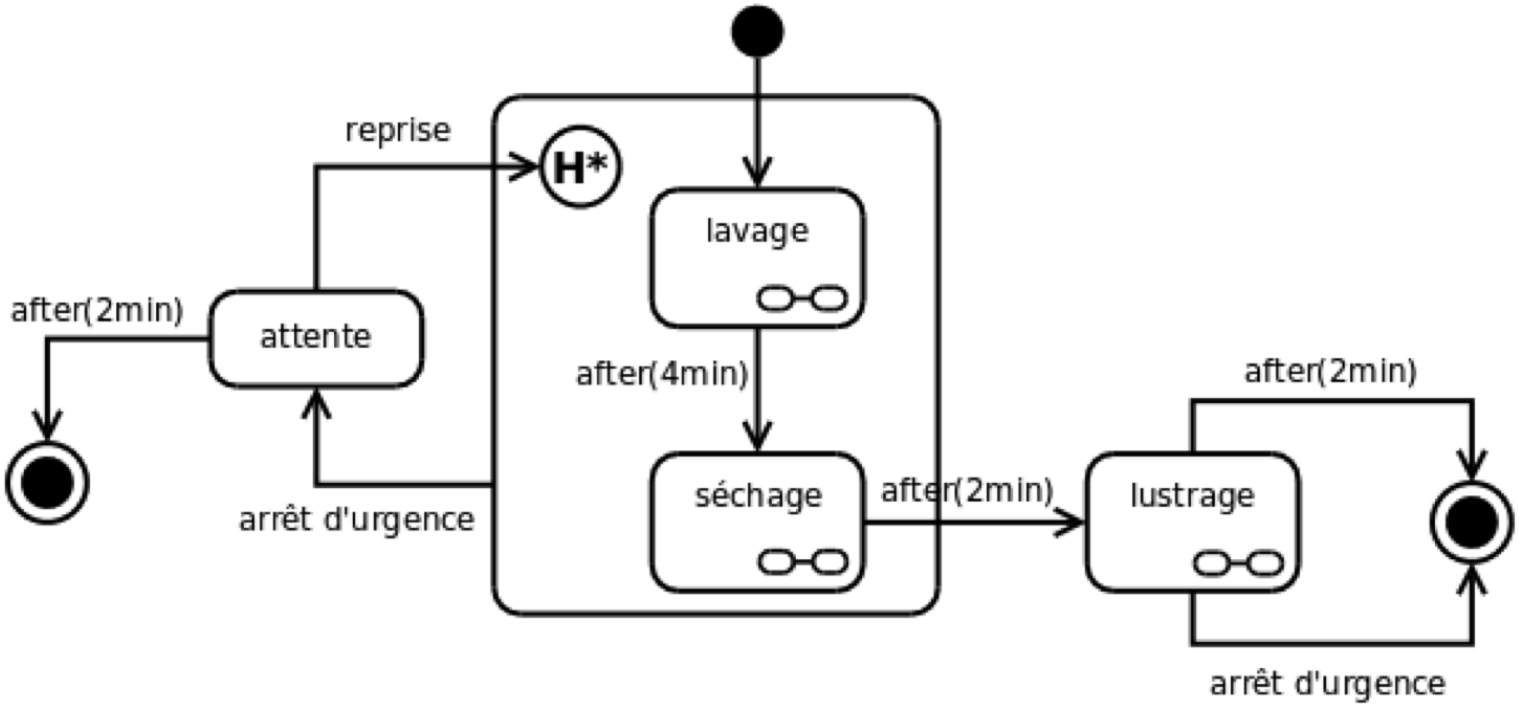
Un **état historique** est un pseudo-état qualifié d'*état historique plat*, qui mémorise le dernier sous-état actif d'un état composite. Graphiquement il est représenté par un cercle contenant un **H**.

- Une transition ayant pour cible l'état historique est équivalente à une transition qui a pour cible le dernier état visité de l'état englobant.
- Un état historique peut avoir une transition sortante non étiquetée indiquant l'état à exécuter si la région n'a pas encore été visitée.

Il est également possible de définir un *état historique profond* représenté graphiquement par un cercle contenant un **H***. Il permet d'atteindre le dernier état visité dans la région, quel que soit son niveau d'imbrication, alors que l'état historique plat limite l'accès aux états de son niveau d'imbrication.



# État historique



# Plan

# Diagramme d'activités : définition

## Diagramme d'activité

Le **diagramme d'activité** est un diagramme comportemental appelé *Activity Diagram* (**act**) dans le langage SysML. Il permet de modéliser le déroulement d'un processus sous la forme d'une activité correspondant à la décomposition séquentielle d'actions aussi appelées tâches. Les éléments graphiques utilisés dans ce diagramme sont principalement :

- des rectangles aux coins arrondis pour les états,
- des flèches orientées de l'état de départ à l'état cible pour les transitions,
- des occurrences attachées à la transition spécifient les conditions de franchissement.

## Remarque

- Ce diagramme ne possède aucun événement associé aux transitions entre actions : la fin d'une action implique automatiquement le passage à la suivante, donc dans un ordre déterminé d'actions menant à un résultat. Lorsque le processus est enclenché il va à son terme selon un ordre précis.
- Ce diagramme permet aussi de représenter des **algorigrammes**, c'est à dire un flux de contrôle.
- Ce diagramme ne figure pas explicitement dans le programme.

# Diagramme d'activités : définition

## Diagramme d'activité

Le **diagramme d'activité** est un diagramme comportemental appelé *Activity Diagram* (**act**) dans le langage SysML. Il permet de modéliser le déroulement d'un processus sous la forme d'une activité correspondant à la décomposition séquentielle d'actions aussi appelées tâches. Les éléments graphiques utilisés dans ce diagramme sont principalement :

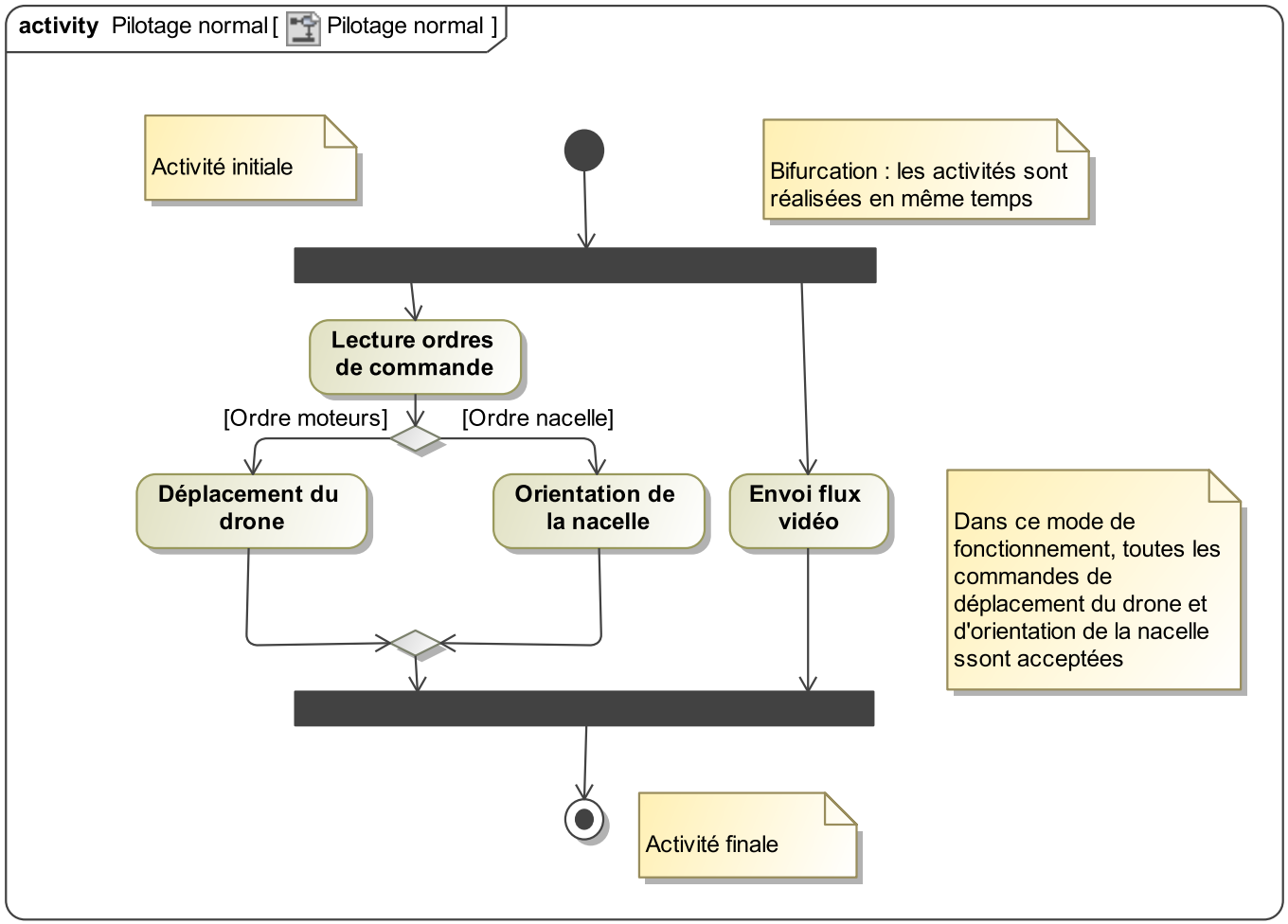
- des rectangles aux coins arrondis pour les états,
- des flèches orientées de l'état de départ à l'état cible pour les transitions,
- des occurrences attachées à la transition spécifient les conditions de franchissement.

## Remarque

- Ce diagramme ne possède aucun événement associé aux transitions entre actions : la fin d'une action implique automatiquement le passage à la suivante, donc dans un ordre déterminé d'actions menant à un résultat. Lorsque le processus est enclenché il va à son terme selon un ordre précis.
- Ce diagramme permet aussi de représenter des **algorigrammes**, c'est à dire un flux de contrôle.
- Ce diagramme ne figure pas explicitement dans le programme.

# Plan

# Diagramme d'activités : exemples



# Diagramme d'activités : exemples