# DIC9150 Concepts fondamentaux de l'informatique cognitive

## Planification

Roger Villemaire

Département d'informatique
UQAM

le 1er octobre 2024



UQÀM
Université du Québec à Montréal

# Plan

# Planification

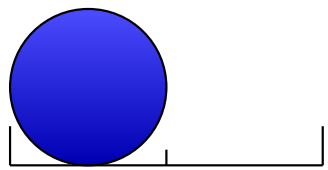
- Étant donnée une tâche, trouver une suite d'actions pour la réaliser.
- Exemple : faire cuire un œuf
  1. disposer une poêle sur la cuisinière,
  2. ouvrir le feu, pas trop fort,
  3. prendre un œuf dans le réfrigérateur,
  4. le casser sur le bord de la poêle,
  5. laisser tomber le contenu dans la poêle,
  6. cuire jusqu'à ce que ça soit prêt.

# Concepts

- État : description de la situation à un moment donné
  - l'œuf est dans la poêle ET le feu est allumé ET l'œuf n'est pas encore cuit, etc.
- État initial : la situation avant d'effectuer la tâche
  - l'œuf est dans le réfrigérateur ET la feu est éteint, etc.
- Action : une étape (passage d'un état à un autre) qu'on peut effectuer dans la réalisation de la tâche
  - mettre la poêle sur la cuisinière
- But : objectif visé
  - l'œuf est prêt à être consommé.

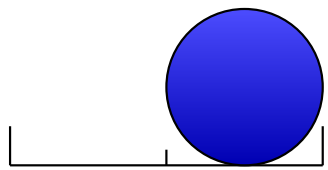
# Exemple

- Un robot peut se déplacer entre deux positions (gauche et droite).
- Initialement le robot est dans la position de gauche.
- Le robot peut faire deux actions : rouler vers la gauche ou vers la droite.
- Le but est que le robot soit à droite (par exemple, pour sortir par la porte).



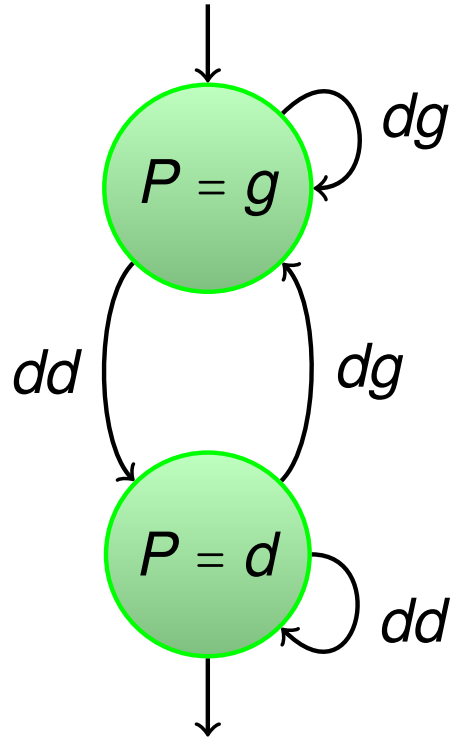
# Exemple

- Un robot peut se déplacer entre deux positions (gauche et droite).
- Initialement le robot est dans la position de gauche.
- Le robot peut faire deux actions : rouler vers la gauche ou vers la droite.
- Le but est que le robot soit à droite (par exemple, pour sortir par la porte).



# Représentation : Système de transition

- État : déterminé par la valeur de $P \in \{g, d\}$,
- État initial : $(P = g)$,
- Transition :
  - *dg* déplacement vers la gauche,
  - *dd* déplacement vers la droite,
- But : $(P = d)$,
- Un Plan : $(P = g) \xrightarrow{dd} (P = d)$.



# Définition : Système de transition

- Un système de transition est formé :
  - d'un ensemble d'*états*
  - un ensemble de *transitions* (ou *d'actions*)
- Un état est déterminé par des *conditions*
  - valeurs de variables, formule, etc.,
- Une transition est un changement d'un état à un autre déterminé par
  - ses *préconditions* qui doivent être satisfaites par l'état de départ,
  - ses *postconditions* qui doivent être satisfaites par l'état d'arrivée.

# Planificateur (Planner)

- Plusieurs outils de planification ont été développés au cours des années
- Le plus ancien, STRIPS (*Stanford Research Institute Problem Solver*), a été développé par Richard Fikes and Nils Nilsson (1971) chez SRI International (Institut sans but lucratif, spinoff de l'université Stanford, CA).
- Cet outil a introduit un langage, encore utilisé aujourd'hui, qui a pris lui aussi le nom de STRIPS.

# PDDL

- Le langage STRIPS a été étendu pour définir le *Planning Domain Definition Language (PDDL)*,
  - développé pour l'*International Planning Competition (IPC)* 1998,
  - `http://icaps-conference.org/index.php/Main/Competitions`,
  - une compétition de planification chaque année !
- La plupart des outils actuels supportent (une partie) de PDDL.

# Références

- Nous allons nous limiter aux aspects principaux de PDDL pour développer quelques exemples.
  - Description détaillée de PDDL 3.1 `https://helios.hud.ac.uk/scommv/IPC-14/repository/kovacs-pddl-3.1-2011.pdf`,
  - Texte plus accessible au sujet de la version 1.2 `http://homepages.inf.ed.ac.uk/mfourman/tools/propplan/pddl.pdf`
  - Beaucoup d'exemples un peu partout sur l'internet.

# PDDL : Domaine et Problème

- Une description PDDL est divisée en deux parties :
  - la description du domaine
  - la description du problème

# Solveur PDDL

- Prend en entrée les description du domaine et celle du problème.
- Produit un plan possible ou indique qu'il n'y en a pas.

# Notation à la Lisp

- `(and x y)` plutôt que `and(x,y)`,
- `(f u v)` plutôt que `f(u,v)`.

# Domaine PDDL : Robot

domaineRobot1.pddl

```
;; commentaire
(define (domain domaineRobot)
  (:constants Porte)
  (:predicates (positionRobot ?p))
  (:action allerALaPorte
           :effect (positionRobot Porte)))
```

# Problème PDDL, Robot

problemeRobot1.pddl

```
(define (problem problemeRobot)
  (:domain domaineRobot)
  (:objects p)
  (:init (positionRobot p))
  (:goal (positionRobot Porte)))
```

# Outil PDDL : Planning.Domains

- SaaS : Software as a Service
- `http://planning.domains/`

# Domaine PDDL : Robot

domaineRobot2.pddl

```
(define (domain domaineRobot)
  (:types position robot)
  (:predicates (voisines ?p1 - position
                         ?p2 - position)
               (robotPosition ?r - robot
                              ?p - position))
  (:action deplacer
           :parameters (?r - robot
                        ?pd - position
                        ?pa - position)
    :precondition (and (robotPosition ?r ?pd)
                       (voisines ?pd ?pa))
    :effect (robotPosition ?r ?pa))
)
```

# Problème PDDL, Robot

problemeRobot2.pddl

```
(define (problem problemeRobot)
  (:domain domaineRobot)

  (:objects r - robot
            p0 - position
            p1 - position)

  (:init (robotPosition r p0)
         (voisines p0 p1))

  (:goal (robotPosition r p1))
)
```

# Conclusion

- La planification est une fonction cognitive fondamentale.
- Les outils de planification permettent de
  - représenter les objets et les actions,
  - et de déterminer un plan permettant de réaliser la tâche.
- Il s'agit d'une application représentative de l'approche symbolique
  - les objets et actions sont représentés sous forme de *symboles*,
  - le plan est déterminé par des manipulations symboliques,
  - l'énorme efficacité de la machine, pour ce genre de manipulations, est donc mise en application pour réaliser la fonction cognitive.