IFT 608 / IFT 702
Planification en intelligence artificielle

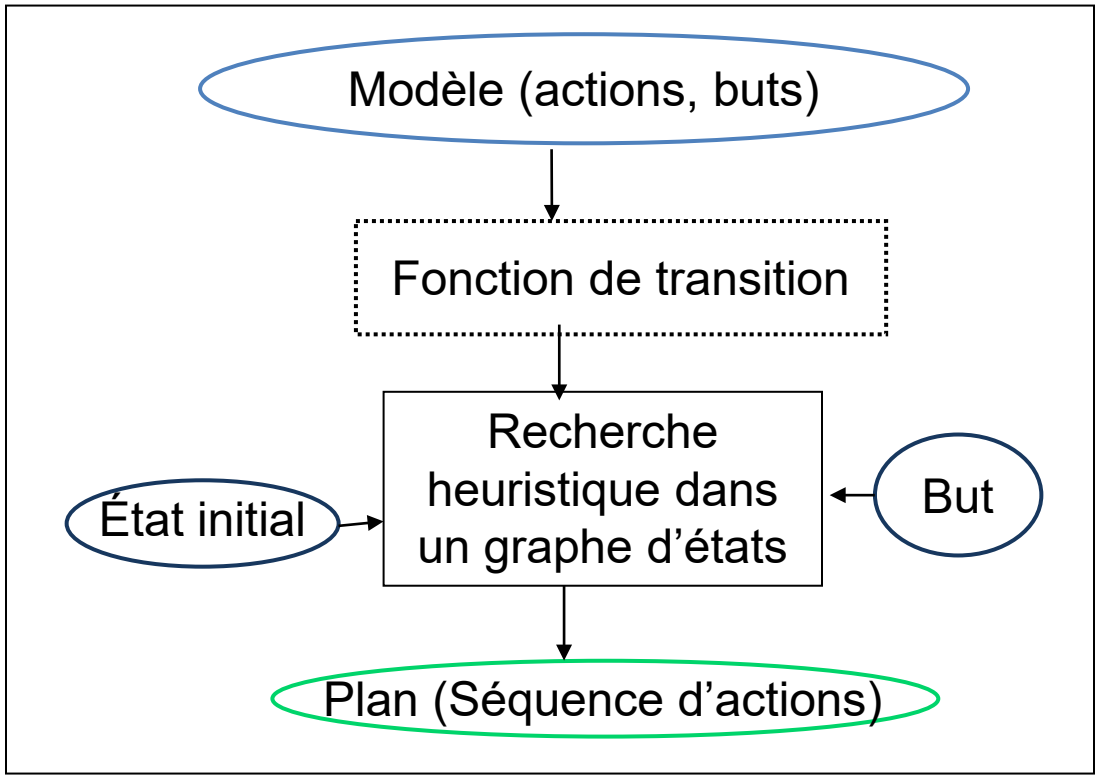
Langage PDDL

Froduald Kabanza
Département d'informatique
Université de Sherbrooke

Contenu

- STRIPS
- PDDL
- Intégration avec un planificateur

Architecture générale d'un planificateur déterministe par recherche dans un espace d'états



- Le modèle ne décrit pas les capteurs puisque l'environnement est déterministe.
- Le modèle est transformé en fonction de transition pour un graphe d'états.

Langages de modélisation

- Langage STRIPS :
  - Décrit une action élémentaire en fonction de trois éléments:
    - Précondition
    - Effets positifs
    - Effets négatif
  - Limitations
    - Pas moyen de spécifier des contraintes sur la durée des actions
    - Pas moyen de spécifier des effets conditionnels
    - Pas moyen de spécifier des contraintes numériques

- Langage PDDL (*Planning Domain Definition Language*) :
  - Une extension de STRIPS levant les restrictions précédentes
  - Un “quasi-standard académique” pour les algorithmes de planification (conférence ICAPS)
  - PDDL 2.1

Prérequis IFT615

- IFT 615 (Raisonnement logique)
  - Logique du premier ordre
  - Algorithme d'unification

Langage STRIPS

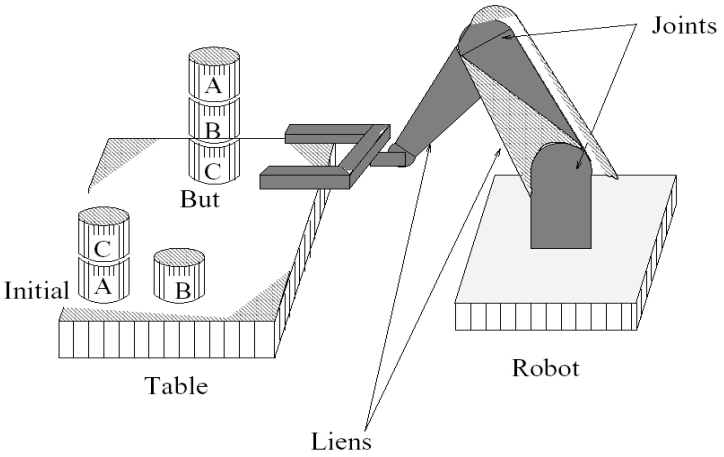
- Dans les cas les plus simples les actions sont décrites par des *opérateurs*. Un *opérateur* :
  - une précondition : conjonction de litéraux;
  - les effets positifs : litéraux positifs (*add-list*);
  - les effets négatifs : litéraux négatifs (*delete-list*);
- Un opérateur peut contenir des variables.
- Une action est un opérateur complétement instantié;
- Une action est possible (*enabled*) dans un état uniquement si l'état contient les préconditions.
- Le successeur est obtenu en supprimant d'abord les *effets negatifs* et en ajoutant ensuite les *effets positifs*.

Transformation du modèle pour avoir une fonction de transition

Exemple 1: Monde des blocs

Un robot doit empiler des blocs dans une configuration indiquée.
C'est une version simplifiée d'un robot de manipulation de conteneurs dans un port.

Monde des blocks (*Blocksworld* en anglais)



Micro-environnement didactique, couramment utilisé en IA.

Définition des actions

```
(define (domain blocksword)
  (:requirements :strips :typing)
  (:types block)
  (:predicates (on ?x - block ?y - block)
               (ontable ?x - block)
               (clear ?x - block)
               (handempty)
               (holding ?x – block))
```

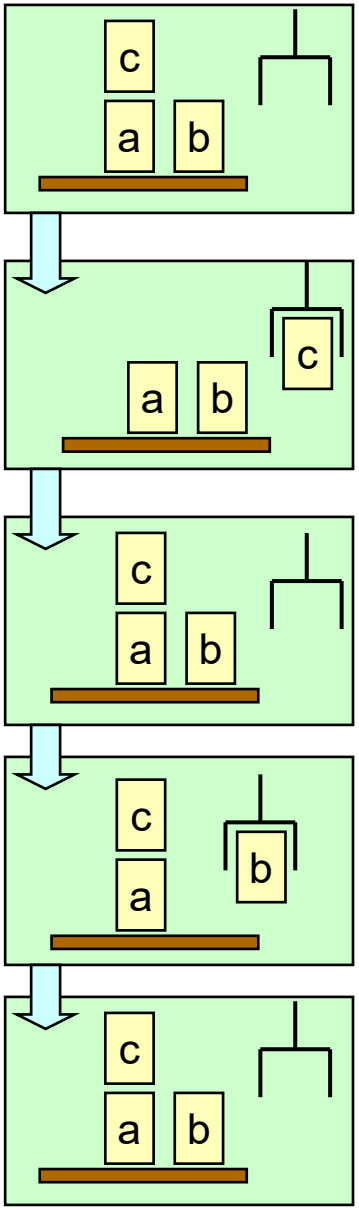
Exemple STRIPS pour le monde des blocks

```
(:action unstack
  :parameters (?x – block ?y - block)
  :precondition  (and (on ?x ?y) (clear ?x) (handempty)
  :effects  (and (not  (on ?x ?y)) (not (clear ?x))
                 (not (handempty)) (holding ?x) (clear ?y))
```

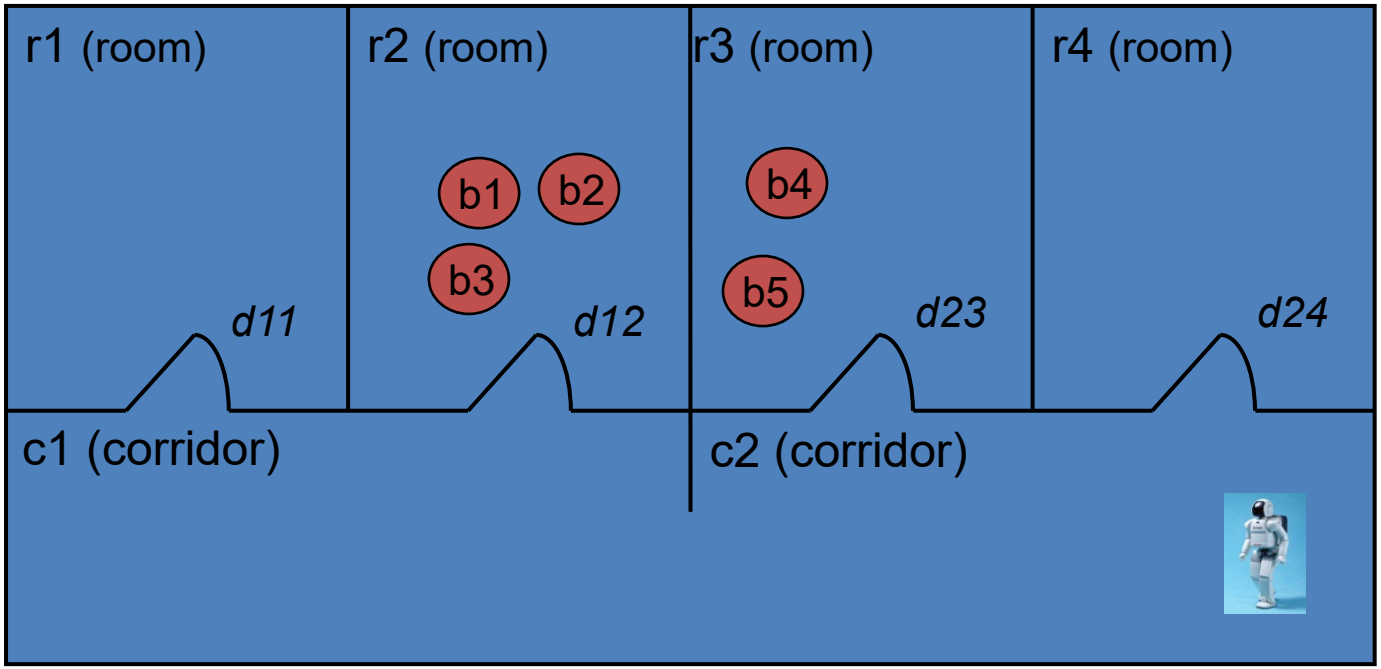
```
(:action stack
   :parameters (?x – block ?y - block)
   :precondition (and (holding ?x) (clear ?y))
   :effects (and (not (holding ?x)) (not (clear ?y))
                 (on ?x ?y) (clear ?x) (handempty))
```

```
(:action pick-up
  :parameters (?x – block)
  :precondition  (and (ontable ?x) (clear ?x) (handempty)
  :effects   (and (not (ontable ?x)) (not (clear ?x)) (not (handempty))
                  (holding ?x))
```

```
(:action put-down
:parameters (?x – block)
 :precondition (holding ?x)
 :effects  (and (not (holding ?x)) (ontable ?x) (clear ?x)
                (handempty))
```



Exemple pour la livraison de colis

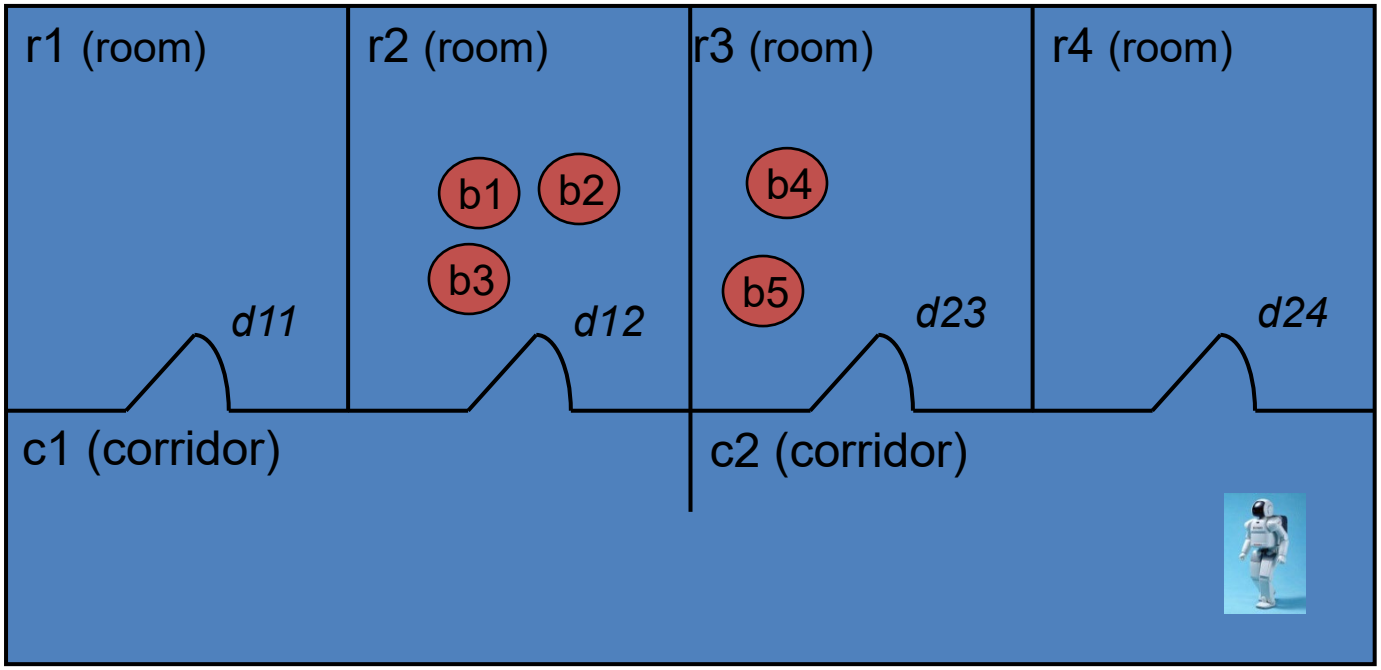



```
(define (domain robotworld)
  (:requirements :typing)
  (:types gripper room ball)
  (:predicates    (atRobot ?r - room)
                  (at ?b - ball ?r - room)
                  (free ?g - gripper)
                  (holding ?g - gripper ?b - ball)  )
```

Exemple pour la livraison de colis, suite

```
(:action pick
  :parameters (?g - gripper ?b - ball ?r - room)
  :precondition (and (free ?g) (atRobot ?r) (at ?b ?r))
  :effect (and (holding ?g ?b) (not (free ?g)) (not (at ?b ?r))))

(:action drop
 :parameters (?g - gripper ?b - ball ?r - room)
 :precondition (and (holding ?g ?b) (atRobot ?r))
 :effect (and (free ?g) (not (holding ?g ?b)) (at ?b ?r)))

(:action move
 :parameters (?from ?to - room)
 :precondition (atRobot ?from)
 :effect (and (not (atRobot ?from)) (atRobot ?to))))

) ; Ferme le “define”
```

Exemple PDDL pour la livraison de colis

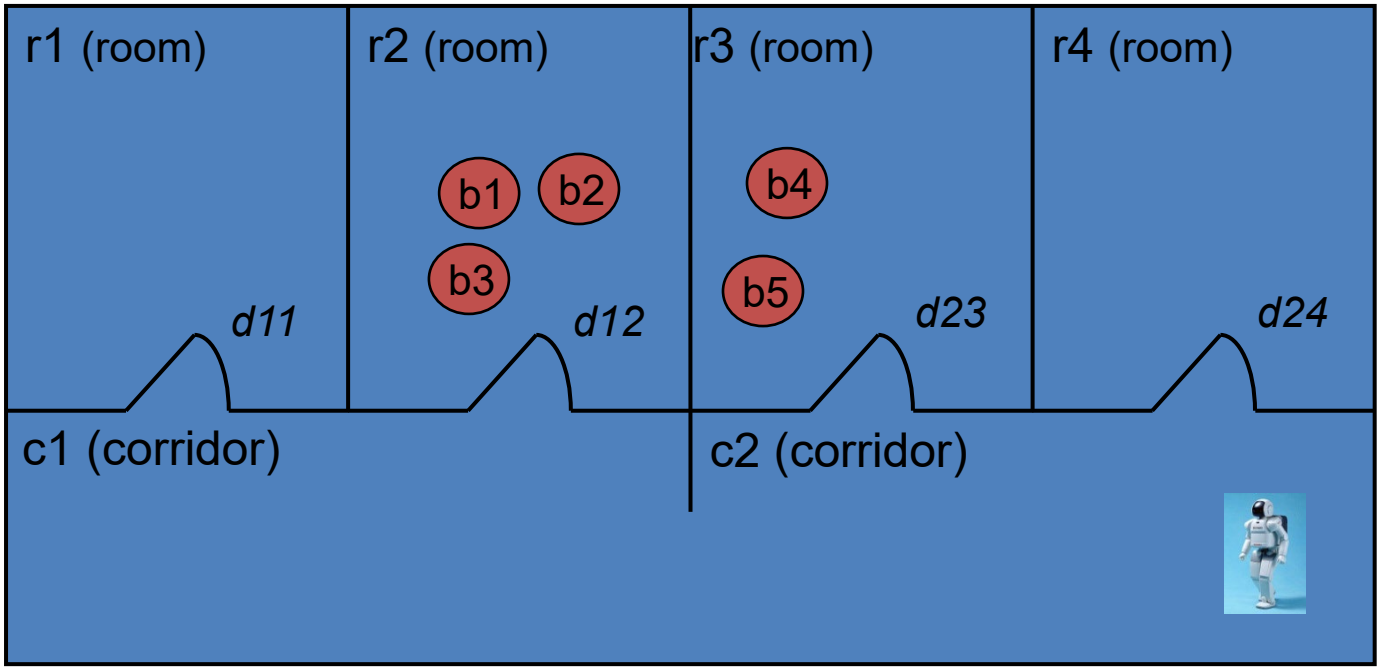



```
(:objects r1 r2 r3 r4 - room
          c1 c2 - corridor)

(:init (atRobot r2)
       (= (free) 2) (= (holding) 0)
       (= (atBalls r3) 2)(= (atBalls r2) 3)

(:goal (= (atBalls r4) 5))
```

Exemple PDDL pour la livraison de colis, suite

```
(define (domain robotWorld2)
  (:types ball room corridor)
  (:predicates at atRobot)

   (:action pickup
    :parameters (?r - room)
    :precondition (and (> (atBalls ?r) 0) (atRobot ?r) (> (free) 0))
    :effect (and (increase (holding) 1) (decrease (atBalls ?r) 1)
                 (decrease (free) 1)))
   (:action release
    :parameters (?r - room)
    :precondition (and (> (holding) 0) (atRobot ?r))
    :effect (and (increase (atBalls ?r) 1) (decrease (holding) 1)
                 (increase (free) 1)))
   (:action move
    :parameters (?rf ?rt - room)
    :precondition (and (atRobot ?rf)))
    :effect (and (atRobot ?rt) (not (atRobot ?rf))))
```

Une autre version PDDL pour la livraison de colis




```
(:objects b1 b2 b3 b4 b5 - ball
          left right - gripper
          r1 r2 r3 r4 - room
          c1 c2 - corridor)
(:init (atRobot c2) (free left) (free right)
       (at b1 r2) (at b2 r2) (at b3 r2) (at b4 r3) (at b5 r3))
(:goal (at b1 r4) (at b2 r4) (at b3 r4) (at b4 r4) (at b5 r4))
(:goal (forall (?x - ball)(at ?x r4)))
```

Une autre version PDDL pour la livraison de colis

```
(define (domain robotWorld3)
  (:types gripper ball room corridor door)
  (:predicates atRobot at free holding connects)

  (:action pick
     :parameters (?b - ball ?g - gripper ?r - room)
     :precondition (and (at ?b ?r) (atRobot ?r) (free ?g)
                        (not (exists (?y - ball ?z gripper)
                                     (holding ?y ?z))))
     :effect (and (holding ?g ?b)(not (free ?g))))

  (:action release
   :parameters (?b - ball ?g - gripper ?r - room)
   :precondition (and (holding ?g ?b) (atRobot ?r))
   :effect (and(not (holding ?g ?b)) (free ?g)))

  (:action move
   :parameters (?rf ?rt)
   :precondition (atRobot ?rf)
   :effect (and (atRobot ?rt) (not (atRobot ?rf)))
                (forall (?b - ball ?g - gripper)
                        (when (holding ?b ?g)
                              (and (not (at ?b ?rf))
                                   (at ?b ?rt))))))
```

Ressources

- PDDL Editor
- *Writing Planning Domains and Problems in PDDL*
- *International Planning Competition*

- Exemple: **PlanSys** (https://plansys2.github.io/)
  - Intégré avec ROSPlan

Ce qu'il faut retenir

- PDDL
  - Syntaxe bien définie
  - Sémantique bien définie
  - Pas toujours applicables:
    - Difficile pour un humain de modéliser les connaissances
    - C'est un défi de raisonner avec de telles connaissances
- Principalement un concept de recherche scientifique (par opposition à « outil largement adopté dans les applications »), largement exploré en IA
  - Représentation compacte de la fonction de transition.
  - Pourrait paver la voie au raisonnement automatique:
    - Extraction automatique d'heuristiques
    - Apprentissage automatique du modèle d'action
    - Interprétabilité
    - Explicabilité