

# Intelligence artificielle

## Introduction à la planification

---

Elise Bonzon

`elise.bonzon@u-paris.fr`

LIPADE - Université Paris Cité

<http://www.math-info.univ-paris5.fr/~bonzon/>

# Introduction à la planification

1. Qu'est ce que la planification ?
2. Langage de représentation STRIPS
3. Planification dans les espaces d'états
4. Planification dans les espaces de plans
5. Conclusion

**Qu'est ce que la planification ?**

---

# Qu'est ce que la planification ?

- Tâche qui consiste à trouver une séquence d'actions qui réalisera un objectif

# Qu'est ce que la planification ?

- Tâche qui consiste à trouver une séquence d'actions qui réalisera un objectif
  - systèmes **autonomes** et **adaptables** capables de raisonner sur leurs capacités et les **contraintes provenant de l'environnement**

# Qu'est ce que la planification ?

- Tâche qui consiste à trouver une séquence d'actions qui réalisera un objectif
  - systèmes **autonomes** et **adaptables** capables de raisonner sur leurs capacités et les **contraintes provenant de l'environnement**
- Définir des **mécanismes génériques** permettant à des agents autonomes de se **coordonner** et de **coopérer** afin de d'accomplir des tâches complexes qu'ils ne peuvent pas réaliser seuls

# Qu'est ce que la planification ?

- Tâche qui consiste à trouver une séquence d'actions qui réalisera un objectif
  - systèmes **autonomes** et **adaptables** capables de raisonner sur leurs capacités et les **contraintes provenant de l'environnement**
- Définir des **mécanismes génériques** permettant à des agents autonomes de se **coordonner** et de **coopérer** afin de d'accomplir des tâches complexes qu'ils ne peuvent pas réaliser seuls
- Définir des mécanismes abstraits pour permettre la **composition automatique** de fonctionnalités

# Qu'est ce que la planification ?

- Représentation des problèmes de planification : états, actions et objectifs
  - Description d'un ou plusieurs état(s) initial(aux)
  - Description d'un ou plusieurs état(s) but(s)
  - Description d'un ensemble d'actions que l'agent peut effectuer
- Trouver une séquence d'actions qui mène l'agent de l'état initial à l'état but



- L'agent peut être submergé par des actions non pertinentes

# Limites des algorithmes de recherche

- L'agent peut être submergé par des actions non pertinentes
- Les heuristiques et les tests de buts sont inadéquats : il faut trouver des heuristiques indépendantes du domaine !

# Limites des algorithmes de recherche

- L'agent peut être submergé par des actions non pertinentes
- Les heuristiques et les tests de buts sont inadéquats : il faut trouver des **heuristiques indépendantes du domaine** !
- Pas d'exploitation de la décomposition du problème

- L'agent peut être submergé par des actions non pertinentes
  - Les heuristiques et les tests de buts sont inadéquats : il faut trouver des **heuristiques indépendantes du domaine** !
  - Pas d'exploitation de la décomposition du problème
- ⇒ Nous voulons trouver un langage qui est à la fois

# Limites des algorithmes de recherche

- L'agent peut être submergé par des actions non pertinentes
- Les heuristiques et les tests de buts sont inadéquats : il faut trouver des **heuristiques indépendantes du domaine** !
- Pas d'exploitation de la décomposition du problème

⇒ Nous voulons trouver un langage qui est à la fois

- **suffisamment expressif** pour décrire une grande variété de problèmes, mais

# Limites des algorithmes de recherche

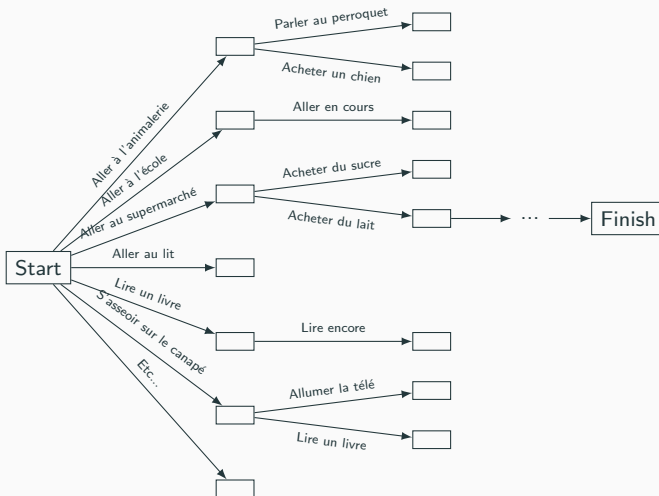
- L'agent peut être submergé par des actions non pertinentes
- Les heuristiques et les tests de buts sont inadéquats : il faut trouver des **heuristiques indépendantes du domaine** !
- Pas d'exploitation de la décomposition du problème

⇒ Nous voulons trouver un langage qui est à la fois

- **suffisamment expressif** pour décrire une grande variété de problèmes, mais
- **assez restrictif** pour permettre à des algorithmes efficaces d'agir

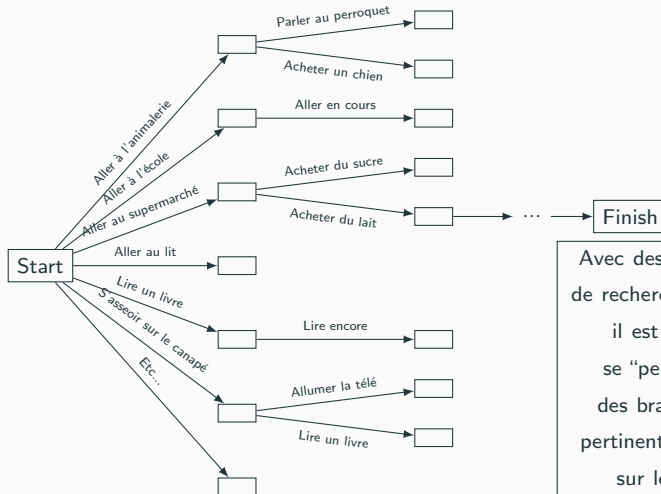
# Limites des algorithmes de recherche

But : Acheter du lait et une perceuse sans fil



# Limites des algorithmes de recherche

But : Acheter du lait et une perceuse sans fil



Avec des algorithmes de recherche classique, il est facile de se "perdre" dans des branches non pertinentes (s'asseoir sur le canapé par exemple...)



# Générer un plan

- Etant donné
  - Une description du monde
  - Un état initial du monde
  - Une description du but
  - Un ensemble d'actions disponibles pour changer le monde
- On veut trouver
  - Une liste d'actions permettant de passer de l'état initial à un état qui satisfait le but
- Représenter les changements
  - Savoir comment une action modifie le monde
  - Garder en mémoire les états du monde (a-t'on déjà été dans cet état avant ?)
  - Pouvoir répondre aux questions à propos des états du monde *potentiels* (qu'est-ce qu'il se passerait si... ?)

- Les environnements considérés dans la **planification classique** sont :
  - Totalement observables
  - Déterministes
  - Finis
  - Statiques – les changements ne sont dus qu'aux seules actions des agents
  - Discrets – le temps est implicite, les actions n'ont pas de durée
- La planification est "offline", le planificateur ne prend pas en compte les changements pouvant survenir pendant qu'il planifie

- Militaire
  - Mission de déminage naval
  - Mission d'hélicoptères de combats
- Aérospaciale
  - Deep Space One
  - Prises de vues satellitaires
- Robotique industrielle
  - Ligne d'assemblage, transport
- Vie de tous les jours
  - Aspirateur automatique, tondeuse automatique
- Informatique
  - Composition de services web
  - Informatique ambiante ⇒ nouvelles applications

# Langage de représentation STRIPS

---

- L'état du monde est décrit par une série de prédicats en **logique du premier ordre**
- **État** = conjonction de littéraux **positifs**
  - Exemple :
$$A(\text{Avion1}, \text{Paris}) \wedge A(\text{Avion2}, \text{Toulouse})$$
  - Les états (littéraux en logique du premier ordre) doivent être **sans variable ni fonction**

# Hypothèse du monde clos

## Hypothèse du monde clos

**Hypothèse du monde clos** : Tout ce qui n'est pas explicitement défini comme étant vrai dans un état est considéré comme étant faux

# Hypothèse du monde clos

## Hypothèse du monde clos

**Hypothèse du monde clos** : Tout ce qui n'est pas explicitement défini comme étant vrai dans un état est considéré comme étant faux

Il est donc inutile de mettre des littéraux **négatifs** dans la description d'un état : pour dire que quelque chose est faux, il suffit de ne pas l'écrire, et ce sera considéré comme étant faux

- **But** = conjonction de littéraux **positifs** sans variable ni fonction
  - Le but est satisfait si **au moins** les littéraux spécifiés sont vrais, d'autre littéraux peuvent aussi être vérifiés
  - Plusieurs états différents peuvent donc satisfaire un but donné
- Représentation des **actions** :
  - Nom de l'action + paramètres (ex : `Voler(p,from,to)`)
  - **Pré-conditions** : conjonction de **littéraux positifs**, *ce qui doit être vrai pour pouvoir appliquer l'action*
  - **Effets** : conjonction de **littéraux (positifs et négatifs)**, *ce qui est vrai ou n'est plus vrai une fois que l'action a été appliquée*



- Une **action** est applicable à chaque état qui satisfait les pré-conditions

- Une **action** est applicable à chaque état qui satisfait les pré-conditions
- Logique du premier ordre : un état satisfait les pré-conditions s'il existe une **substitution** permettant d'unifier les pré-conditions avec les littéraux de l'état

- Une **action** est applicable à chaque état qui satisfait les pré-conditions
- Logique du premier ordre : un état satisfait les pré-conditions s'il existe une **substitution** permettant d'unifier les pré-conditions avec les littéraux de l'état
- Exemple

- Une **action** est applicable à chaque état qui satisfait les pré-conditions
- Logique du premier ordre : un état satisfait les pré-conditions s'il existe une **substitution** permettant d'unifier les pré-conditions avec les littéraux de l'état
- Exemple
  - **Etat** :  $A(A1, Orly) \wedge A(A2, Blagnac) \wedge Avion(A1) \wedge Avion(A2) \wedge Aeroport(Orly) \wedge Aeroport(Blagnac)$

- Une **action** est applicable à chaque état qui satisfait les pré-conditions
- Logique du premier ordre : un état satisfait les pré-conditions s'il existe une **substitution** permettant d'unifier les pré-conditions avec les littéraux de l'état
- Exemple
  - **Etat** :  $A(A1, Orly) \wedge A(A2, Blagnac) \wedge Avion(A1) \wedge Avion(A2) \wedge Aeroport(Orly) \wedge Aeroport(Blagnac)$
  - **Pré-conditions** :  $A(x, from) \wedge Avion(x) \wedge Aeroport(from) \wedge Aeroport(to)$

- Une **action** est applicable à chaque état qui satisfait les pré-conditions
- Logique du premier ordre : un état satisfait les pré-conditions s'il existe une **substitution** permettant d'unifier les pré-conditions avec les littéraux de l'état
- Exemple
  - **Etat** :  $A(A1, Orly) \wedge A(A2, Blagnac) \wedge Avion(A1) \wedge Avion(A2) \wedge Aeroport(Orly) \wedge Aeroport(Blagnac)$
  - **Pré-conditions** :  $A(x, from) \wedge Avion(x) \wedge Aeroport(from) \wedge Aeroport(to)$
  - **Substitution** :  $\theta = \{x/A1, from/Orly, to/Blagnac\}$

- Une **action** est applicable à chaque état qui satisfait les pré-conditions
- Logique du premier ordre : un état satisfait les pré-conditions s'il existe une **substitution** permettant d'unifier les pré-conditions avec les littéraux de l'état
- Exemple
  - **Etat** :  $A(A1, Orly) \wedge A(A2, Blagnac) \wedge Avion(A1) \wedge Avion(A2) \wedge Aeroport(Orly) \wedge Aeroport(Blagnac)$
  - **Pré-conditions** :  $A(x, from) \wedge Avion(x) \wedge Aeroport(from) \wedge Aeroport(to)$
  - **Substitution** :  $\theta = \{x/A1, from/Orly, to/Blagnac\}$
  - **Substitution 2** :  $\theta = \{x/A2, from/Blagnac, to/Orly\}$

- Action  $a$  menant d'un état  $s_1$  à un état  $s_2$ 
  - Chaque littéral positif  $l$  dans l'**effet** de  $a$  est ajouté à  $s_2$ 
    - Si  $l$  est déjà dans  $s_2$  il n'est pas ajouté deux fois
  - Chaque littéral négatif  $\neg l$  dans l'**effet** de  $a$  est supprimé de  $s_2$ 
    - Si  $l$  n'est pas dans  $s_2$ , rien ne change
- **Solution** d'un problème de planification : séquence d'actions permettant de mener d'un état initial à un état qui satisfait le but



- **STRIPS** : **Stanford Research Institute Problem Solver** (1971)
  - **Action** : nom de l'action et paramètres nécessaires pour appliquer l'action
  - **Préconditions** : ensemble des prédicats devant être **vrais** pour pouvoir appliquer l'action
  - **Effets** : modifications des prédicats après l'exécution de l'action
    - **Retraits** : prédicats qui étaient vrais et qui sont maintenant faux
    - **Ajouts** : prédicats qui étaient faux et qui sont maintenant vrais
- Suffisamment expressif pour décrire de nombreux problèmes
- Suffisamment restrictif pour être utilisé par des algorithmes efficaces

## Exemple : le monde des blocs

- L'univers est composé d'un ensemble de blocs cubiques et d'une table
- Les blocs sont mobiles, la table est immobile
- Un agent déplace les blocs
- Un bloc peut être sur la table ou sur un autre bloc
- Il ne peut pas y avoir plus d'un bloc sur un autre bloc
- La table est assez grande pour que tous les blocs puissent y prendre place
- L'ordre des blocs sur l'axe horizontal n'est pas important

## Exemple : le monde des blocs

- **Prédicats :**
  - **Bloc(x)** : Bloc(A), Bloc(B), Bloc(C), Bloc(D)
  - **Sur(x, y)** : Sur(C,B), Sur(B,A)
  - **SurTable(x)** : SurTable(A), SurTable(D)
  - **Libre(x)** : Libre(C), Libre(D)

## Exemple : le monde des blocs

- **Prédicats** :
  - **Bloc(x)** : Bloc(A), Bloc(B), Bloc(C), Bloc(D)
  - **Sur(x, y)** : Sur(C,B), Sur(B,A)
  - **SurTable(x)** : SurTable(A), SurTable(D)
  - **Libre(x)** : Libre(C), Libre(D)
- L'**état** du monde est une conjonction de prédicats :  
$$\text{Bloc(A)} \wedge \text{Bloc(B)} \wedge \text{Bloc(C)} \wedge \text{Bloc(D)} \wedge \text{SurTable(A)} \wedge$$
$$\text{Sur(B, A)} \wedge \text{Sur(C, B)} \wedge \text{SurTable(D)} \wedge \text{Libre(C)} \wedge$$
$$\text{Libre(D)}$$

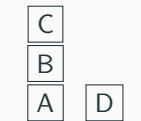
## Exemple : le monde des blocs

- **Prédicats :**

- **Bloc(x)** : Bloc(A), Bloc(B), Bloc(C), Bloc(D)
- **Sur(x, y)** : Sur(C,B), Sur(B,A)
- **SurTable(x)** : SurTable(A), SurTable(D)
- **Libre(x)** : Libre(C), Libre(D)

- **L'état** du monde est une conjonction de prédicats :

$\text{Bloc(A)} \wedge \text{Bloc(B)} \wedge \text{Bloc(C)} \wedge \text{Bloc(D)} \wedge \text{SurTable(A)} \wedge$   
 $\text{Sur(B, A)} \wedge \text{Sur(C, B)} \wedge \text{SurTable(D)} \wedge \text{Libre(C)} \wedge$   
 $\text{Libre(D)}$



## Exemple : le monde des blocs

### Actions :

- Déplacer le bloc  $b$ , qui est sur le bloc  $x$ , pour le mettre sur le bloc  $y$

Action(Déplacer( $b, x, y$ ) :

PRECOND :  $\text{Sur}(b, x) \wedge \text{Libre}(b) \wedge \text{Libre}(y) \wedge \text{Bloc}(b) \wedge$   
 $\text{Bloc}(x) \wedge \text{Bloc}(y) \wedge (b \neq x) \wedge (b \neq y) \wedge (x \neq y)$

EFFET :  $\text{Sur}(b, y) \wedge \text{Libre}(x) \wedge \neg \text{Sur}(b, x) \wedge \neg \text{Libre}(y)$

# Exemple : le monde des blocs

## Actions :

- Déplacer le bloc  $b$ , qui est sur le bloc  $x$ , pour le mettre sur le bloc  $y$

$\text{Action}(\text{Deplacer}(b,x,y) :$

$\text{PRECOND} : \text{Sur}(b,x) \wedge \text{Libre}(b) \wedge \text{Libre}(y) \wedge \text{Bloc}(b) \wedge$   
 $\text{Bloc}(x) \wedge \text{Bloc}(y) \wedge (b \neq x) \wedge (b \neq y) \wedge (x \neq y)$

$\text{EFFET} : \text{Sur}(b,y) \wedge \text{Libre}(x) \wedge \neg \text{Sur}(b,x) \wedge \neg \text{Libre}(y))$

- Déplacer le bloc  $b$ , qui est sur  $x$ , pour le mettre sur la table

$\text{Action}(\text{DeplacerSurTable}(b,x)$

$\text{PRECOND} : \text{Sur}(b,x) \wedge \text{Libre}(b) \wedge \text{Bloc}(b) \wedge \text{Bloc}(x) \wedge (b \neq$   
 $x)$

$\text{EFFET} : \text{Sur}(b,\text{Table}) \wedge \text{Libre}(x) \wedge \neg \text{Sur}(b,x))$

# Exemple : le monde des blocs

## Actions :

- Déplacer le bloc  $b$ , qui est sur le bloc  $x$ , pour le mettre sur le bloc  $y$

Action(Déplacer( $b, x, y$ ) :

PRECOND :  $\text{Sur}(b, x) \wedge \text{Libre}(b) \wedge \text{Libre}(y) \wedge \text{Bloc}(b) \wedge$   
 $\text{Bloc}(x) \wedge \text{Bloc}(y) \wedge (b \neq x) \wedge (b \neq y) \wedge (x \neq y)$

EFFET :  $\text{Sur}(b, y) \wedge \text{Libre}(x) \wedge \neg \text{Sur}(b, x) \wedge \neg \text{Libre}(y)$

- Déplacer le bloc  $b$ , qui est sur  $x$ , pour le mettre sur la table

Action(DéplacerSurTable( $b, x$ )

PRECOND :  $\text{Sur}(b, x) \wedge \text{Libre}(b) \wedge \text{Bloc}(b) \wedge \text{Bloc}(x) \wedge (b \neq$   
 $x)$

EFFET :  $\text{Sur}(b, \text{Table}) \wedge \text{Libre}(x) \wedge \neg \text{Sur}(b, x)$

- Déplacer le bloc  $b$ , qui est sur la table, pour le mettre sur  $x$

Action(DéplacerDeTable( $b, x$ )

PRECOND :  $\text{Sur}(b, \text{Table}) \wedge \text{Libre}(b) \wedge \text{Libre}(x) \wedge \text{Bloc}(b) \wedge$   
 $\text{Bloc}(x) \wedge (b \neq x)$

EFFET :  $\neg \text{Sur}(b, \text{Table}) \wedge \neg \text{Libre}(x) \wedge \text{Sur}(b, x)$



## Exemple : le monde des blocs

- Etant donné un problème
  - **Etat initial** :  $\text{Bloc}(A) \wedge \text{Bloc}(B) \wedge \text{Bloc}(C) \wedge \text{Bloc}(D) \wedge \text{Sur}(C, B) \wedge \text{Sur}(B, A) \wedge \text{SurTable}(A) \wedge \text{SurTable}(D) \wedge \text{Libre}(C) \wedge \text{Libre}(D)$
  - **But** :  $\text{Sur}(A, B) \wedge \text{Sur}(C, D) \wedge \text{SurTable}(B) \wedge \text{SurTable}(D) \wedge \text{Libre}(A) \wedge \text{Libre}(C)$
- Planification : trouver un enchaînement d'actions permettant de passer de l'état initial à un état satisfaisant le but
- Problème difficile
- Plusieurs méthodes de planifications existent

# Planification dans les espaces d'états

---

# Planification dans les espaces d'états

- Algorithmes les plus simples pour la planification
- Algorithmes de recherche dans lesquels l'espace de recherche est un sous-ensemble de l'espace d'états
  - **Nœud** : état du monde
  - **Arc** : transition, action
  - **Chemin** entre l'état initial et un état satisfaisant le but : plan d'actions
- Tout algorithme d'exploration complet (comme par exemple  $A^*$ ) est un algorithme de planification complet

# **Planification dans les espaces d'états**

---

**Planification par recherche en chaînage  
avant**

# Planification par recherche en chaînage avant

- Procède de l'état initial vers l'état final en utilisant les techniques d'exploration

# Planification par recherche en chaînage avant

- Procède de l'état initial vers l'état final en utilisant les techniques d'exploration
- Le facteur de branchement est élevé

# Planification par recherche en chaînage avant

- Procède de l'état initial vers l'état final en utilisant les techniques d'exploration
- Le facteur de branchement est élevé
  - Toutes les actions applicables sont prises en considération

# Planification par recherche en chaînage avant

- Procède de l'état initial vers l'état final en utilisant les techniques d'exploration
- Le facteur de branchement est élevé
  - Toutes les actions applicables sont prises en considération
- Nécessité d'heuristiques



# Planification par recherche en chaînage avant

- Procède de l'état initial vers l'état final en utilisant les techniques d'exploration
- Le facteur de branchement est élevé
  - Toutes les actions applicables sont prises en considération
- Nécessité d'heuristiques
- Algorithme de recherche en chaînage avant : **déterministe**

# Planification par recherche en chaînage avant

- Procède de l'état initial vers l'état final en utilisant les techniques d'exploration
- Le facteur de branchement est élevé
  - Toutes les actions applicables sont prises en considération
- Nécessité d'heuristiques
- Algorithme de recherche en chaînage avant : **déterministe**
- Algorithme de recherche en chaînage avant : **correct** et **complet**

# Planification par recherche en chaînage avant

- Procède de l'état initial vers l'état final en utilisant les techniques d'exploration
- Le facteur de branchement est élevé
  - Toutes les actions applicables sont prises en considération
- Nécessité d'heuristiques
- Algorithme de recherche en chaînage avant : **déterministe**
- Algorithme de recherche en chaînage avant : **correct** et **complet**
  - Si le problème de planification a un plan solution, l'algorithme en chaînage avant va le retourner. Sinon, il retourne un échec.

# Algorithme de recherche en chaînage avant

---

**Algorithm 1:** RechAvant( $A, s_0, g$ )

---

**début**

**si**  $s_0$  satisfait  $g$  **alors retourner** un plan vide  $\pi$

**sinon**

$\text{active} \leftarrow \{a \mid a \text{ est une action de } A \text{ et } \text{precond}(a) \text{ est vrai dans } s_0\}$

**si**  $\text{active} = \emptyset$  **alors retourner** Echec

**sinon**

            Choisir\_action  $a_1 \in \text{active}$

$s_1 \leftarrow \gamma(s_0, a_1)$

$\pi \leftarrow \text{RechAvant}(A, s_1, g)$

**si**  $\pi \neq \text{Echec}$  **alors retourner**  $a_1.\pi$ ; **sinon retourner** Echec

**fin**

---

## Exemple : changer un pneu

- Actions :

    Action(Sortir(NvPneu,Coffre)

        PRECOND : Dans(NvPneu,Coffre)

        EFFET :  $\neg$  Dans(NvPneu,Coffre)  $\wedge$  Au(NvPneu,Sol)

    Action(Retirer(PneuPlat,Essieu)

        PRECOND : Sur(PneuPlat,Essieu)

        EFFET :  $\neg$  Sur(PneuPlat,Essieu)  $\wedge$  Au(PneuPlat,Sol)

- Action(Mettre(NvPneu,Essieu)

    PRECOND : Au(NvPneu,Sol)  $\wedge$  Au(PneuPlat,Sol)

    EFFET : Sur(NvPneu,Essieu)  $\wedge$   $\neg$  Au(NvPneu,Sol)

- Etat initial : Sur(PneuPlat, Essieu)  $\wedge$  Dans(NvPneu,Coffre)

- But : Sur(NvPneu,Essieu)

## Exemple : changer un pneu

- Etat initial :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Dans}(\text{NvPneu}, \text{Coffre})$

## Exemple : changer un pneu

- Etat initial :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Dans}(\text{NvPneu}, \text{Coffre})$ 
  - Action( $\text{Sortir}(\text{NvPneu}, \text{Coffre})$ )
    - PRECOND :  $\text{Dans}(\text{NvPneu}, \text{Coffre})$
    - EFFET :  $\neg \text{Dans}(\text{NvPneu}, \text{Coffre}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$

## Exemple : changer un pneu

- Etat initial :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Dans}(\text{NvPneu}, \text{Coffre})$ 
  - Action( $\text{Sortir}(\text{NvPneu}, \text{Coffre})$ )
    - PRECOND :  $\text{Dans}(\text{NvPneu}, \text{Coffre})$
    - EFFET :  $\neg \text{Dans}(\text{NvPneu}, \text{Coffre}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$



## Exemple : changer un pneu

- Etat initial :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Dans}(\text{NvPneu}, \text{Coffre})$ 
  - Action( $\text{Sortir}(\text{NvPneu}, \text{Coffre})$ )
    - PRECOND :  $\text{Dans}(\text{NvPneu}, \text{Coffre})$
    - EFFET :  $\neg \text{Dans}(\text{NvPneu}, \text{Coffre}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$
- Etat 2 :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$

## Exemple : changer un pneu

- Etat initial :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Dans}(\text{NvPneu}, \text{Coffre})$ 
  - Action( $\text{Sortir}(\text{NvPneu}, \text{Coffre})$ )
    - PRECOND :  $\text{Dans}(\text{NvPneu}, \text{Coffre})$
    - EFFET :  $\neg \text{Dans}(\text{NvPneu}, \text{Coffre}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$
- Etat 2 :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$ 
  - Action( $\text{Retirer}(\text{PneuPlat}, \text{Essieu})$ )
    - PRECOND :  $\text{Sur}(\text{PneuPlat}, \text{Essieu})$
    - EFFET :  $\neg \text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$

## Exemple : changer un pneu

- Etat initial :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Dans}(\text{NvPneu}, \text{Coffre})$ 
  - Action( $\text{Sortir}(\text{NvPneu}, \text{Coffre})$ )  
PRECOND :  $\text{Dans}(\text{NvPneu}, \text{Coffre})$   
EFFET :  $\neg \text{Dans}(\text{NvPneu}, \text{Coffre}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$
- Etat 2 :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$ 
  - Action( $\text{Retirer}(\text{PneuPlat}, \text{Essieu})$ )  
PRECOND :  $\text{Sur}(\text{PneuPlat}, \text{Essieu})$   
EFFET :  $\neg \text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$
- Etat 3 :  $\text{Au}(\text{NvPneu}, \text{Sol}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$

## Exemple : changer un pneu

- Etat initial :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Dans}(\text{NvPneu}, \text{Coffre})$ 
  - Action( $\text{Sortir}(\text{NvPneu}, \text{Coffre})$ )  
PRECOND :  $\text{Dans}(\text{NvPneu}, \text{Coffre})$   
EFFET :  $\neg \text{Dans}(\text{NvPneu}, \text{Coffre}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$
- Etat 2 :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$ 
  - Action( $\text{Retirer}(\text{PneuPlat}, \text{Essieu})$ )  
PRECOND :  $\text{Sur}(\text{PneuPlat}, \text{Essieu})$   
EFFET :  $\neg \text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$
- Etat 3 :  $\text{Au}(\text{NvPneu}, \text{Sol}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$ 
  - Action( $\text{Mettre}(\text{NvPneu}, \text{Essieu})$ )  
PRECOND :  $\text{Au}(\text{NvPneu}, \text{Sol}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$   
EFFET :  $\text{Sur}(\text{NvPneu}, \text{Essieu}) \wedge \neg \text{Au}(\text{NvPneu}, \text{Sol})$

## Exemple : changer un pneu

- Etat initial :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Dans}(\text{NvPneu}, \text{Coffre})$ 
  - Action( $\text{Sortir}(\text{NvPneu}, \text{Coffre})$ )  
PRECOND :  $\text{Dans}(\text{NvPneu}, \text{Coffre})$   
EFFET :  $\neg \text{Dans}(\text{NvPneu}, \text{Coffre}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$
- Etat 2 :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$ 
  - Action( $\text{Retirer}(\text{PneuPlat}, \text{Essieu})$ )  
PRECOND :  $\text{Sur}(\text{PneuPlat}, \text{Essieu})$   
EFFET :  $\neg \text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$
- Etat 3 :  $\text{Au}(\text{NvPneu}, \text{Sol}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$ 
  - Action( $\text{Mettre}(\text{NvPneu}, \text{Essieu})$ )  
PRECOND :  $\text{Au}(\text{NvPneu}, \text{Sol}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$   
EFFET :  $\text{Sur}(\text{NvPneu}, \text{Essieu}) \wedge \neg \text{Au}(\text{NvPneu}, \text{Sol})$
- Etat 3 :  $\text{Sur}(\text{NvPneu}, \text{Essieu}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$

## Exemple : changer un pneu

- Etat initial :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Dans}(\text{NvPneu}, \text{Coffre})$ 
  - Action( $\text{Sortir}(\text{NvPneu}, \text{Coffre})$ )  
PRECOND :  $\text{Dans}(\text{NvPneu}, \text{Coffre})$   
EFFET :  $\neg \text{Dans}(\text{NvPneu}, \text{Coffre}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$
- Etat 2 :  $\text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{NvPneu}, \text{Sol})$ 
  - Action( $\text{Retirer}(\text{PneuPlat}, \text{Essieu})$ )  
PRECOND :  $\text{Sur}(\text{PneuPlat}, \text{Essieu})$   
EFFET :  $\neg \text{Sur}(\text{PneuPlat}, \text{Essieu}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$
- Etat 3 :  $\text{Au}(\text{NvPneu}, \text{Sol}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$ 
  - Action( $\text{Mettre}(\text{NvPneu}, \text{Essieu})$ )  
PRECOND :  $\text{Au}(\text{NvPneu}, \text{Sol}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$   
EFFET :  $\text{Sur}(\text{NvPneu}, \text{Essieu}) \wedge \neg \text{Au}(\text{NvPneu}, \text{Sol})$
- Etat 3 :  $\text{Sur}(\text{NvPneu}, \text{Essieu}) \wedge \text{Au}(\text{PneuPlat}, \text{Sol})$
- But :  $\text{Sur}(\text{NvPneu}, \text{Essieu})$  satisfait dans cet état

# Amélioration du chaînage avant : planificateur "Fast Forward"

- "Fast Forward" s'appuie sur :
  - un algorithme spécifique de gradient appelé [Enforced Hill Climbing](#)
  - une heuristique qui s'appuie sur une étude d'atteignabilité
  - un agenda de buts
  - une heuristique pour élaguer les successeurs non prometteurs

# **Planification dans les espaces d'états**

---

**Planification par recherche en chaînage  
arrière**



- Procède de l'état final vers l'état initial

# Planification par recherche en chaînage arrière

- Procède de l'état final vers l'état initial
- Limite le facteur de branchement

# Planification par recherche en chaînage arrière

- Procède de l'état final vers l'état initial
- Limite le facteur de branchement
  - Habituellement, le but a peu d'opérateurs applicables

# Planification par recherche en chaînage arrière

- Procède de l'état final vers l'état initial
- Limite le facteur de branchement
  - Habituellement, le but a peu d'opérateurs applicables
  - Ne considère que les actions **pertinentes**

# Planification par recherche en chaînage arrière

- Procède de l'état final vers l'état initial
- Limite le facteur de branchement
  - Habituellement, le but a peu d'opérateurs applicables
  - Ne considère que les actions **pertinentes**
- Action pertinente pour le but sous forme conjonctive :

# Planification par recherche en chaînage arrière

- Procède de l'état final vers l'état initial
- Limite le facteur de branchement
  - Habituellement, le but a peu d'opérateurs applicables
  - Ne considère que les actions **pertinentes**
- Action pertinente pour le but sous forme conjonctive :
  - Accomplir l'un des termes de la conjonction

# Planification par recherche en chaînage arrière

- Procède de l'état final vers l'état initial
- Limite le facteur de branchement
  - Habituellement, le but a peu d'opérateurs applicables
  - Ne considère que les actions **pertinentes**
- Action pertinente pour le but sous forme conjonctive :
  - Accomplir l'un des termes de la conjonction
  - Ne pas annuler un des terme désiré

# Planification par recherche en chaînage arrière

- Procède de l'état final vers l'état initial
- Limite le facteur de branchement
  - Habituellement, le but a peu d'opérateurs applicables
  - Ne considère que les actions **pertinentes**
- Action pertinente pour le but sous forme conjonctive :
  - Accomplir l'un des termes de la conjonction
  - Ne pas annuler un des terme désiré
  - **action consistante**



# Planification par recherche en chaînage arrière

- Procède de l'état final vers l'état initial
- Limite le facteur de branchement
  - Habituellement, le but a peu d'opérateurs applicables
  - Ne considère que les actions **pertinentes**
- Action pertinente pour le but sous forme conjonctive :
  - Accomplir l'un des termes de la conjonction
  - Ne pas annuler un des terme désiré
  - **action consistante**
- Arrêt lorsqu'on obtient un sous-ensemble de littéraux qui satisfont l'état initial

# Planification par recherche en chaînage arrière

- Procède de l'état final vers l'état initial
- Limite le facteur de branchement
  - Habituellement, le but a peu d'opérateurs applicables
  - Ne considère que les actions **pertinentes**
- Action pertinente pour le but sous forme conjonctive :
  - Accomplir l'un des termes de la conjonction
  - Ne pas annuler un des terme désiré
  - **action consistante**
- Arrêt lorsqu'on obtient un sous-ensemble de littéraux qui satisfont l'état initial
- Nécessité d'heuristiques pour choisir l'action

# Planification par recherche en chaînage arrière

- Procède de l'état final vers l'état initial
- Limite le facteur de branchement
  - Habituellement, le but a peu d'opérateurs applicables
  - Ne considère que les actions **pertinentes**
- Action pertinente pour le but sous forme conjonctive :
  - Accomplir l'un des termes de la conjonction
  - Ne pas annuler un des terme désiré
  - **action consistante**
- Arrêt lorsqu'on obtient un sous-ensemble de littéraux qui satisfont l'état initial
- Nécessité d'heuristiques pour choisir l'action
- Algorithme de recherche en chaînage arrière : **correct** et **complet**

- Recherche dans les espaces d'états : recherche d'un **plan totalement ordonné**
- Séquence d'actions strictement linéaires
- Ordre chronologique de planification
- Une approche plus souple peut être nécessaire : résoudre différents sous-buts et combiner les sous-plans

⇒ Planification en ordre partiel

# Planification dans les espaces de plans

---

## Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**

## Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**
- Les nœuds sont des plans partiellement spécifiés

## Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**
- Les nœuds sont des plans partiellement spécifiés
- Les arcs sont des opérateurs de raffinement permettant de compléter le plan partiel



# Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**
- Les nœuds sont des plans partiellement spécifiés
- Les arcs sont des opérateurs de raffinement permettant de compléter le plan partiel
  - permettre d'atteindre un but encore non atteint

# Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**
- Les nœuds sont des plans partiellement spécifiés
- Les arcs sont des opérateurs de raffinement permettant de compléter le plan partiel
  - permettre d'atteindre un but encore non atteint
  - supprimer une incohérence dans le plan, etc.

# Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**
- Les nœuds sont des plans partiellement spécifiés
- Les arcs sont des opérateurs de raffinement permettant de compléter le plan partiel
  - permettre d'atteindre un but encore non atteint
  - supprimer une incohérence dans le plan, etc.
- La solution d'un plan change, on parle de plan partiel

# Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**
- Les nœuds sont des plans partiellement spécifiés
- Les arcs sont des opérateurs de raffinement permettant de compléter le plan partiel
  - permettre d'atteindre un but encore non atteint
  - supprimer une incohérence dans le plan, etc.
- La solution d'un plan change, on parle de plan partiel
- La planification est considérée comme un processus de raffinement de plans qui peut :

# Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**
- Les nœuds sont des plans partiellement spécifiés
- Les arcs sont des opérateurs de raffinement permettant de compléter le plan partiel
  - permettre d'atteindre un but encore non atteint
  - supprimer une incohérence dans le plan, etc.
- La solution d'un plan change, on parle de plan partiel
- La planification est considérée comme un processus de raffinement de plans qui peut :
  1. ajouter des actions

# Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**
- Les nœuds sont des plans partiellement spécifiés
- Les arcs sont des opérateurs de raffinement permettant de compléter le plan partiel
  - permettre d'atteindre un but encore non atteint
  - supprimer une incohérence dans le plan, etc.
- La solution d'un plan change, on parle de plan partiel
- La planification est considérée comme un processus de raffinement de plans qui peut :
  1. ajouter des actions
  2. ajouter des contraintes d'ordre entre les actions

# Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**
- Les nœuds sont des plans partiellement spécifiés
- Les arcs sont des opérateurs de raffinement permettant de compléter le plan partiel
  - permettre d'atteindre un but encore non atteint
  - supprimer une incohérence dans le plan, etc.
- La solution d'un plan change, on parle de plan partiel
- La planification est considérée comme un processus de raffinement de plans qui peut :
  1. ajouter des actions
  2. ajouter des contraintes d'ordre entre les actions
  3. ajouter des liens causaux

# Qu'est ce que la recherche dans un espace de plans ?

- L'espace de recherche n'est plus un espace d'états mais un **espace de plans**
- Les nœuds sont des plans partiellement spécifiés
- Les arcs sont des opérateurs de raffinement permettant de compléter le plan partiel
  - permettre d'atteindre un but encore non atteint
  - supprimer une incohérence dans le plan, etc.
- La solution d'un plan change, on parle de plan partiel
- La planification est considérée comme un processus de raffinement de plans qui peut :
  1. ajouter des actions
  2. ajouter des contraintes d'ordre entre les actions
  3. ajouter des liens causaux
  4. ajouter des contraintes d'instanciation sur les variables



- Stratégie du moindre engagement (*least commitment*) : se compromettre uniquement sur les aspects pertinents des actions, en laissant les autres aspects à plus tard

- Stratégie du moindre engagement (*least commitment*) : se compromettre uniquement sur les aspects pertinents des actions, en laissant les autres aspects à plus tard
- Un planificateur qui peut placer deux actions dans un plan sans spécifier laquelle intervient en premier est appelé **planificateur en ordre partiel**

- Stratégie du moindre engagement (*least commitment*) : se compromettre uniquement sur les aspects pertinents des actions, en laissant les autres aspects à plus tard
- Un planificateur qui peut placer deux actions dans un plan sans spécifier laquelle intervient en premier est appelé **planificateur en ordre partiel**
- La solution est présentée comme un graphe d'actions

- Stratégie du moindre engagement (*least commitment*) : se compromettre uniquement sur les aspects pertinents des actions, en laissant les autres aspects à plus tard
- Un planificateur qui peut placer deux actions dans un plan sans spécifier laquelle intervient en premier est appelé **planificateur en ordre partiel**
- La solution est présentée comme un graphe d'actions
- Linéarisation du plan : le planificateur représente un ordre entre toutes les étapes

## Exemple : chaussures et chaussettes

- Problème : mettre chaussures et chaussettes
- Actions :

Action(LeftSock())

*pas de paramètre*

PRECOND : BareLeftFoot

EFFET : LeftStockOn  $\wedge$   $\neg$ BareLeftFoot)

Action(RightSock())

PRECOND : BareRightFoot

EFFET : RightStockOn  $\wedge$   $\neg$ BareRightFoot)

Action(LeftShoe())

PRECOND : LeftStockOn

EFFET : LeftShoeOn)

Action(RightShoe())

PRECOND : RightStockOn

EFFET : RightShoeOn)

- Etat initial : BareLeftFoot  $\wedge$  BareRightFoot
- But : LeftShoeOn  $\wedge$  RightShoeOn

## Exemple : chaussures et chaussettes

- Le but peut être atteint de plusieurs façons différentes
  - chaussette droite, chaussure droite, chaussette gauche, chaussure gauche
  - l'inverse
  - les deux chaussettes puis les deux chaussures...

## Exemple : chaussures et chaussettes

- Le but peut être atteint de plusieurs façons différentes
  - chaussette droite, chaussure droite, chaussette gauche, chaussure gauche
  - l'inverse
  - les deux chaussettes puis les deux chaussures...
- L'idée est donc de représenter en *parallèle* les actions pouvant être effectuées dans un ordre quelconque

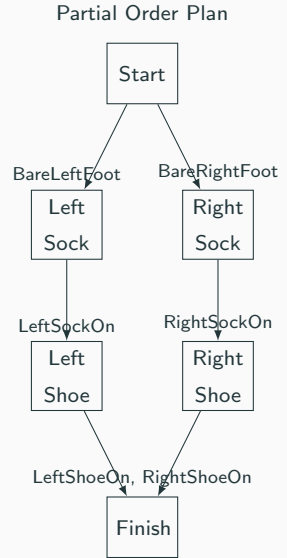
## Exemple : chaussures et chaussettes

- Le but peut être atteint de plusieurs façons différentes
  - chaussette droite, chaussure droite, chaussette gauche, chaussure gauche
  - l'inverse
  - les deux chaussettes puis les deux chaussures...
- L'idée est donc de représenter en *parallèle* les actions pouvant être effectuées dans un ordre quelconque
- et de conserver les liens temporels existants



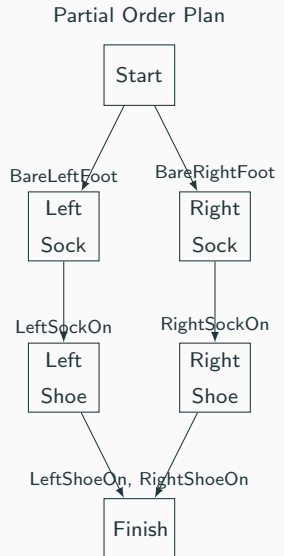
# Exemple : chaussures et chaussettes

- Le but peut être atteint de plusieurs façons différentes
  - chaussette droite, chaussure droite, chaussette gauche, chaussure gauche
  - l'inverse
  - les deux chaussettes puis les deux chaussures...
- L'idée est donc de représenter en *parallèle* les actions pouvant être effectuées dans un ordre quelconque
- et de conserver les liens temporels existants



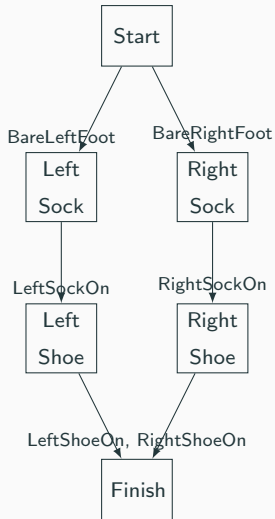
# Exemple : chaussures et chaussettes

- Le but peut être atteint de plusieurs façons différentes
  - chaussette droite, chaussure droite, chaussette gauche, chaussure gauche
  - l'inverse
  - les deux chaussettes puis les deux chaussures...
- L'idée est donc de représenter en *parallèle* les actions pouvant être effectuées dans un ordre quelconque
- et de conserver les liens temporels existants
- **Ce plan partiellement ordonné correspond à 6 plans totalement ordonnés**



# Exemple : chaussures et chaussettes

Partial Order Plan



Total Order Plans



# Composantes d'un plan partiellement ordonné

Un plan partiellement ordonné est un tuple  $\langle A, \prec, B, L \rangle$  où

- $A = \{a_1, \dots, a_n\}$  est un ensemble d'**actions**
  - Un plan vide contient uniquement les actions *départ* et *fin*
- $\prec$  est un ensemble de **contraintes d'ordre** entre les actions
  - Chaque contrainte spécifie le fait qu'une action doit survenir avant une autre
  - Si l'action  $a_i$  doit être exécutée avant  $a_j$ , on écrira  $a_i \prec a_j$
- $B$  est un ensemble de **contraintes d'instanciation** sur les variables de  $A$
- $L$  est un ensemble de **liens causaux** entre les actions, noté  $a_i \xrightarrow{p} a_j$ , tels que
  - $a_i, a_j \in A$  sont deux actions
  - $a_i \prec a_j$  est une contrainte d'ordre définie dans  $\prec$
  - Signifie que la proposition  $p$  est un effet de  $a_i$  et une précondition de  $a_j$

- Une solution est un **plan complet et consistant** que l'agent peut accomplir et qui garantit l'accomplissement du but
  - **Complet** : toutes les préconditions sont satisfaites par une action précédente, et ne sont pas annulées par une autre action
  - **Consistant** : Il n'y a pas de cycle dans les contraintes d'ordonnancement, et pas de conflits entre les liens causaux

- Une action  $a_k$  est une **menace** pour un lien causal  $a_i \xrightarrow{p} a_j$  ssi :
  - $a_k$  a un effet  $\neg q$  qui est potentiellement incohérent avec  $p$
  - les contraintes d'ordres  $a_i \prec a_k$  et  $a_k \prec a_j$  sont cohérentes avec  $\prec$
  - les contraintes d'instanciation résultant de l'unification de  $p$  et de  $q$  sont cohérentes avec  $B$
- Un **défaut** dans un plan partiellement ordonné est soit
  - un sous-but, c'est à dire la précondition d'une action de  $A$  qui n'est pas supportée par un lien causal
  - une menace, c'est à dire une action qui interfère avec un lien causal

## Exemple : changer un pneu

- Actions :

    Action(Sortir(NvPneu,Coffre)

        PRECOND : Dans(NvPneu,Coffre)

        EFFET :  $\neg$  Dans(NvPneu,Coffre)  $\wedge$  Au(NvPneu,Sol))

    Action(Retirer(PneuPlat,Essieu)

        PRECOND : Sur(PneuPlat,Essieu)

        EFFET :  $\neg$  Sur(PneuPlat,Essieu)  $\wedge$  Au(PneuPlat,Sol))

- Action(Mettre(NvPneu,Essieu)

        PRECOND : Au(NvPneu,Sol)  $\wedge$  Au(PneuPlat,Sol)

        EFFET : Sur(NvPneu,Essieu)  $\wedge$   $\neg$  Au(NvPneu,Sol))

- Etat initial : Sur(PneuPlat, Essieu)  $\wedge$  Dans(NvPneu,Coffre))

- But : Sur(NvPneu,Essieu)

## Exemple : changer un pneu

Début

Dans(NvPneu,Coffre)

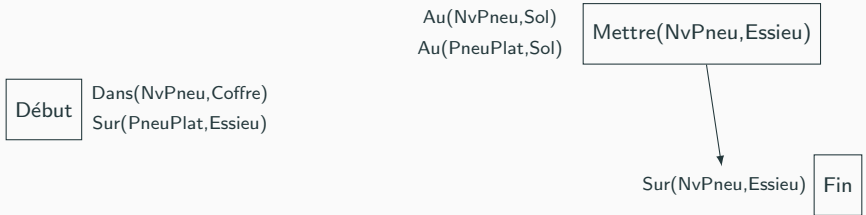
Sur(PneuPlat,Essieu)

Sur(NvPneu,Essieu)

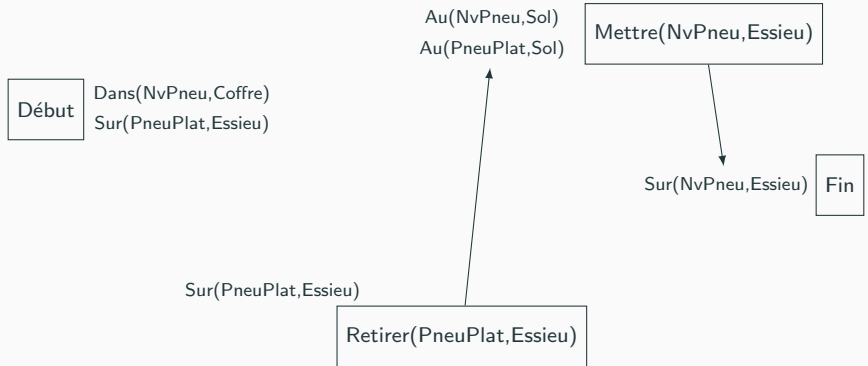
Fin



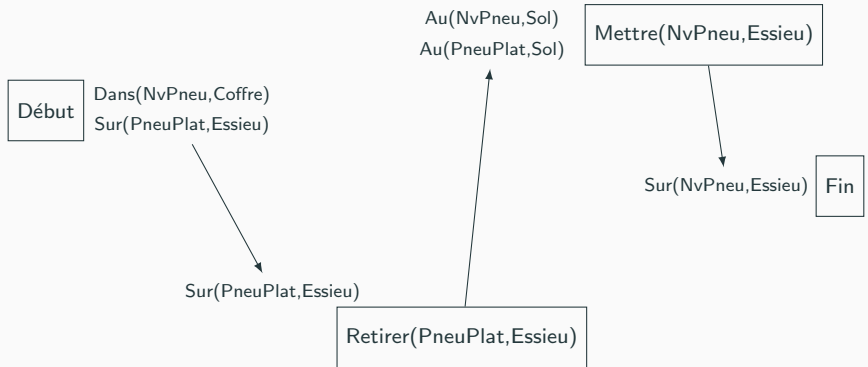
## Exemple : changer un pneu



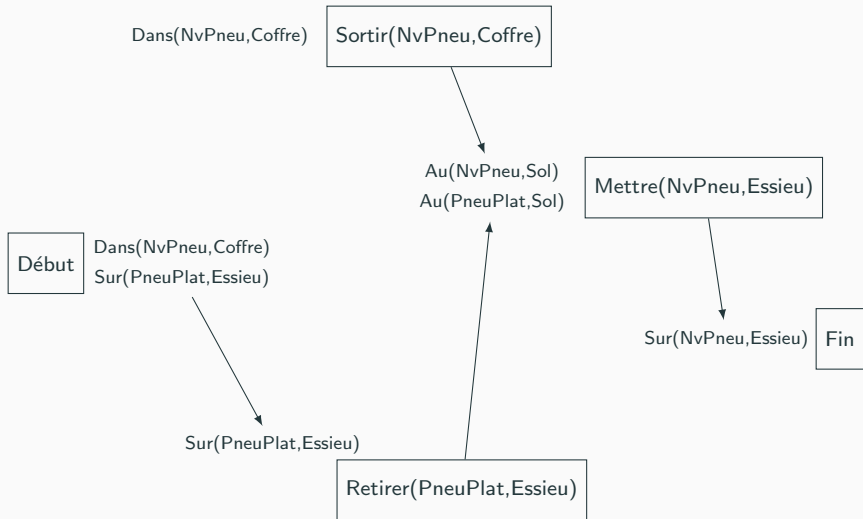
## Exemple : changer un pneu



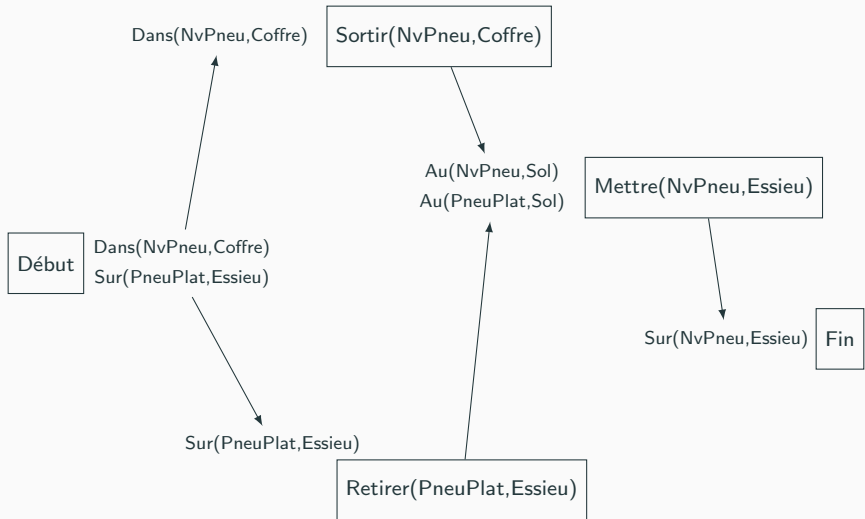
## Exemple : changer un pneu



## Exemple : changer un pneu



## Exemple : changer un pneu



# Conclusion

---

- Langage de représentation STRIPS
- Planification dans les espaces d'états
  - Produit des séquences d'actions
- Planification dans les espaces de plans
  - Produit des plans partiellement ordonnés
- Il existe aussi la *planification dans les graphes*
  - Produit des plans solution qui sont des séquences d'ensembles d'actions
    - Graphe de planification
    - Moins expressive que dans les espaces de plans, mais plus que dans les espaces d'états

# Extension de la représentation classique

- La représentation classique STRIPS est très limitée. Des extensions sont nécessaires pour décrire des problèmes intéressants.
- Les principales extensions sont :
  - Le typage des variables
  - Les opérateurs de planification conditionnels
  - Les expressions quantifiées
  - Les préconditions disjonctives
  - L'axiomatique et l'inférence
  - etc.
- Le langage de planification **PDDL (Planning Domain Description Language)** permet de prendre compte ces extensions