Représentation des Connaissances et Raisonnement

Introduction à la planification

Langage STRIPS, Progression, Régression,
Planification Partiellement Ordonnée

Julien Rossit
Julien.Rossit@u-paris.fr

(d'après Pavlos Moraitis)

Planification: généralités

- **Planification** c'est la tâche qui consiste à trouver une séquence d'actions qui réalisera un objectif

- La représentation des problèmes de planification concerne des **états**, des **actions** et des **objectifs**

- Le but est de trouver un langage qui est à la fois suffisamment expressif pour décrire une grande varieté de problèmes mais assez restrictif pour permettre à des algorithmes efficaces d'agir

- Nous allons considérer des environnements de **planification classique** (c.a.d. totalement observables, déterministes, finis, statiques -les changements n'étant dûs qu'aux seules actions des agents - et discrets-en temps, action, objets et effets)

- Le langage basique de représentation des planificateurs classiques est le **langage STRIPS** (Stanford Research Institute Problem Solver)

Le Langage STRIPS (*Stanford Research Institute Problem Solver*)

- Représentation des états: Les planificateurs décomposent le monde en conditions logiques et représentent un état comme une conjonction de littéraux positifs
  - On peut considérer des littéraux propositionnels : Pauvre$\wedge$Inconnu peut représenter l'état d'un agent infortuné
  - On peut considérer des littéraux du premier-ordre: Dans(Avion$_1$,Paris)$\wedge$Dans(Avion$_2$, Athènes) peuvent représenter un état du problème de livraison de paquets
  - Les littéraux du premier-ordre doivent être **instanciés (ground)** et **libres de fonctions**; des littéraux comme Sur(x,y) ou habite(Pére(Paul), Paris) ne sont pas permis
  - L'hypothèse du **monde-clos** est utilisé; cela signifie que toute condition non mentionnée dans un état est considérée fausse.

Le Langage STRIPS (2)

- Représentation des objectifs: un objectif est un état partiellement spécifié, représenté comme une conjonction des littéraux instanciés positifs comme Riche$\wedge$Reputé ou Dans(Avion$_1$, Paris)
  - Un état propositionnel *s satisfait* un objectif *g* si *s* contient tous les atomes dans *g* (et possible d'autres); l'état Riche$\wedge$Reputé$\wedge$Misérable satisfait l'objectif Riche$\wedge$Reputé

- Représentation des actions: une action est spécifiée en termes de préconditions qui doivent être valables avant qu'elle puisse être exécutée et en terme d'effets qui s'ensuivent quand elle est exécutée
  - Une action pour faire voler un avion d'un endroit à un autre est:

    Action(Voler(p, de, à),

    PRECOND: $A(p, de) \wedge Avion(p) \wedge Aéroport(de) \wedge Aéroport(à)$

    EFFET: $\neg A(p, de) \wedge A(p, à))$
  - Nous appelons cela un **schéma d'action**, signifiant qu'il représente un nombre d'actions différentes qui peuvent être dérivées en instanciant *p, de*, et *à,* à des constantes différentes.

Le Langage STRIPS (3)

- Plus généralement un shéma d'action est constitué de trois parties:
  - Le **nom de l'action** et une **liste de paramètres**-p. ex. Voler(p, de, à)-sert à identifier l'action
  - La **précondition** est une conjonction de littéraux positifs, libres de fonctions, faisant état de ce qui doit être vrai dans un état avant que l'action puisse être exécutée. Toute variable apparaissant dans la précondition doit aussi apparaître dans la liste de paramètres de l'action
  - L'**effet** est une conjonction de littéraux, libres de fonctions décrivant comment l'état change quand l'action est exécutée. Un littéral positif P dans l'effet est supposé être vrai dans l'état résultant de l'action, tandis qu'un littéral négatif ¬P être faux. Les variables dans les effets doivent aussi apparaître dans la liste de paramètres de l'action
  - Certains systèmes de planification divisent les effets en **liste d'addition (add list)** pour les littéraux positifs et en **liste de suppression (delete list)** pour les littéraux négatifs

Le Langage STRIPS (4)

- Ayant défini la syntaxe pour la représentation des problèmes de planification nous pouvons maintenant définir la sémantique:
  - La manière la plus directe est de définir comment les actions affectent les états
  - Une action est **applicable** à chaque état qui satisfait la précondition; autrement l'action n'a pas d'effet
  - Pour un shéma d'action de premier-ordre, établir l'applicabilité impliquera une substitution θ pour les variables dans la précondition
    - Par exemple considérons que l'état courant est décrit par:
      $Dans(P_1, JFK) \wedge Dans(P_2, CDG) \wedge Avion(P_1) \wedge Avion(P_2) \wedge Aéroport(JFK) \wedge Aéroport(CDG)$
    - Cet état satisfait la précondition
      $Dans(p, de) \wedge Avion(p) \wedge Aéroport(de) \wedge Aéroport(à)$
      avec la substitution $\{p/P_1, de/JFK, à/CDG\}$ (entre autres)
    - Alors l'action concrète $Voler(P_1, JFK, CDG)$ est applicable
  - En commençant à l'état **s**, le **résultat** de l'exécution une action applicable α est un état **s'** qui est le même que s excepté le fait que chaque littéral positif P dans l'effet de α est ajouté dans s' et que chaque littéral négatif $\neg$P est retiré de s'
    - Après $Voler(P_1, JFK, CDG)$ l'état courant devient $Dans(P_1, CDG) \wedge Dans(P_2, CDG) \wedge Avion(P_1) \wedge Avion(P_2) \wedge Aéroport(JFK) \wedge Aéroport(CDG)$

Le Langage STRIPS (5)

- Il faut noter que si un effet positif est déjà dans s celui-ci n'est pas ajouté deux fois et si un effet négatif n'est pas dans s alors cette partie de l'effet est ignorée

- Cette définition incorpore l'**hypothèse de STRIPS**: chaque littéral non mentionné dans l'effet reste inchangé. De cette façon STRIPS évite le **frame problem** (c.a.d. représenter toutes les choses qui restent les mêmes)

- La **solution** d'un problème de planification consiste (dans sa forme la plus simple) en une séquence d'actions qui quand elle est exécutée à l'état initial, a pour résultat un état où l'objectif est satisfait

- La notation STRIPS est adéquate pour plusieurs domaines réels. Cependant elle ne peut pas résoudre de façon naturelle les **ramifications** des actions
  - S'il existe des personnes, des paquets ou des bagages dans l'avion alors ils doivent aussi changer de position quand l'avion vole.
  - Nous pouvons représenter ces changements comme les effets directs du vol, tandis qu'il semble plus naturel de représenter la position du contenu de l'avion comme une conséquence logique de la position de l'avion
  - Les systèmes de planification classique ne peuvent aussi résoudre le problème de **qualification** (c.a.d. le problème de circonstances non représentées qui pourraient causer l'échec de l'action)

Exemple 1: Transport de Frêt Aérien (1)

- Problème de transport de frêt aérien impliquant chargement et déchargement de frêt (cargo), et d'avions et emmenant celui-ci d'une place à l'autre.

- Le problème peut être défini avec trois actions: *Charger, Décharger* et *Voler*

- Les actions affectent deux prédicats: *Dans(f,p)* signifie que le frêt f est dans l'avion *p* et *A(x,α)* signifie que l'objet *x* (soit avion soit frêt) est à l'aéroport *α*

- Le frêt n'est *A* nulle part quand il est *Dans* un avion, et par conséquent *A* signifie "disponible pour utilisation à une position donnée"

Exemple 1: Transport de Frêt Aérien (2)

Init($A(F_1,CDG) \wedge A(F_2, JFK) \wedge A(P_1, CDG) \wedge A(P_2, JFK) \wedge Frêt(F_1) \wedge Frêt(F_2) \wedge Avion(P_1) \wedge Avion(P_2) \wedge Aéroport(CDG) \wedge Aéroport(JFK)$)

Objectif($A(F_1, JFK) \wedge A(F_2, CDG)$)

Action(Charger(f, p, α),

PRECOND: $A(f, \alpha) \wedge A(p, \alpha) \wedge Frêt(f) \wedge Avion(p) \wedge Aéroport(\alpha)$

EFFET: $\neg A(f, \alpha) \wedge Dans(f, p)$)

Action(Décharger(f, p, α),

PRECOND: $Dans(f, p) \wedge A(p, \alpha) \wedge Frêt(f) \wedge Avion(p) \wedge Aéroport(\alpha)$

EFFET: $A(f, \alpha) \wedge \neg Dans(f, p)$)

Action(Voler(p, de, à),

PRECOND: $A(p, de) \wedge Avion(p) \wedge Aéroport(de) \wedge Aéroport(à)$

EFFET: $\neg A(p, de) \wedge A(p, à)$)

Solution: [Charger(F1, P1, CDG), Voler(P1, CDG, JFK), Décharger(F1, P1, JFK), Charger(F2, P2, JFK), Voler(P2, JFK, CDG), Décharger(F2, P2, CDG)]

Exemple 2: le Monde de Blocs (1)

- C'est le plus fameux exemple des domaines de planification
- Il consiste en un ensemble de blocs, de forme cubique posés sur une table
- Les blocs peuvent s'entasser les uns sur les autres, mais seulement un bloc peut être mis directement sur un autre bloc
- Un bras de robot peut tenir seulement un bloc à la fois, donc il ne peut pas tenir un bloc qui a un autre bloc sur lui
- L'objectif sera toujours de bâtir une ou plusieurs piles de blocs, spécifiées en termes de quels blocs sont au dessus d'autres blocs
  - Par exemple un objectif pourrait être de poser le bloc A sur B et le bloc C sur D

Exemple 2: le Monde de Blocs (2)

- Nous utiliserons *Sur(b, x)* pour indiquer que le bloc b est sur x, où x est un autre bloc ou la table
- L'action pour déplacer le bloc *b* du dessus de *x* au dessus de *y* sera *Déplacer(b, x, y)*
  - Une des préconditions pour déplacer b est qu'il n'y ait pas un autre bloc sur lui
  - Nous pouvons introduire un prédicat Dégagé(x) qui est vrai quand x n'a rien sur lui
- L'action *Déplacer*, déplace un bloc *b* de *x* à *y* si les deux (b et y) sont dégagés. Après que le déplacement soit fait, x est dégagé mais y ne l'est plus

Action(Déplacer(b, x, y),

PRECOND: Sur(b, x) $\wedge$ Dégagé(b) $\wedge$ Dégagé(y)

EFFET: Sur(b, y) $\wedge$ Dégagé(x) $\wedge \neg$Sur(b, x) $\wedge \neg$Dégagé(y))

Exemple 2: le Monde de Blocs (3)

- Cette action ne maintient pas Dégagé proprement quand x ou y est la table
- Quand *x=Table* cette action a comme effet *Dégagé(Table)* mais la table ne doit pas être dégagée et quand *y=Table*, elle a la précondition *Dégagé(Table)* mais la table n'a pas à être dégagée pour poser un bloc sur elle
- Pour corriger ce problème nous faisons deux choses:
  - Nous introduisons une autre action, *DéplacerSurTable(b, x)*, pour déplacer un bloc b de x à la table
  - Nous considérons l'interprétation de *Dégagé(b)* comme "il existe un espace libre sur b pour placer un bloc". Sous cette interprétation Dégagé(Table) sera toujours vrai
  - Cependant rien ne peut empêcher le planificateur d'utiliser Déplacer(b, x, Table) au lieu de *DéplacerSurTable(b, x)*

Action(DéplacerSurTable(b, x),

PRECOND: Sur(b, x) $\wedge$ Dégagé(b)

EFFET: Sur(b, Table) $\wedge$ Dégagé(x) $\wedge \neg$Sur(b, x))

Exemple 2: le Monde de Blocs (4)

Init(Sur(A, Table) $\wedge$ Sur(B, Table) $\wedge$ Sur(C, Table) $\wedge$ Bloc(A) $\wedge$ Bloc(B) $\wedge$ Bloc(C) $\wedge$ Dégagé(A) $\wedge$ Dégagé(B) $\wedge$ Dégagé(C))

Objectif(Sur(A, B) $\wedge$ Sur(B, C))

Action(Déplacer(b, x, y),

PRECOND: Sur(b, x) $\wedge$ Dégagé(b) $\wedge$ Dégagé(y) $\wedge$ Bloc(b) $\wedge$ (b$\neq$x) $\wedge$ (b$\neq$y) $\wedge$ (x$\neq$y)

EFFET: Sur(b, y) $\wedge$ Dégagé(x) $\wedge$ $\neg$Sur(b, x) $\wedge$ $\neg$Dégagé(y))

Action(DéplacerSurTable(b, x),

PRECOND: Sur(b, x) $\wedge$ Dégagé(b) $\wedge$ Bloc(b) $\wedge$ (b$\neq$x)

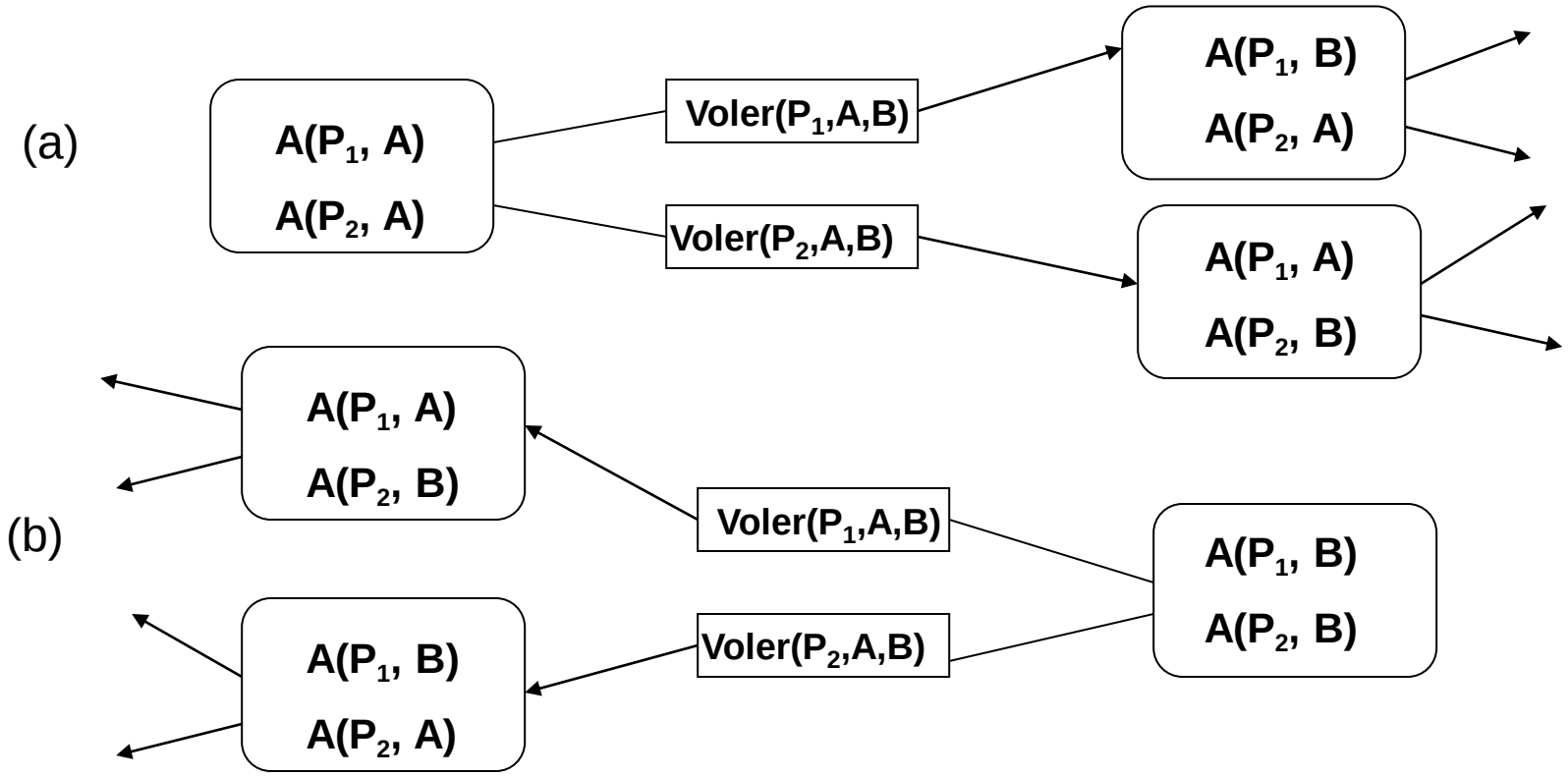
EFFET: Sur(b, Table) $\wedge$ Dégagé(x) $\wedge$ $\neg$Sur(b, x))

Une solution possible est: [Déplacer(B,Table, C), Déplacer(A, Table, B)]

Planification comme Recherche dans Espace-d'Etats (State-Space)

- L'approche la plus directe est l'utilisation de state-space recherche

- Puisque les descriptions des actions dans un problème de planification spécifient les deux, à savoir les préconditions et les effets, il est possible de rechercher dans les deux directions: soit chaînage avant (forward) à partir de l'état initial, soit chaînage arrière (backward) à partir de l'objectif



Recherche par Chaînage Avant dans Espace-d'Etats (State-Space)

- Cette approche est appelée **planification progressive** (progressive planning) puisqu'elle se déplace vers l'avant

- Nous commencons à l'état initial du problème en considérant des séquences d'actions jusqu'à ce qu'on trouve une séquence qui satisfait un état objectif. La formulation des problèmes de planification comme problèmes de recherche dans l'espace d'états, est comme suit:
  - L'**état initial** de la recherche est l'état initial du problème de planification; chaque état sera un ensemble de littéraux positifs instanciés; des littéraux qui n'apparaissent pas sont considérés faux
  - Les **actions** qui sont applicables dans un état sont celles dont les préconditions sont satisfaites; L'état suivant résultant d'une action est généré en ajoutant les littéraux de l'effet positif et en supprimant les littéraux de l'effet négatif
  - Le **test d'objectif** (goal test) contrôle si l'état satisfait l'objectif du problème de planification
  - Le **coût de l'étape** (step cost) de chaque action est typiquement 1 (même s'il est possible de considérer des coûts différents)

Recherche par Chaînage Arrière dans Espace-d'Etats (State-Space)

- Cette approche est appelée **planification régressive** (regression planning) puisqu'elle se déplace vers l'arrière
- L'avantage principal de cette approche est qu'elle permet de considérer seulement les actions pertinentes. Une action est pertinente pour un objectif conjonctif si elle accomplit un des conjoints de l'objectif
- Il faut noter que des actions non pertinentes peuvent aussi conduire à l'état objectif. Une recherche par chaînage arrière qui permet des actions non pertinentes sera toujours complète mais elle sera moins efficace
- La question principale en planification régressive est: quels sont les états à partir desquels l'application d'une action donnée peut conduire à l'objectif
- Calculer la description de ces états est appelée régresser l'objectif par cette action

Recherche par Chaînage Arrière dans Espace-d'Etats (State-Space)

- Considérons l'exemple du frêt aérien et soit l'objectif:
  $A(F_1,B) \wedge A(F_2, B) \wedge A(F_3, B) \ldots A(F_{20}, B)$ et l'action pertinente Décharger(F_1, p, B) qui réalise le premier conjoint
  - L'action va être exécutée seulement si ses préconditions sont satisfaites. Par conséquent chaque état prédécesseur doit inclure ces préconditions: $Dans(F_1, p) \wedge A(p, B)$
  - Le sous-goal $A(F_1, B)$ ne doit pas être vrai à l'état prédécesseur. La description du prédécesseur sera donc: $Dans(F_1, p) \wedge A(p, B) \wedge A(F_2, B) \wedge A(F_3, B) \ldots A(F_{20}, B)$
  - Les actions qui accomplissent quelques littéraux désirés, ne doivent pas détruire d'autres littéraux désirés. Une action qui satisfait cette restriction est appelée **cohérente** (consistent). L'action Charger(F_2, P) ne serait pas cohérente puisque elle produirait la négation du littéral $A(F_2, B)$

Recherche par Chaînage Arrière dans Espace-d'Etats (State-Space)

- Etant données les définitions de pertinence et cohérence nous pouvons décrire le processus général de construction de prédécesseurs pour une recherche en chaînage arrière.

- Etant donnée une description d'objectif G, soit A une action pertinente et cohérente. Le prédécesseur correspondant est comme suit:
  - Chacun des effets positifs de A apparaissant dans G est détruit
  - Chaque littéral de précondition de A est ajouté à moins qu'il apparaisse déjà

- Tous les algorithmes classiques de recherche peuvent être utilisés pour effectuer la recherche

- La fin est atteinte quand est générée une description de prédécesseur qui est satisfaite par l'état initial du problème de planification

Planification Partiellement Ordonnée (1)

- La recherche dans l'espace des états par chaînage avant et arrière sont des formes de recherche de plans **totalement ordonnés**
- Elles exploitent seulement des séquences strictement linéaires d'actions, directement connectées au début ou à l'objectif
- Cela signifie qu'elles ne peuvent pas tirer profit de la décomposition du problème. Au lieu de travailler sur chaque problème séparément, elles doivent toujours prendre des décisions sur comment créer une séquence d'actions par tous les sous-problèmes
- Il est préférable d'avoir une approche qui travaille sur différents sous-problèmes indépendamment, qui les résout avec des sous-plans différents et qui ensuite les combine
- Une telle approche a aussi l'avantage de souplesse dans l'ordre dans lequel elle construit le plan

Planification Partiellement Ordonnée (2)

- La stratégie générale pour retarder un choix durant la recherche est appelé stratégie du **plus petit engagement** (*least commitment*)

- Le plus petit engagement est un concept utile pour analyser quand des décisions devraient être prises à chaque problème de recherche

- Considérons le simple problème de mise d'une paire de chaussures. Nous pouvons décrire ce problème comme un problème formel de planification comme suit:

  Objectif*(ChaussureDroiteMise∧ChaussureGaucheMise)*

  Init*()*

  Action(ChaussureDroite*, PRECOND:ChaussetteDroiteMise, EFFET:ChaussureDroiteMise)*

  Action(ChaussetteDroite*, EFFET:ChaussetteDroiteMise)*

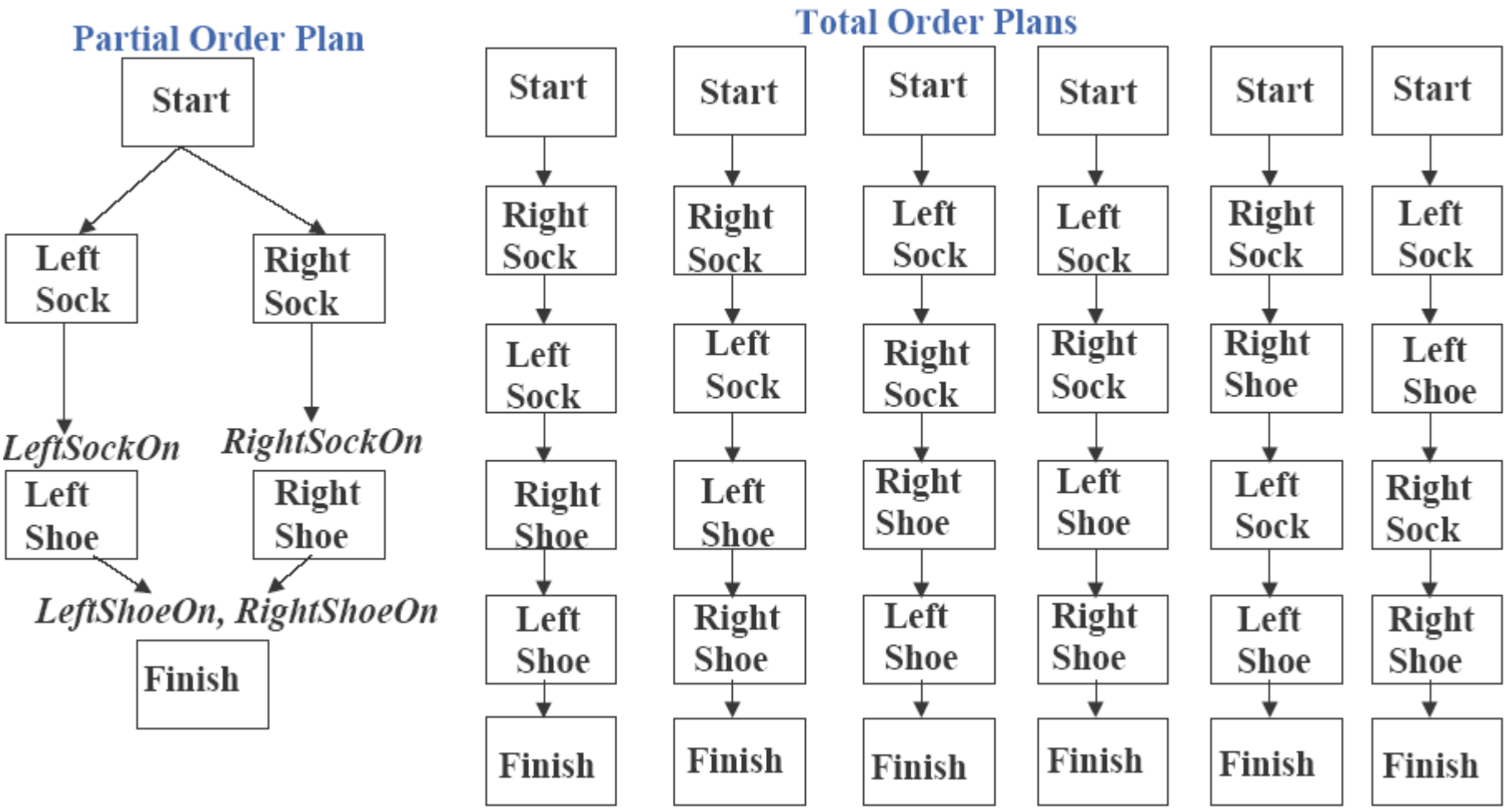
  Action(ChaussureGauche*, PRECOND:ChaussetteGaucheMise, EFFET:ChaussureGaucheMise)*

  Action(ChaussetteGauche*, EFFET:ChaussetteGaucheMise)*

Planification Partiellement Ordonnée (3)

- Un planificateur peut réaliser l'objectif en exécutant la séquence de deux actions *ChaussetteDroite* suivie de *ChaussureDroite* pour accomplir le premier conjoint de l'objectif et la séquence *ChaussetteGauche* suivie de *ChaussureGauche* pour le second conjoint
- Les deux séquences peuvent être combinées pour produire le plan final
- Le planificateur manipulera les deux sous-séquences indépendamment, sans s'engager sur le fait qu'une action dans une séquence sera avant ou après une action dans l'autre
- Tout algorithme de planification, qui peut placer deux actions dans un plan sans spécifier quelle action vient en premier, s'appelle **planificateur d'ordre partiel** (partial-order planner)
- Une solution de plans partiellement-ordonnés peut correspondre à plusieurs plans totalement-ordonnés. Chacun d'eux s'appelle **linéarisation** de plans partiellement-ordonnés

Planification Partiellement Ordonnée (4)



Planification Partiellement Ordonnée (5)

- La planification partiellement-ordonnée peut être implémentée comme une recherche dans **l'espace de plans** partiellement-ordonnés
- Nous commençons avec un plan vide et après nous cherchons comment raffiner le plan jusqu'à ce qu'on arrive à un plan complet qui résout le problème
- Les actions dans cette recherche ne sont pas des actions dans le monde mais des actions sur des plans, à savoir ajouter un pas dans un plan, imposer un ordre qui met une action avant une autre, etc.
- Différentes méthodes heuristiques de recherche peuvent être appliquées quand le problème de recherche est formulé
- Les états du problème de recherche seront des plans non finis. Pour éviter la confusion avec les états du monde nous parlerons par la suite de plans et non d'états

Planification Partiellement Ordonnée (6)

- Chaque plan a les quatre composants suivants, où les deux premiers définissent les pas de plan et les deux derniers servent une fonction de comptabilité pour déterminer comment les plans peuvent être étendus
- Un ensemble **d'actions** qui forment les pas du plan. Elles sont récupérées par l'ensemble des actions dans le problème de planification
  - Le plan vide contient juste les actions *Début* et *Fin*
  - *Début* n'a pas de préconditions et a comme effets tous les littéraux de l'état initial du problème de planification
  - *Fin* n'a pas d'effets et a comme préconditions les littéraux de l'objectif du problème de planification
- Un ensemble de **contraintes d'ordre**. Chaque contrainte d'ordre est de la forme $A \prec B$ ce qui est lu "A avant B" et signifie que l'action A doit être exécutée avant B mais pas nécessairement juste avant A
  - Les contraintes d'ordre doivent décrire un ordre partiel propre
  - Chaque cycle comme $A \prec B$ et $B \prec A$ représentent une contradiction. Donc une contrainte d'ordre ne peut être rajoutée dans le plan si elle crée un cycle

Planification Partiellement Ordonnée (7)

- Un ensemble de **liens causaux**. Un lien causal entre deux actions A et B dans un plan, est écrit $A \xrightarrow{p} B$ et lu “A **accomplie** p pour B”

  ChaussetteDroiteMise

  – Le lien causal *ChaussetteDroite* → *ChaussureDroite* montre que *ChaussetteDroiteMise* est un effet de l’action *ChaussetteDroite* et une précondition de *ChaussureDroite*

  – Il montre aussi que *ChaussetteDroiteMise* doit rester vrai du temps de l’action *ChaussetteDroite* jusqu’au temps de l’action *ChaussetteDroite*

  – Le plan ne doit pas être étendu en ajoutant une nouvelle action C en **conflit** avec le lien causal

  – Une action C est en conflit avec $A \xrightarrow{p} B$ si *C* a un effet $\neg p$ et si *C* (selon les contraintes d’ordre) peut arriver après *A* et avant *B*

  – Les liens causaux sont appelés parfois **intervalles de protection**, parce-que le lien $A \xrightarrow{p} B$ protège *p* d’être nié dans l’intervalle A à B

Planification Partiellement Ordonnée (8)

- Un ensemble des **préconditions ouvertes**. Une précondition est ouverte si elle n'est pas satisfaite par une action dans le plan
  - Les planificateurs essayeront de réduire l'ensemble de préconditions ouvertes à l'ensemble vide sans introduire une contradiction

- *Actions:{ChaussetteDroite,ChaussureDroite,ChaussetteGauche, ChaussureGauche,Début,Fin}*
  Ordres:{ChaussetteDroite $\prec$ ChaussureDroite, ChaussetteGauche $\prec$ ChaussureGauche}
  Liens:{ChaussetteDroite $\xrightarrow{ChaussetteDroiteMise}$ ChaussureDroite, ChaussetteGauche $\xrightarrow{ChaussetteGaucheMise}$ ChaussureGauche, ChaussureDroite $\xrightarrow{ChaussureDroiteMise}$ Fin, ChaussureGauche $\xrightarrow{ChaussureGaucheMise}$ Fin}
  Préconditions ouvertes:{ }

Planification Partiellement Ordonnée (9)

- Un plan **cohérent** est un plan dans lequel il n'existe pas de cycles dans les contraintes d'ordre et de conflits avec des liens causaux
- Un plan sans préconditions ouvertes est une **solution**
- Le problème de recherche que la PPO résout peut être formulé comme suit. Nous allons considérer une formulation appropriée pour des problèmes de planification propositionnels
  - Le plan initial contient Début et Fin, les contraintes d'ordre Début$\prec$Fin, pas de liens causaux et a toutes les préconditions dans Fin comme des préconditions ouvertes
  - La fonction du successeur arbitrairement pique une précondition ouverte *p* sur une action B et génère un plan successeur pour chaque moyen cohérent de sélection d'une action A qui accomplie *p*. La cohérence est renforcée comme suit
    1. Le lien causal $A \xrightarrow{p} B$ et la contrainte d'ordre A$\prec$B sont ajoutés dans le plan. L'action A peut être une action existante dans le plan ou une nouvelle. Si elle est nouvelle, ajoute elle dans le plan avec aussi Début$\prec$A et A$\prec$Fin

Planification Partiellement Ordonnée (10)

2. Nous résolvons les conflits entre le nouveau lien causal et toutes les actions existantes et entre l'action A (si elle nouvelle) et tous les liens causaux existants. Un conflit entre $A \xrightarrow{p} B$ et C est résolu en faisant intervenir C à un temps en dehors de l'intervalle de protection, en ajoutant $B \prec C$ ou $C \prec A$. Nous ajoutons des états successeurs pour l'un de deux ou pour les deux si ils ont comme résultat des plans cohérents

- Le test d'objectif contrôle quand un plan est une solution pour le problème original de planification. Puisqu' ils ne sont générés que de plans cohérents, le test d'objectif a juste besoin de contrôler qu'il n'existe pas de préconditions ouvertes

- Rappel: les actions considérées par l'algorithme de recherche sous cette formulation, sont des pas de raffinement du plan plutôt que des actions réelles du domaine lui-même
  - Le coût du chemin est sans importance puisque la seule chose qui intéresse est le coût total des actions réelles dans le plan, auquel le chemin emmène.

Exemple: Changement de Pneu

Init(Sur(PneuAPlat, Essieu) $\wedge$ Dans(PneuDeRechange, Huche))
Objectif(Sur(PneuDeRechange,Essieu))

Action(Rétirer(PneuDeRechange,Huche),
PRECOND: Dans(PneuDeRechange,Huche)
EFFET: $\neg$Dans(PneuDeRechange, Huche)$\wedge$Au(PneuDeRechange,Sol))

Action(Rétirer(PneuPlat, Essieu),
PRECOND: Sur(PneuAPlat, Essieu)
EFFET: $\neg$Sur(PneuAPlat, Essieu) $\wedge$Au(PneuAPlat,Sol))

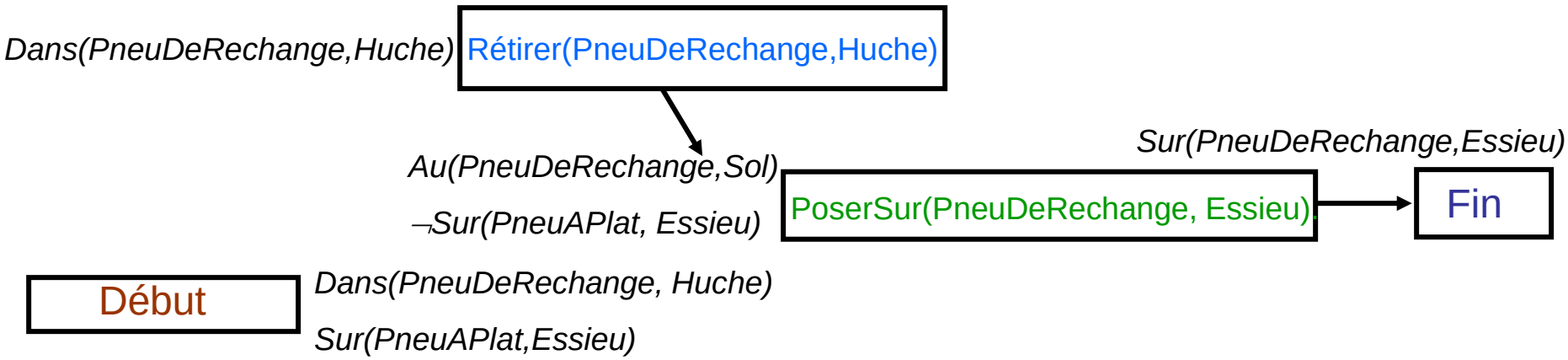
Action(PoserSur(PneuDeRechange, Essieu),
PRECOND: Au(PneuDeRechange,Sol)$\wedge\neg$Sur(PneuAPlat, Essieu)
EFFET: $\neg$Au(PneuDeRechange,Sol)$\wedge$Sur(PneuDeRechange,Essieu))

Action(PartirPendantLaNuit,
PRECOND:
EFFET: $\neg$Au(PneuDeRechange,Sol)$\wedge\neg$Sur(PneuDeRechange,Essieu)$\wedge$
$\neg$Dans(PneuDeRechange, Huche)$\wedge\neg$Au(PneuAPlat,Sol)$\wedge\neg$
Sur(PneuAPlat, Essieu)

Solution (1)

- La recherche d'une solution avec PPO commence avec le plan initial, contenant une action *Début* avec effet *Sur(PneuAPlat,Essieu)* $\wedge$*Dans(PneuDeRechange, Huche)* et une action *Fin* avec la seule précondition *Sur(PneuDeRechange,Essieu)*

- Nous générons des successeurs en prenant une précondition ouverte pour travailler avec et en choisissant une parmi les actions possibles pour la satisfaire. La séquence des événements est comme suit:
  1. Prendre la seule précondition ouverte, *Sur(PneuDeRechange,Essieu)* de *Fin*. Choisir la seule action applicable, *PoserSur(PneuDeRechange, Essieu)*
  2. Prendre la précondition *Au(PneuDeRechange,Sol)* de *PoserSur(PneuDeRechange, Essieu)*. Choisir la seule action applicable *Rétirer(PneuDeRechange,Huche)* pour la satisfaire



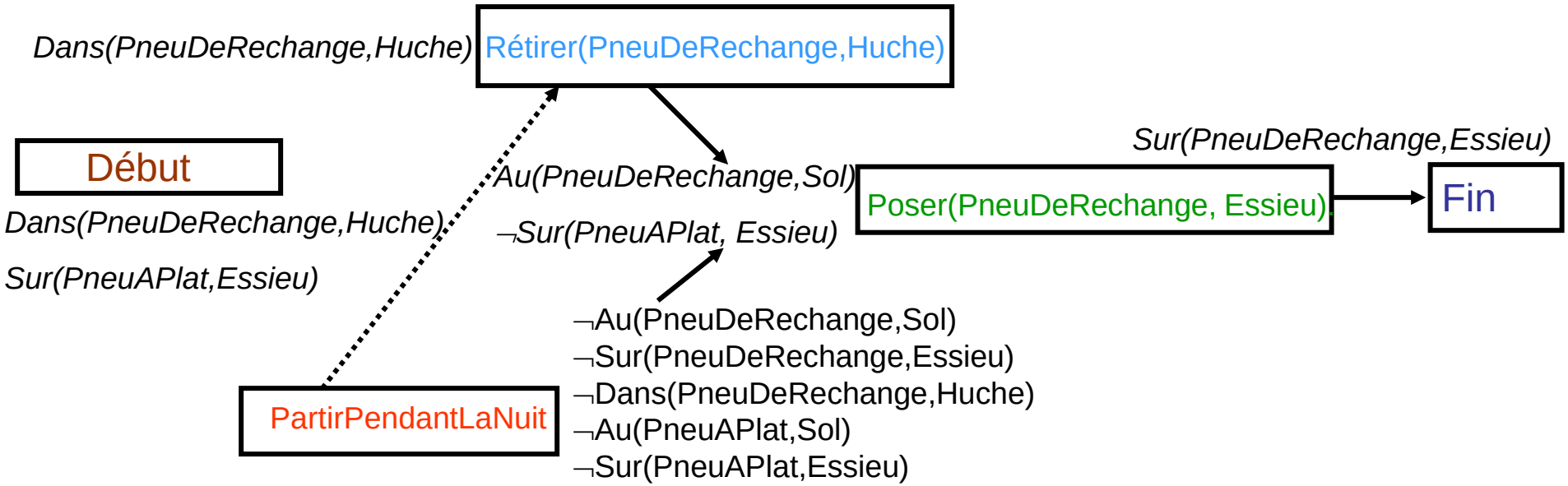
Solution (2)

3. Prendre la précondition ¬*Sur(PneuAPlat, Essieu)* de *PoserSur(PneuDeRechange, Essieu)*. Juste pour pour aller à contre sens, choisissons l'action *PartirPendantLaNuit* au lieu de l'action *Rétirer(PneuPlat, Essieu).* Notons que *PartirPendantLaNuit* a aussi comme effet ¬*Au(PneuDeRechange,Sol)* qui est en conflit avec avec le lien causal

Au(PneuDeRechange,Sol)

Rétirer(PneuDeRechange,Huche)→ PoserSur(PneuDeRechange, Essieu)

Pour résoudre ce conflit nous ajoutons une contrainte d'ordre en mettant *PartirPendantLaNuit* avant *Rétirer(PneuDeRechange,Huche)*.



Solution (3)

4. La seule précondition ouverte à ce point est *Dans(PneuDeRechange,Huche)* de l'action *Rétirer(PneuDeRechange,Huche)*

   – La seule action qui peut la satisfaire est l'action *Début,* mais le lien causal du *Début* à *Rétirer(PneuDeRechange,Huche)* est en conflit avec l'effet ¬*Dans(PneuDeRechange, Huche)* de l'action *PartirPendantLaNuit*

   – Cette fois ci il n'y a pas de manière de résoudre le conflit avec *PartirPendantLaNuit*

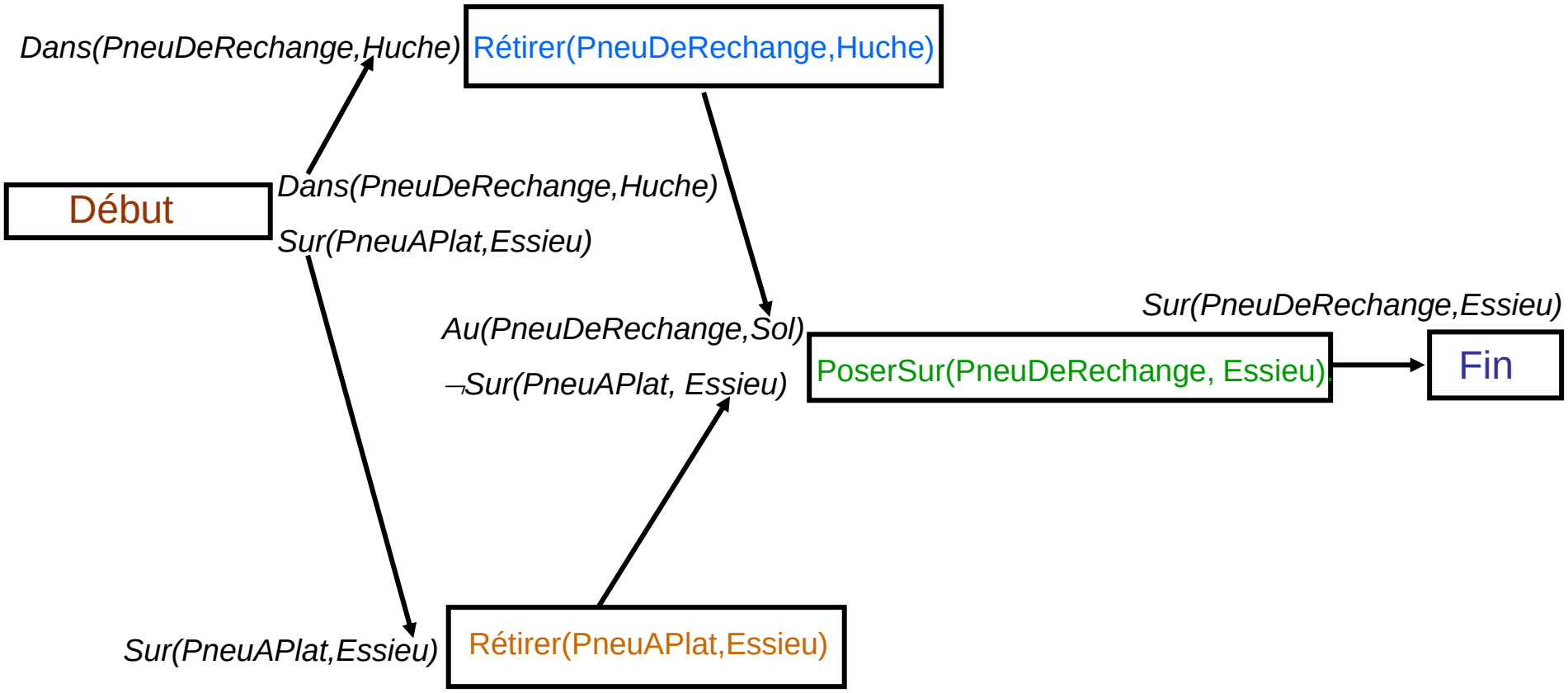
   – Nous ne pouvons pas la placer avant *Début* (puisque rien ne peut être avant *Début*) et nous ne pouvons pas la placer après *Rétirer(PneuDeRechange,Huche)* (puisque il y a déjà une contrainte d'ordre avant *Rétirer(PneuDeRechange,Huche))*
   – Nous sommes forcés de revenir en arrière et retirer l'action *PartirPendantLaNuit* et les deux derniers liens causaux et retourner à l'état précédent. Le planificateur a démontré que l'action *PartirPendantLaNuit* ne peut pas être utilisée comme moyen pour changer un pneu

5. Considérer à nouveau la précondition ¬*Sur(PneuAPlat, Essieu)* de l'action *PoserSur(PneuDeRechange, Essieu)*. Cette fois ci nous choisissons l'action *Rétirer(PneuPlat, Essieu)*

6. Prendre la précondition *Dans(PneuDeRechange,Huche)* de l'action *Rétirer(PneuDeRechange,Huche)* et choisir *Début* pour la satisfaire. Cette fois ci il n'y a pas de conflits

7. Prendre la précondition *Sur(PneuAPlat, Essieu)* de *Rétirer(PneuPlat, Essieu)* et choisir *Début* pour la satisfaire

Solution Finale



Remarques

- Les liens causaux conduisent à un élagage d'une partie de l'espace de recherche, qui à cause de conflits non résolvables, ne contient pas de solutions

- La solution présentée précédemment est un plan partiellement ordonné. Dans cet exemple l'avantage paraît petit puisque il existe seulement deux linéarisations

Références

- *Artificial Intelligence: A Modern Approach*, S. Russell, P. Norvig