SYSTEMES ELEMENTAIRES DE GENERATION DE PLANS D'ACTIONS EN ROBOTIQUE

0. Introduction Les problèmes où les buts sont atteints au moyen de séquences d'actions sont des problèmes classiques en I.A. La résolution des problèmes en robotique est un domaine dans lesquels ces types de problèmes peuvent se poser.

1. Position des problèmes en Robotique: Un robot a un repertoire d'actions qu'il peut accomplir dans un monde facile à comprendre. Par exemple un véhicule qui accomplit des taches, déplacer des objets dans un environnement contenant d'autres objets etc….

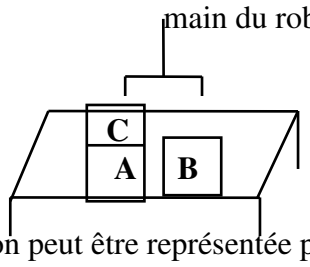
Exemple: Dans le monde des cubes, nous imaginons plusieurs cubes se trouvant sur une table ou les uns sur les autres et un robot ayant une main mobile capable de ramasser et de déplacer des cubes.

La programmation d'un robot nécessite l'intégration de nombreuses fonctions :
- la perception du monde qui l'entoure (le monde où il évolue, les états),
- les actions qu'il peut accomplir (les règles),
- la formulation de plans d'actions et le contrôle de l'exécution de ces plans.

Problème: Synthétiser une séquence d'actions, qui si elle est exécutée convenablement, le robot atteindra le but fixé à partir d'un état initial. La synthèse d'actions d'un tel problème peut être résolue par un système de production (SP) où la BDG décrit l'état du monde dans lequel évolue le robot et les règles représentent les actions du robot.

1.1 Description d'états du monde et de buts: Les descriptions d'états du monde et de buts des problèmes de robotique peuvent être construites à partir des fbfs:

Exemple :

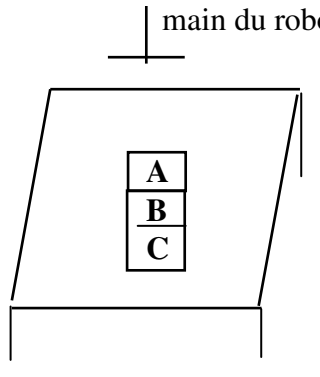


Cette situation peut être représentée par la conjonction suivante:

$découvert(B) \wedge découvert(C) \wedge sur(C,A) \wedge surtable(A) \wedge surtable(B) \wedge mainvide$

Toute conjonction finie de formules décrit une situation du monde différente, où chaque membre peut être considéré comme une interprétation satisfaisant les formules.

Les buts peuvent être exprimés aussi comme des formules en logique des prédicats.



but par la conjonction: $sur(B,C) \wedge sur(A,B)$

1.2 Les Actions du robot: Les actions d'un robot transforment un état en un autre. On peut les modéliser par des regles-av qui transforment un état en un autre. Une technique simple mais extrêmement efficace pour représenter les actions des robots a été utilisée par un système de résolution de problème en robotique appelé STRIPS. Lorsqu'une regle-av est appliquée, on change l'état en ajoutant une structure supplémentaire et en supprimant des expressions qui pourraient ne plus être vraies.

Exemple: Si le robot ramasse B, alors *surtable(B)* et *mainvide* ne sont plus vérifiés, on les supprime de la description d'état et on ajoute *tenu(B)* dans la nouvelle description.

Les regles-av de format STRIPS spécifient les expressions à supprimer au moyen d'une liste. Ces règles sont des quadruplés de la forme:

(<Nom_de_l'action (Paramètres)>, < Preconditions> , <Ajouts> , <Retraits>)
où : **<Nom_de_l'action>:** est le nom de la règle.
<Paramètres>: Sont les arguments de la règle. Les variables sont existentielles.

<Pré condition>: Analogue à l'antécédent d'une implication. C'est une expression qui doit être une Conséquence Logique des faits de la description d'état pour que la règle-av puisse être appliquée via une substitution σ.

<Retraits>: C'est une liste de littéraux appelée liste des retraits. Lorsqu'une regle-av est appliquée à une description d'états, la substitution est appliquée aux littéraux dans la liste des retraits et les instances closes ainsi obtenues sont supprimées de l'ancienne description d'état avant d'en construire une nouvelle.

<Ajouts>: Elle consiste en une conjonction de littéraux et est semblable à une consequence d'une regle-av. Lorsqu'une regle-av est appliquée à une description d'état, la substitution est appliquée à la liste des ajouts et l'instance résultante est ajoutée à l'ancienne description.

Rmq Toutes les variables des Retraits et Ajouts doivent apparaitre dans la précondition.
Exemple: *Ramasser(x)*
Pré condition: surtable(x) ∧ mainvide ∧ découvert(x)
Retraits: surtbale(x), mainvide, découvert(x)
Ajouts: tenu(x)

Avec l'état initial « *découvert(B)∧découvert(C)∧sur(C,A)∧surtable(A)∧surtable(B)∧ mainvide* » et en appliquant cette règle via la substitution x/B, l'état devient *découvert(C)∧ sur(C,A) ∧ surtbale(A) ∧ tenu(B)*

2. Comme Un Système de Production en Chainage-Av:

On peut utiliser comme système de résolution d'un problème de robotique un système de production SP. Un SP simple est un système utilisant comme BDG la description d'état et les règles modélisant les actions. On sélectionne des regles-av applicables jusqu'à ce qu'on produise une description d'états qui s'unifie avec l'expression de but.

Considérons les règles de formats STRIPS correspondantes aux actions du robot de notre exemple des cubes.

1) ramasser(x): Pré condition & Retrait: surtable(x), découvert(x), mainvide
Ajout: tenu(x)
2) poser(x): Pré condition & Retrait: tenu(x)
Ajout: surtable(x), découvert(x), mainvide
3) empiler(x, y): Pré condition & Retrait: tenu(x), découvert(y)
Ajout: mainvide, sur(x, y), decouvert(x)
4) désempiler(x, y): Pré condition & Retrait: mainvide, sur(x, y), decouvert(x)
Ajout: tenu(x), découvert(y).

Supposons que le but à atteindre soit le but précédent (voir plus haut). En appliquant ces règles suivant une certaine stratégie, on obtient un espace de recherche. On cherche un chemin qui nous fait passer de l'état initial vers l'état final. La sequence d'actions est la suivante, appelée plan d'actions:

désempiler(C,A), poser(C), ramasser(B), empiler(B,C), ramasser(A), empiler(A,B).

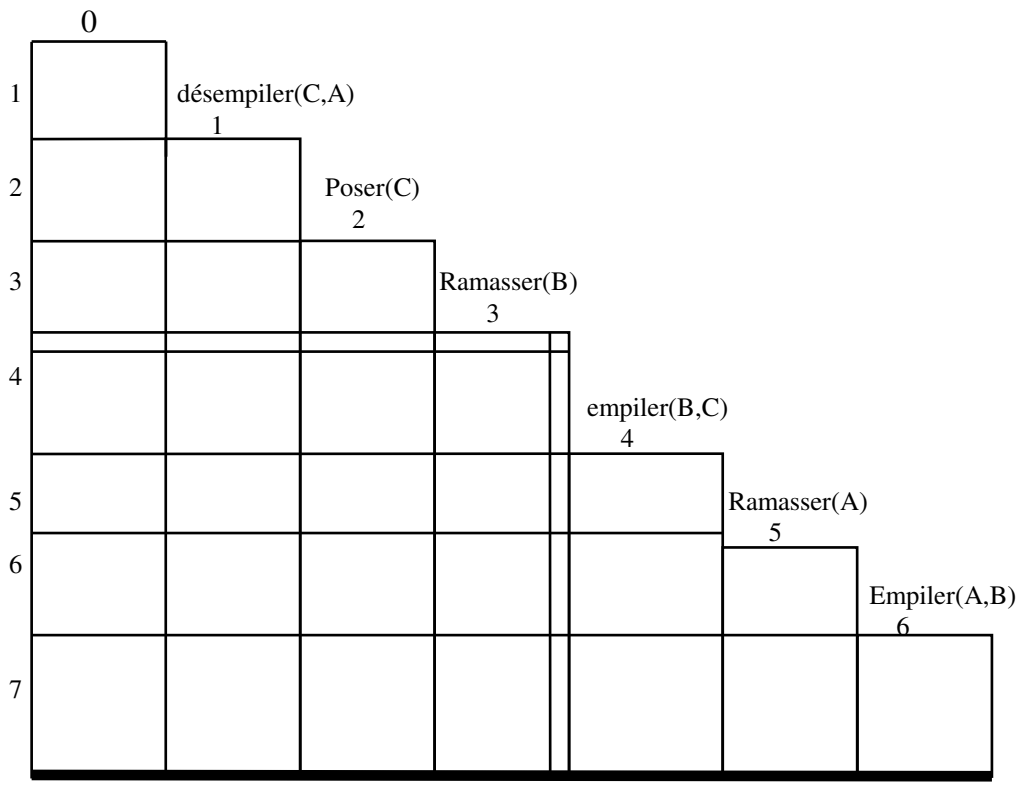
Exercice Donner l'espace de recherche

3. Table Triangulaire (Représentation des plans)

Il est utile d'avoir des informations supplémentaires incluses dans la specification d'un plan; Il se peut par exemple que nous souhaiterions connaître les relations entre les regles-av, les preconditions qu'elles fournissent pour d'autres règles. Ce type d'informations peut être donné par une table triangulaire dont les entrées correspondent aux preconditions et aux ajouts des règles-av.

Soit N le nombre de règles du plan d'actions (pour notre exemple précédent N=6 règles)
- Les noms des colonnes 1-N représentent les regles-av de la sequence.
- La colonne 0 contient les littéraux de l'état initial
- La dernière ligne (N+1) contient les littéraux du but.
- Les rubriques de la case (i,j) de la table pour i<N+1 et j>0 sont les littéraux ajoutés à la description d'états par la j^{eme} règle qui se retrouvent comme precondition de la i^{eme} règle.
- Les elements dans la case (i,0), pour i<N+1 sont les elements de la description initiale qui se retrouvent comme precondition de la i^{eme} règle.
- Les rubriques de la N+1eme ligne sont les littéraux de la description de l'état initial et ceux ajoutés (appartiennent à la liste d'ajouts) par les regles-av qui sont des composantes du But.
- Les rubriques à gauche de la ieme règle sont ses precondtions.

Ces tables peuvent donc être construites à partir de la description de l'état initial, des regles-av, de la séquence d'action et de la description du but.



Exercice : Compléter cette table triangulaire de la séquence précédente.

Définition : On définit le i^{eme} noyau d'une table comme l'intersection de toutes lignes sous la i^{eme} ligne (i^{eme} comprise) avec toutes les colonnes à gauche de la i^{eme} colonne.

Ces rubriques correspondent aux conditions qui doivent être unifiées avec une description d'états afin que la séquence composée par la ieme regle-av et les règles ultérieures soit applicable et atteigne le but.

Cas particuliers: Le 1er noyau correspond à l'état initial, le (N+1)eme contient le but.

Ces tables sont utilisées en robotique pour contrôler l'exécution des plans d'actions, pour prendre en compte les effets imprévus et indésirables. Supposons que le robot a subi un effet indésirable lors de l'exécution de l'action, on s'écarte alors du but. On peut produire un nouveau plan mais cette solution est coûteuse. On cherche alors un moyen intelligent pour continuer le travail.

Supposons que le système vient d'exécuter les (i-1) actions. Alors pour que la i^{eme} regle-av et les suivantes soient applicables et appropriées, les littéraux du noyau doivent s'unifier avec la description du nouvel état tel qu'il est maintenant. On cherche alors le noyau qui filtre et qui a le plus grand numéro dans la table.

Pour trouver le noyau adéquat qui s'unifie avec une description d'état, nous vérifions chacun des noyaux en commençant par celui qui porte le numéro le plus élevé:
- Si la dernière ligne (but) filtre alors arrêt
- Si le noyau qui filtre est le i^{eme} alors nous savons que la i^{eme} règle est applicable. On execute la règle et les suivantes.

Remarque: S'il n'y a pas de plan qui filtre, il faut générer un nouveau plan.

4. Comme Un Système de Production en Chaînage arrière :

Nous voulons maintenant définir un système qui travaille en chaînage arrière (c-à-d allant du but vers l'état initial). Un tel système commence avec la description de but comme BDG initiale et lui applique des regles-ar pour produire des descriptions de sous-buts. Il s'achève avec succès lorsqu'il produit une description de sous-buts qui s'unifie avec la description initiale. La 1^{ere} étape est de produire des règles-ar qui transforment des buts en sous-buts.

4.1 La notion de régression Pour pouvoir utiliser un tel système en chaînage arrière, on définit ce qu'on appelle la **Régression.** Cette stratégie consiste à utiliser des règles-ar qui reposent sur les règles-av. Une règle-ar qui transforme un but B en un sous-but B' repose d'un point de vue logique sur la règle-av qui lorsqu'elle est appliquée à une description d'état filtrée par B' donne lieu à une description d'état filtrée par B.

On fait remarquer que dans les règles-av définies, l'application de l'une d'elles à n'importe quelle description d'état produit une description d'état qui s'unifie avec les littéraux de la liste d'ajouts. C'est pourquoi si une expression de but contient un littéral L qui s'unifie avec un des littéraux de la liste d'ajouts d'une règle-av, alors nous savons que si nous produisons une description d'état filtrée par des instances adéquates des préconditions de cette règle-av, la règle-av peut être appliquée pour produire une description filtrée par L. Par conséquent, l'expression de sous but produite par une application en chaînage arrière d'une règle-av doit certainement contenir des instances des préconditions de cette règle-av. Cependant si l'expression de but contient d'autres littéraux autre que L alors l'expression de sous-but doit aussi contenir d'autres littéraux qui après l'application de la règle-av, sont transformés en ces littéraux différents de L dans l'expression de but.

Formellement soit un but *(L ∩ B1 ∩.... ∩ Bn).* Nous voulons utiliser une règle-av (en chaînage arrière) pour produire une expression de sous-but. Supposons qu'une règle-av dont la formule de précondition est P ait une liste d'ajouts A contenant un littéral L' qui s'unifie avec L par σ. Les littéraux dans σP forment un sous-ensemble des littéraux du sous-but que nous recherchons. Nous devons inclure les expressions B1', B2',....,Bn' dans le sous-but complet; Ces expressions B1',...,Bn' doivent être telles que l'application de l'instance de la règle-av à n'importe quelle description d'état filtrée par ces expressions produise une description d'état filtrée par B1,...,Bn. **Chaque Bi' est appelée la régression de Bi à travers l'instance de la règle-av.**

Soit *R(Q; σF)* la régression d'un littéral Q à travers une instance close σF d'une règle-av F dont la pré condition est P, la liste des retraits S, et la liste des ajouts A. Alors:

Si ***σQ est un littéral dans σA***
Alors ***R(Q; σF) = Vraie***
Sinon Si σQ est un littéral dans σS
Alors R(Q;σF) = Faux
Sinon R(Q;σF)=Qσ

En résumé une règle-av peut être utilisée comme règle-ar de la manière suivante:

- La condition d'applicabilité de la règle-ar est que l'expression de but contienne un littéral qui s'unifie par σ avec l'un des littéraux de la liste des ajouts de la règle-av .
- L'expression de sous-buts est créée en régressant les autres littéraux (non unifiés) de l'expression de but à travers l'instance par σ de la règle-av et en connectant ceux-ci au moyen d'une conjonction à l'instance de précondition de la règle-av.

Exemple 1 : Supposons que notre expression de but soit *sur(A, B) ∧ sur(B, C).*
Il y'a 2 façons d'utiliser *empiler(x,y)* comme regle-ar sur ce but. Les substitutions correspondantes aux 2 sous-buts sont respectivement {*x/A,y/B}* et *{x/B,y/C}*.
Soit la 1ere , La description de sous but est produite comme suit : Régresser les expressions (non unifiés) *sur(B,C)* à travers *empiler(A,B)* ce qui nous donne *sur(B,C) (voir algorithme précédent).* Ajouter les expressions *Tenu(A), découvert(B)* (qui sont les pré conditions de la règle) pour donner le sous but : *sur(B,C) ∧Tenu(A)∧ découvert(B).*

Exemple 2 : Soit une expression de but contenant *découvert(A).* 3 règles ont ce littéral dans leur liste d'ajouts. Soit *désempiler(x,y*) par {y/A}, l'expression de sous but créée est {*mainvide∧ découvert(x)∧sur(x,A)* où x est une variable existentielle qui signifie un cube vide sur A.

Exemple 3 : Supposons que nous voulions appliquer la règle *désempiler* à une expression de but *découvert(A) ∧ Mainvide*. La substitution est {*y/A}*. La régression de *Mainvid*e à travers *désempiler(x,A)* est *Faux (voir alg précédent).* Nous voyons donc que l'application de cette règle a créée un sous-but impossible.

4.2 Exemple de solution :

Soit le but *sur(A,B) ∧ sur(B,C).* Dans cet exemple l'espace de sous but engendré par l'application de toutes les regles-ar est plus grand que l'espace produit en utilisant les regles-av. Cependant une grande partie de description de sous buts est impossible (contiennent *Faux* ou sont impossibles).

Exercice : Donner l'espace de recherche

Elagage des nœuds : Il ne faut pas développer les nœuds dans lesquels on a une situation impossible comme par exemple : *Tenu(B) ∧ sur(A,B), mainvide ∧Tenu(A), Tenu(B) ∧ sur(B,C), toute description contenant Faux,* etc..