Planifier et agir dans le monde réel

1

Plan

- Temps, ordonnancement et ressources
- Décomposition hiérarchique
- Planification dans des environnements non déterministes
  - Planification conditionnelle
  - Replanification
  - Planification continue

2

Complexité du monde réel

- Hypothèses des chapitres précédents:
  - Le monde est accessible, statique et déterministe.
  - La description des actions est complète et correcte et les conséquences sont exactes.
- La réalité est différente pour la plupart des applications du monde réel:
  - Les informations peuvent être incorrectes et incomplètes.
  - Impossibilité d'énumérer toutes les possibilités qui peuvent amener à un échec. Il est impossible d'énumérer toutes les préconditions requises et tous les effets possibles.

3

Temps, ordonnancement et ressources

- La représentation STRIPS spécifie qu'est-ce que l'action fait, mais elle ne spécifie pas la durée de l'action ou quand elle survient.
  - Dans certains domaines, il est important de savoir quand une action commence et quand elle finit.
- Le temps est très important pour une classe d'applications appelée: Ordonnancement d'ateliers (job shop scheduling).
  - Comment accomplir les tâches le plus rapidement possible en respectant les contraintes sur les ressources.

4

Exemple: assemblage de deux autos

Il y a 2 tâches : assembler la voiture C_1 et la voiture C_2. Chacune de ces 2 tâches se compose de 3 actions: ajouter moteur, ajouter roues et contrôler les résultats. Contraintes : Placer le moteur avant roues avant contrôle.

$$Init(Chassis(C_1) \wedge Chassis(C_2) \\ \wedge Engine(E_1, C_1, 30) \wedge Engine(E_2, C_2, 60) \\ \wedge Wheels(W_1, C_1, 30) \wedge Wheels(W_2, C_2, 15))$$

$$Goal(Done(C_1) \wedge Done(C_2))$$

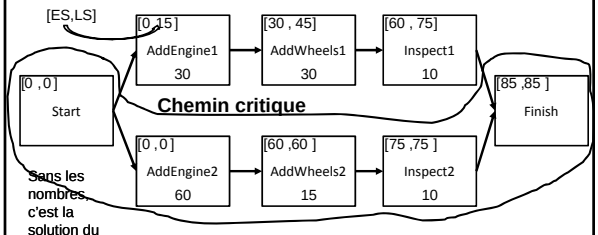
$$Action(AddEngine(e, c, m), \\ \text{PRECOND:} Engine(e, c, d) \wedge Chassis(c) \wedge \neg EngineIn(c), \\ \text{EFFECT:} EngineIn(c) \wedge Duration(d))$$

$$Action(AddWheels(w, c), \text{PRECOND:} Wheels(w, c, d) \wedge Chassis(c), \\ \text{EFFECT:} WheelsOn(c) \wedge Duration(d))$$

$$Action(Inspect(c), \text{PRECOND:} EngineIn(c) \wedge WheelsOn(c) \wedge Chassis(c), \\ \text{EFFECT:} Done(c) \wedge Duration(10))$$

5

Exemple: assemblage de deux autos



Sans les nombres, c'est la solution du POP.

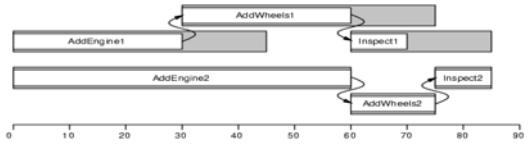
ES = temps de début au plus tôt; LS = temps de début au plus tard

$$ES(Start) = 0$$
$$ES(B) = max_{A \prec B}\ ES(A) + Duration(A)$$
$$LS(Finish) = ES(Finish)$$
$$LS(A) = min_{A \prec B}\ LS(B) - Duration(A)$$

6

Exemple: assemblage de deux autos

- La ligne temporelle: les rectangles gris représentent les intervalles durant lesquelles une action peut être exécutée, en respectant les contraintes d'ordre. La partie inoccupée d'un rectangle gris correspond à la marge



7

Ordonnancement avec des ressources

- Les contraintes sur les ressources complexifient les problèmes d'ordonnancement.
- Par exemple, si on a un seul appareil de levage de moteur (Palan), on ne peut pas mettre les deux moteurs en même temps.
- Les ressources sont représentées en ajoutant un autre champ *Ressource: R(k)* où *k* est le nombre d'unités de la ressource nécessaires pour exécuter l'action.

8

Exemple

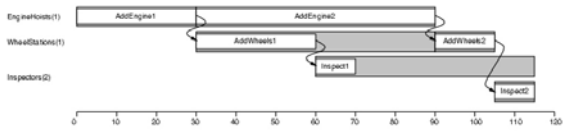
$Init(Chassis(C_1) \land Chassis(C_2)$
$\quad \land Engine(E_1, C_1, 30) \land Engine(E_2, C_2, 60)$
$\quad \land Wheels(W_1, C_1, 30) \land Wheels(W_2, C_2, 15)$
$\quad \land EngineHoists(1) \land WheelStations(1) \land Inspectors(2))$
$Goal(Done(C_1) \land Done(C_2))$

$Action(AddEngine(e, c, m),$
$\quad \text{PRECOND:} Engine(e, c, d) \land Chassis(c) \land \neg EngineIn(c),$
$\quad \text{EFFECT:} EngineIn(c) \land Duration(d),$
$\quad \text{RESOURCE:} EngineHoists(1))$
$Action(AddWheels(w, c),$
$\quad \text{PRECOND:} Wheels(w, c, d) \land Chassis(c),$
$\quad \text{EFFECT:} WheelsOn(c) \land Duration(d),$
$\quad \text{RESOURCE:} WheelStations(1))$
$Action(Inspect(c),$
$\quad \text{PRECOND:} EngineIn(c) \land WheelsOn(c),$
$\quad \text{EFFECT:} Done(c) \land Duration(10),$
$\quad \text{RESOURCE:} Inspectors(1))$

9

Exemple

- En tenant compte des contraintes sur les ressources, le plan est plus long (115 min. au lieu de 85 min.).



10

Algorithmes d'ordonnancement

- L'ordonnancement de tâches en tenant compte des contraintes sur les ressources est un problème très complexe.
  - C'est un domaine de recherche très actif.
- Exemples d'algorithmes
  - « Minimum Slack » (marge minimale): Approche gloutonne qui choisit parmi les actions non ordonnancées, l'action qui peut être exécutée le plus tôt.
  - « Plan first, schedule later »: Approche qui consiste à crée un plan partiellement ordonnée et de l'ordonnancer après.

11

Exercice

Initial(m,n,o)
But(x,y,z)
Action(A, Précondition: m, Effet: p ∧ durée(10))
Action(B, Précondition: p, Effet: t ∧ x ∧ durée(15))
Action(C, Précondition: n, Effet: s ∧ durée(20))
Action(D, Précondition: s ∧ t ∧ v, Effet: y ∧ durée(50))
Action(E, Précondition: o, Effet: w ∧ v ∧ durée(30))
Action(F, Précondition: w, Effet: z ∧ durée(70))

- Construire le plan
- Calculer les intervalles de temps [ES,LS].
- Trouver le chemin critique
- Construire la ligne de temps
- Construire la ligne de temps si A, C et E utilisent la même ressource

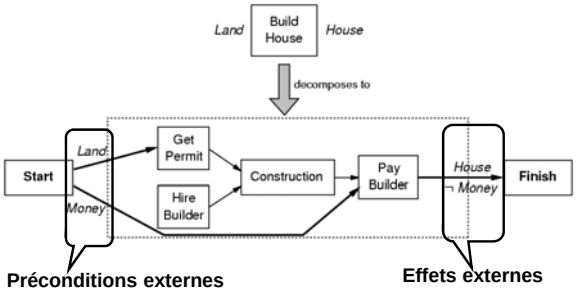
12

Décomposition hiérarchique

- On utilise une décomposition hiérarchique des actions pour simplifier la planification.
  - On commence par planifier des actions de haut niveau.
  - Par la suite, on décompose graduellement les actions en sous plans.
  - On arrête la décomposition lorsque l'on a que des actions primitives.

13

Exemple de décomposition



14

Représentation des décompositions

- Les différentes décompositions possibles d'une action sont représentées par des sous plans.
- Chaque décomposition est de la forme: *Decompose*(*a*,*d*), où *a* est une action qui peut être décomposée par le plan partiellement ordonné *d*.

15

Exemple: représentation d'une décomposition

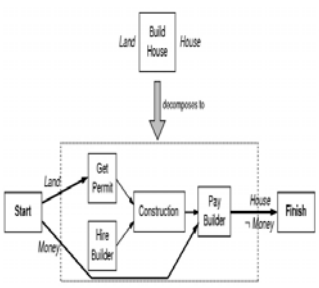
$Action(BuyLand,$ PRECOND:$Money,$ EFFECT:$Land \wedge \neg Money)$
$Action(GetLoan,$ PRECOND:$GoodCredit,$ EFFECT:$Money \wedge Mortgage)$
$Action(BuildHouse,$ PRECOND:$Land,$ EFFECT:$House)$

$Action(GetPermit,$ PRECOND:$Land,$ EFFECT:$Permit)$
$Action(HireBuilder,$ EFFECT:$Contract)$
$Action(Construction,$ PRECOND:$Permit \wedge Contract,$
EFFECT:$HouseBuilt \wedge \neg Permit)$
$Action(PayBuilder,$ PRECOND:$Money \wedge HouseBuilt,$
EFFECT: $\neg Money \wedge House \wedge \neg Contract)$

$Decompose(BuildHouse,$
$Plan($STEPS: $\{S_1 : GetPermit, S_2 : HireBuilder,$
$S_3 : Construction, S_4 : PayBuilder\}$
ORDERINGS: $\{Start \prec S_1 \prec S_3 \prec S_4 \prec Finish, \quad Start \prec S_2 \prec S_3\},$
LINKS: $\{Start \xrightarrow{Land} S_1, Start \xrightarrow{Money} S_4,$
$S_1 \xrightarrow{Permit} S_3, S_2 \xrightarrow{Contract} S_3, S_3 \xrightarrow{HouseBuilt} S_4,$
$S_4 \xrightarrow{House} Finish, S_4 \xrightarrow{\neg Money} Finish\}))$

16

Exemple: représentation d'une décomposition (2)



$Action(BuyLand,$ PRECOND:$Money,$ EFFECT:$Land \wedge \neg Money)$
$Action(GetLoan,$ PRECOND:$GoodCredit,$ EFFECT:$Money \wedge Mortgage)$
$Action(BuildHouse,$ PRECOND:$Land,$ EFFECT:$House)$

$Action(GetPermit,$ PRECOND:$Land,$ EFFECT:$Permit)$
$Action(HireBuilder,$ EFFECT:$Contract)$
$Action(Construction,$ PRECOND:$Permit \wedge Contract,$
EFFECT:$HouseBuilt \wedge \neg Permit)$
$Action(PayBuilder,$ PRECOND:$Money \wedge HouseBuilt,$
EFFECT: $\neg Money \wedge House \wedge \neg Contract)$

$Decompose(BuildHouse,$
$Plan($STEPS: $\{S_1 : GetPermit, S_2 : HireBuilder,$
$S_3 : Construction, S_4 : PayBuilder\}$
ORDERINGS: $\{Start \prec S_1 \prec S_3 \prec S_4 \prec Finish, \quad Start \prec S_2 \prec S_3\},$
LINKS: $\{Start \xrightarrow{Land} S_1, Start \xrightarrow{Money} S_4,$
$S_1 \xrightarrow{Permit} S_3, S_2 \xrightarrow{Contract} S_3, S_3 \xrightarrow{HouseBuilt} S_4,$
$S_4 \xrightarrow{House} Finish, S_4 \xrightarrow{\neg Money} Finish\}))$

17

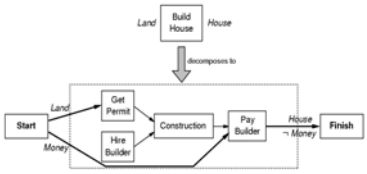
Préconditions et effets

- Il peut y avoir plus qu'une décomposition pour une action.
  - Exemple, construire une maison de ses propres mains à partir de pierres.
- Les préconditions de l'action de haut niveau sont l'intersection des préconditions externes de ses décompositions.
- Les effets de l'action de haut niveau sont l'intersection des effets externes de ses décompositions.

18

Préconditions et effets

- Construire de ses mains: Préconditions: *Rocks* et *Land*, Effets: *House* et *BadBack*.
- L'intersection des préconditions et des effets donne *Land* comme précondition à *BuildHouse* et *House* comme effet.



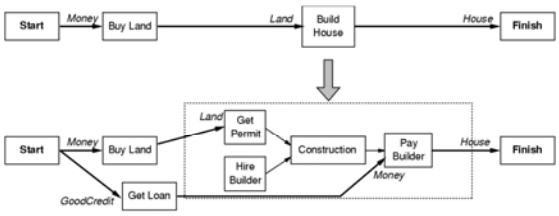
19

Modification du planificateur pour les décompositions

- Pour ajouter une décomposition:
  - Enlever l'action *a'* du plan *P*. Ensuite, pour chaque action *s* de la décomposition *d'*, il faut choisir une action *s'* et l'ajouter au plan. *s'* peut être une nouvelle action ou une action déjà présente dans le plan.
  - Toutes les contraintes internes de *d'* sont copiées dans le plan.
  - Par la suite, il faut ajuster les contraintes d'ordonnancement.
  - Les liens causaux sont ajoutés.

20

Exemple de décomposition



- *Money* devient une précondition non réalisée, donc on doit ajouter l'action *GetLoan* par la suite.

21

Avantages des planificateurs hiérarchiques

- Les planificateurs hiérarchiques permettent à des experts humains de diriger la planification en disant au planificateur comment réaliser des tâches complexes.
- Ce type de planificateur est très utile en pratique.
  - Par exemple, O-PLAN a été utilisé pour produire des plans de production pour Hitachi.
  - Le problème consiste en une ligne de production de 350 produits différents, 35 machines d'assemblages et plus de 2000 opérations différentes.
  - Le planificateur permet de générer un horaire de 30 jours, avec trois quarts de travail de huit heures par jour, contenant des millions d'étapes.

22

Planification dans des environnements non déterministes

- Jusqu'à maintenant, on a considéré des domaines de planification classiques, c'est-à-dire, complètement observables, statiques et déterministes.
  - L'agent peut exécuter le plan les yeux fermés
- Dans un environnement incertain, l'agent doit pouvoir percevoir ce qui se passe pendant l'exécution du plan et le modifier si nécessaire.

23

Information incomplète et incorrecte

- Les agents doivent gérer les informations incomplètes et incorrectes:
  - Incomplète: parce que le monde est partiellement observable, non déterministe ou les deux.
    - Ex: la porte du bureau peut être ou ne pas être barrée.
    - Une de mes clés peut ou ne peut pas ouvrir la porte.
  - Incorrecte: parce que le monde ne correspond pas nécessairement à mon modèle du monde.
    - Ex: Je crois que ma clef ouvre la porte du bureau, mais ça peut être faux si quelqu'un a changé la serrure.

24

Degré d'incertitude

- Incertitude bornée: les actions ont des effets imprédictibles, mais les effets possibles peuvent être listés.
  - Ex: Pile ou Face
- Incertitude non bornée: L'ensemble des préconditions ou des effets est soit inconnu ou trop grand pour être énuméré.
  - Ex: Conduire une auto.

25

Méthodes pour gérer l'incertitude

- «Planification sans capteurs»: Construit des plans pour être exécutés sans perception. L'algorithme doit assurer que le plan atteint le but dans toutes les circonstances possibles. Souvent pas applicable.
- «Planification conditionnelle»: Construit des plans conditionnels contenant des branches pour les différentes éventualités. À chaque jonction, l'agent effectue une action de perception pour voir qu'elle branche il doit prendre.

26

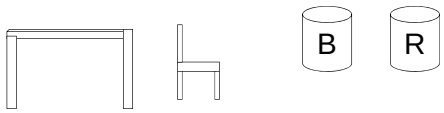
Méthodes pour gérer l'incertitude

- « Monitoring de l'exécution et replanification»: L'agent construit un plan, mais suit le développement de celui-ci pendant l'exécution. Si quelque chose n'est pas correcte, il reconstruit un nouveau plan.
- «Planification continue»: Un planificateur qui persiste dans le temps. Il peut gérer des circonstances inattendues, même s'il est en train de construire un plan. Il peut gérer l'abandon ou l'ajout d'un but.

27

Exemple

- Une chaise, une table et 2 pots de peinture. Les couleurs de chaise et table n'étant pas connues, il faudra atteindre le but : **chaise et table même couleur**



28

Exemple (2)

- **Un agent planificateur classique** ne peut résoudre ce problème car l'état initial n'est pas totalement spécifié
- **Un agent de planification sans capteurs:** Il doit trouver un plan qui fonctionne sans avoir besoin de capteurs pendant l'exécution du plan. La solution consiste à ouvrir n'importe quel pot et à peinturer la table et la chaise. Ce type de forçage est particulièrement adapté quand sont coûteuses et impossibles à prévoir.
  - Ex: les médecins prescrivent des médicaments à large spectre au lieu de passer par des tests coûteux.

29

Exemple (3)

- **Un agent planificateur conditionnelle** peut générer un meilleur plan,
  - en commençant par détecter la couleur de la table et de la chaise.
    - Si les couleurs sont identiques, le plan est accompli.
    - Si ce n'est pas le cas, il lit les étiquettes sur les pots. Si une boite est de la même couleur que l'un des deux meubles, il l'applique au 2ème meuble.
  - Dans les autres cas, il peint les 2 meubles en utilisant une couleur quelconque.

30

Exemple (4)

- **Un agent de replanification** peut générer un même plan que précédemment, il pourrait en plus vérifier les effets des actions.
  - Si par exemple, il se rend compte qu'un des pieds n'a pas été peint, il pourrait alors établir un nouveau plan pour le pied.

31

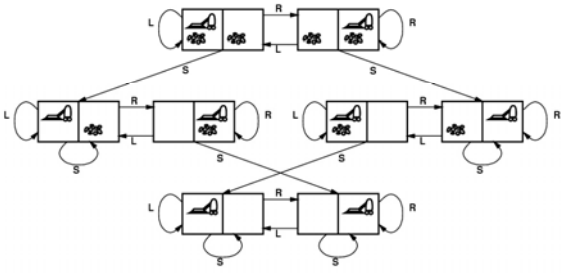
Planification conditionnelle

- Dans un environnement non déterministe, les agents ne peuvent pas prédire les résultats de leurs actions.
- La planification conditionnelle permet de gérer le non déterminisme en ajoutant dans le plan des actions conditionnelles permettant de choisir le bon sous-plan selon la situation.
- L'agent a donc un plan pour toutes les éventualités possibles.

32

Exemple: « Vacuum world »

- Espace d'états:



33

Exemple: « Vacuum world »

- Dans le « vacuum world », il y a trois actions possibles: *Right*, *Left*, *Suck*.
- Pour définir les états, on va utiliser les propositions:
  - *AtL (AtR):* l'agent est à gauche ou à droite.
  - *CleanL (CleanR):* la case de gauche ou de droite est propre.

34

Augmenter le langage STRIPS

- Pour tenir compte du non déterminisme, on va devoir ajouter des effets disjonctifs aux actions.
- Par exemple, si l'action de bouger à gauche peut ne pas fonctionner,
  - la définition de l'action va passer de:

$$Action(Left, \text{PRECOND}: AtR, \text{EFFECT}: AtL)$$

  - à:

$$Action(Left, \text{PRECOND}: AtR, \text{EFFECT}: AtL \vee AtR)$$

35

Augmenter le langage STRIPS

- On peut aussi ajouter des effets conditionnels lorsque l'effet dépend de l'état dans lequel il est exécuté.
  - La syntaxe est: « **when** <condition>: <effet> »
  - Exemple:

$$Action(Suck, \text{PRECOND}:, \text{EFFECT}: (\textbf{when}\ AtL: CleanL) \wedge (\textbf{when}\ AtR: CleanR))$$

36

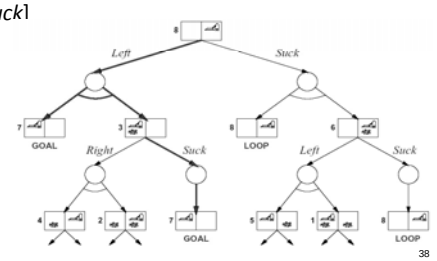
Étapes conditionnelles

- Une étape conditionnelle est définie de la manière suivante:
  - « **if** <test> **then** planA **else** planB »
  - Exemple: **if** *AtL* ∧ *CleanL* **then** *Right* **else** *Suck*
- En ajoutant les étapes conditionnelles, les plans deviennent des graphes ET-OU.

37

Exemple d'un graphe ET-OU

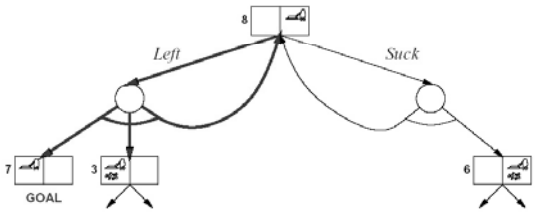
- Bouger vers une case propre et aspirer une case propre peut la salir.
- Plan solution de ce graphe ET-OU = [*Left,* **if** *CleanL* **then** [] **else** *Suck*]



38

Exemple graphe avec cycles

- Bouger peut ne rien faire.



39

Exercice

- On suppose qu'une personne est malade et qu'il existe deux médicaments (*A* et *B*) pour la guérir.
  - Les états possibles sont *Malade* et *EnSanté*.
  - Les actions des deux médicaments sont identiques:
    - La précondition est que le patient est malade.
    - L'effet est soit que le patient est encore malade ou qu'il est en santé
- Décrire les actions avec le langage STRIPS augmenté.
- Construire le graphe ET-OU.

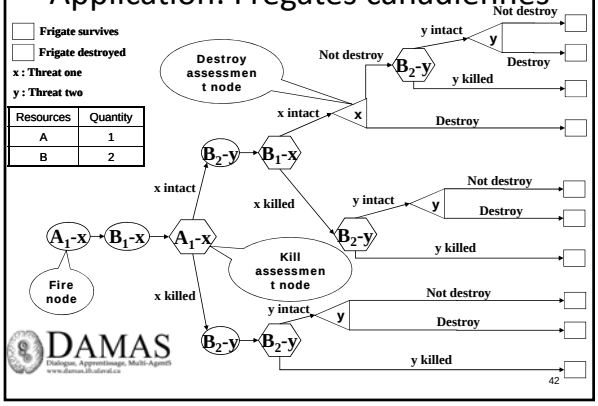
40

Remarques

- Pour que la planification conditionnelle soit applicable, il faut que l'agent connaisse toutes les éventualités possibles.
- C'est une technique coûteuse parce qu'elle planifie pour beaucoup de cas improbables.
- Par contre, si c'est possible de connaître toutes les éventualités, l'exécution du plan est très rapide, car on n'a pas besoin de replanifier.

41

Application: Frégates canadiennes



42

Replanification

- L'agent surveille l'exécution du plan pour voir si tout se passe comme prévu, sinon il replanifie.
- La surveillance de l'exécution d'un plan est essentielle dans des environnements réels, parce que l'agent ne peut pas tout prévoir.
- On va considérer deux types de surveillance:
  - Surveillance des actions
  - Surveillance du plan

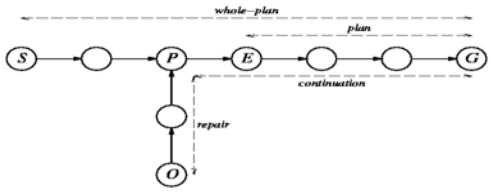
43

Surveillance des actions

- L'agent crée un plan initial.
- Il exécute les actions une par une.
- Avant d'exécuter une action, il vérifie si ses préconditions sont encore satisfaites.
- Si elles ne le sont pas, l'agent va replanifier une séquence d'actions pour revenir à une position valide dans le plan.

44

Exemple de replanification

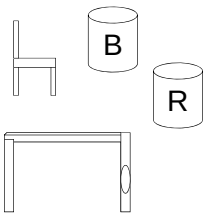


Avant l'exécution, l'agent planificateur propose un plan : *whole-plan* pour aller de *S* à *G*. L'agent exécute jusqu'au point *E*. Avant d'exécuter le restant du plan, il vérifie et il se rend compte qu'il se trouve en *O* au lieu de *E*. Il appel son algo de planification pour proposer une réparation allant de *O* à un point quelconque *P* de *whole-plan* (*P* doit être avant *E*). Le nouveau plan est alors Repair + Continuation.

45

Exemple de la peinture

- Le but de l'agent est que la table et la chaise soient de la même couleur. Au départ, la chaise est bleue, la table est verte et l'agent a deux pots de peinture, un bleu et un rouge.



- Le plan initial trouvé par l'agent serait: [Départ; Ouvrir(B); Peinturer(Table, Bleu); Fin]
- L'agent fait les actions et se rend compte qu'il a oublié une partie. Il peut réparer son plan et essayer de nouveau l'action de peinturer.

46

Surveillance du plan

- Supposons que l'agent veut peinturer la chaise et la table en rouge et qu'il n'a pas suffisamment de peinture.
  - L'agent va peinturer la chaise et seulement après, il va se rendre compte qu'il ne peut pas exécuter l'action *Peindre*(*Table*, *Rouge*) parce que la précondition *Avoir*(*Peinture Rouge*) n'est plus valide.
- Pour détecter l'erreur plus rapidement, on va vérifier si tout le plan restant est valide, et non seulement la prochaine action.

47

Surveillance du plan

- Avec cette technique, on élimine les plans le plus tôt possible.
- Peut empêcher l'agent de faire des actions irréparables.
- Peut tirer avantage de la fortuité (succès accidentels).

48

Désavantages de la replanification

- Ne fonctionne pas dans des environnements temps-réel.
  - Il n'y a pas de limite dans le temps de replanification.
- Par ailleurs, il ne peut pas formuler ou recevoir de nouveaux buts.

49

Planification continue

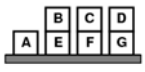
- Un agent de planification continue est un agent qui persiste indéfiniment dans l'environnement.
  - Il passe continuellement par des phases de changement de but, de planification et d'exécution.
  - L'agent est vu comme étant toujours en train d'exécuter un plan, « le grand plan de sa vie ».
  - L'agent surveille le monde continuellement pour mettre à jour son modèle du monde.

50

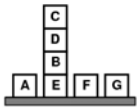
Exemple planification continue

- Monde des blocs.
- Il y a une action possible: *Move*(*x*,*y*)

$$Action(Move(x,y),$$
$$\text{PRECOND:}Clear(x) \wedge Clear(y) \wedge On(x,z),$$
$$\text{EFFECT:}On(x,y) \wedge Clear(z) \wedge \neg On(x,z) \wedge \neg Clear(y))$$



État de départ

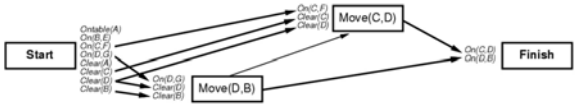


But

51

Exemple planification continue

- L'agent construit son plan de manière incrémentale en intercalant la planification et la perception. Il obtient le plan suivant:

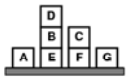


- En planification continue, l'état *Start* représente toujours l'état courant.

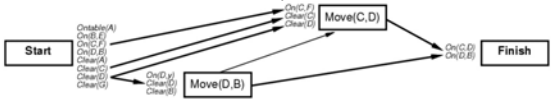
52

Exemple planification continue

- Le plan est prêt à être exécuté, mais avant que l'agent ait le temps de faire une action, un agent externe déplace *D* sur *B*.

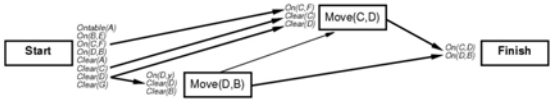


- L'agent perçoit ce changement, met à jour son modèle du monde et son plan.



53

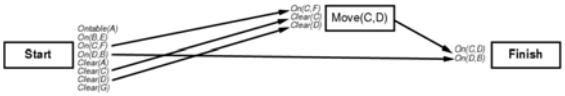
Exemple planification continue



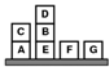
- Le plan est incomplet, il y a des préconditions non résolues.
- On voit aussi que le lien causal entre *Move*(*D*,*B*) et Finish peut être remplacé par un lien causal partant de Start.
- En enlevant ce lien causal, l'action *Move*(*D*,*B*) ne sert plus à rien, donc on peut l'enlever.

54

Exemple planification continue



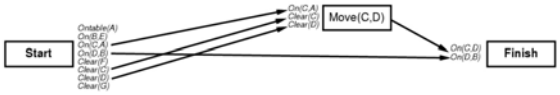
- L'agent peut exécuter l'action *Move*(*C*,*D*) parce que
  - Toutes ses préconditions sont satisfaites par Start
  - Il n'y a pas d'étape qui doivent être faite avant.
  - Elle n'est pas en conflit avec les autres liens dans le plan.
- L'agent est maladroit et met le bloc *C* sur *A* au lieu de sur *D*.



55

Exemple planification continue

- Il n'y a plus d'action dans le plan, mais il y a une précondition non satisfaite pour l'étape *Finish*.
- L'agent planifie pour la précondition non satisfaite.



56

Exemple planification continue

- Finalement, l'action *Move*(*C*,*D*) est exécutée.
- Comme les buts sont satisfaits par l'étape Start et qu'il n'y a plus d'action, l'agent a accomplit le but et il peut donc l'enlever. Il peut alors formuler un autre burt.

57

Enlever les défauts du plan

- Planification continue est similaire à POP, à chaque itération, l'agent enlève un défaut au plan. Les défauts peuvent être:
  - Buts manquants
    - L'agent peut ajouter un nouveau but à l'état *Finish*.
  - Préconditions non satisfaites
    - Ajouter un lien causal vers la précondition, en ajoutant une nouvelle action ou en utilisant une action existante.
  - Conflit avec les liens causaux
    - S'il y a un lien causal entre *A* et *B* en conflit avec un effet d'une action *C*, alors ajouter une contrainte d'ordonnancement pour que *C* soit avant *A* ou après *B*.

58

Enlever les défauts du plan (suite)

- Lien non supporté
  - S'il y a un lien causal à partir de *Start* qui n'est plus vrai, enlever le lien.
- Action redondante
  - Si une action ne fournit plus de lien causal, l'enlever du plan.
- Action pas exécutée
  - Si une action A a toutes ses préconditions satisfaites par Start, qu'il n'y a pas d'étape qui doivent être faite avant et qu'elle n'est pas en conflit avec des liens causaux, alors enlever A et ses liens causaux du plan et retourner A comme action à exécuter.
- Ancien but non nécessaire
  - S'il n'y a plus de préconditions non satisfaites et qu'il n'y a plus d'action dans le plan, le but a été atteint, enlever le but.

59

Algorithme

- L'agent effectue un cycle « perçoit, enlève défaut, agit ».
- L'action retournée par *REMOVE-FLAW*(*plan*) est souvent l'action *NoOp* pour dire que l'agent ne fait rien pour ce tour.

function CONTINUOUS-POP-AGENT(*percept*) **returns** an *action*
static: *plan*, a plan, initially with just *Start*, *Finish*

action ← *NoOp* (the default)
EFFECTS[*Start*] = UPDATE(EFFECTS[*Start*], *percept*)
REMOVE-FLAW(*plan*) *// possibly updating action*
return *action*

60

Planification multiagent

- 3 types d'environnements multiagents:
  - Compétitif
  - Coopératif
  - Coexistence

61

Agents coopératifs

- But commun
- Plan conjoint
  - Définit les actions à effectuer par chaque agent pour atteindre le but.
- Les agents doivent s'entendre sur le plan conjoint à exécuter.
- Ils doivent donc se coordonner.
- À chaque instant, les agents exécutent une action conjointe.

62

Coordination

- Par convention ou loi sociale.
  - Les agents s'entendent sur des conventions à respecter avant de s'engager dans une activité conjointe.
  - Les conventions guident les agents dans leurs choix des plans conjoints.
- Par communication
  - Les agents s'informent des actions qu'ils vont exécuter.
- Les agents doivent déterminer une intention conjointe.

63

Agents en compétition

- Chaque agent doit modéliser les autres agents pour estimer ce qu'ils vont faire et agir de manière à maximiser leur propre utilité.
- Exemple: Othello
- Ils peuvent aussi être en compétition pour une ressource comme une imprimante.
  - Chaque agent devra évaluer l'impact des actions des autres agents de manière à accomplir sa tâche à l'aide de la ressource le plus efficacement possible.

64