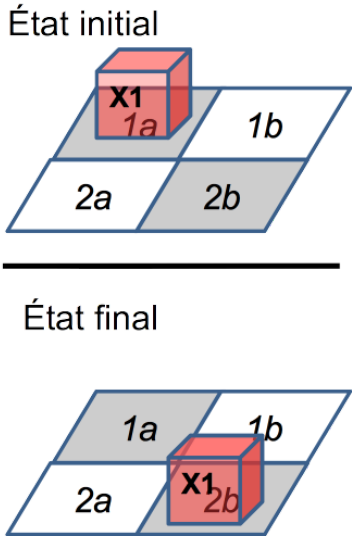
Protocole d'expérimentation pour les concepts de la planification

Personnes présentes

- K: expérimentateur
- S: sujet de l'expérimentation



Niveau 1 – Tâche 1

Nom d'objet	Type	Propriété (état initial)	Propriété généralisé
X1	cube	est_rouge (X1)	est_rouge (cube)
		est_sur(X1,1a)	est_sur(cube,position)
1a	position	non_vide(1a)	non_vide(position)
1b	position	est_vide(1b)	est_vide(position)
2a	position	est_vide(2a)	est_vide(position)
2b	position	est_vide(2b)	est_vide(position)

(Pour chaque nouvelle tâche, K déplace les objets sur le damier pour les états initials)

Pour qu'il fasse les déplacements soi-même, il faut qu'il comprenne les objets qui existent et qu'il peut déplacer. Donc, il faut lui expliquer les objets qui existent et que tu vois sur la table dans un langage qu'il comprend. On va faire la premiere exemple ensemble.

1. Echauffement: Voici un premier état initial.

K met le cube rouge sur le damier à la position 1a.

1.1 Types: - cube

- Quels sont les noms d'objet que tu vois sur le damier?
- Quel est le type de l'objet X1?

Le type est une façon de décrire l'objet X1 d'une manière plus généralisée.

1.2 Propriétés (initial): - cube

- Quelle est la propriété de l'objet X1 que tu observes dans cet état sur le damier?
    - Si S ne dit pas, K propose de regarder la couleur (ou la position)

Dans le langage que comprend le robot, on écrit

- 'est_rouge(X1)' pour dire 'X1 est rouge'
- 'est_sur(X1,1a)' pour dire 'X1 est sur 1a'

'verbe' ('sujet', 'compléments'): Le verbe est au début, puis le sujet, et à la fin il y a des compléments.

1.3 Propriétés (généralisée): - cube

- Comment peux-tu exprimer cette propriété dans une facon plus généralisé pour qu'on peut l'utiliser pour n'importe quel objet de ce type?

- K demande "qu'est-ce qui manque encore?" (position/couleur)

1.2. Types/Propriétés: - position

- Est-ce que le tableau est complet?
    - *Si S dit oui, K demandes "Est-ce que tu vois d'autre noms/types sur le damier?"*

- Remplissez le tableau (nom, type). (K continue à guider S)
- Quelles sont leurs propriétés?
    - Si S n'y arrive pas, *K guide "Quelle est la différence entre la position 1a et les autres?"*
- Propriété généralisée?
    - Si S n'y arrive pas, *K "Comparez-la avec les propriétés généralisées qu'on a déjà rempli"*

1.3 Actions: *K donne à S la fiche d'action I avec le tableau vide*
Maintenant on va voir comment on peut modéliser une action. Observez les trois graphiques.

Une action consiste des préconditions, c'est-à-dire des conditions qui sont nécessaire pour effectuer l'action, et des effets, les conditions qui s'appliquent après l'action.

1.3.1: l'action dans le langage: déplace**(X1,1a,2b)**

- Quel est l'action qui est effectuée?
    - Si S n'y arrive pas, K *"On peut dire que le cube est déplacé de 1a à 2b"*
- Comment peut-on représenter cette action dans le langage? Remplissez la case 'action'.

déplace**(X1,1a,2b):** *si S n'écrit pas toutes les paramètres X1, 1a, 2b - p.ex. déplacer(X1):*

- Si on lis que l'action, est-ce qu'on peut comprendre ce qu'il faut faire? (de déplacer de 1a à 2b)
- Qu'est-ce que cette action que tu viens d'écrire veut dire en français?
- Qu'est-ce qu'il manque pour que l'action dise "*déplacé X1 de 1a à 2b"?*

1.3.2: préconditions: est_sur(X1,1a)

- Quelles sont les conditions nécessaire pour déplacer le cube? Observez l'état initial.

Si S donne la bonne réponse: est_sur(X1,1a)

- K explique: C'est exacte, pour déplacer le cube X1 de 1a *à 2b, il faut que X1 soit d'abord sur 1a.*

Si S donne plus de conditions que nécessaire:

- Est-ce qu'on a besoin de toutes les conditions pour déplacer le cube?
- Est-ce qu'il est nécessaire que X1 soit rouge pour déplacer le cube X1*?*

1.3.3: effets: est_sur(X1,2b)

- Quelles sont les effets après l'action a été effectuée?
- Qu'est-ce qu'il *change* après le déplacement?

1.4 Action généralisée:

De la même manière qu'on a déjà fait dans le tableau, on veut généraliser l'action.

- Que seraient l'action généralisée pour qu'on puisse utiliser cette action pour n'import quel objet de ce type?
    - *Si S n'y arrive pas, K propose "Comment on a généralisé la propriété dans le tableau?"*

- Que seraient les conditions généralisées?
    - *Si S n'y arrive pas, K propose "Qu'est-ce qu'on peut généraliser? Comparez avec le tableau"*

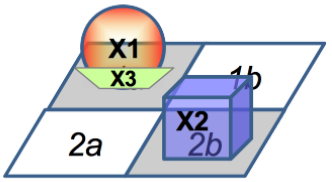
1.5 Debriefing:

- Qu'est-que vous pensez de ce langage - les types, les propriétés et leur généralisation *?*
- Qu'est-que vous pensez de la représentation de l'action avec les préconditions et les effets?
- Est-ce que la représentation utilisée, vous semble correcte ?
- Est-elle claire ? Facile à comprendre ?
- Quelles améliorations apporteriez-vous ?

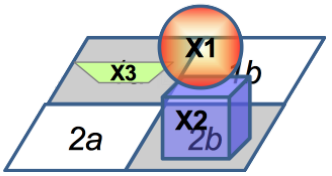
Niveau 1 – Tâche 2 - ajouter des contraintes nécessaire pour l'action

Nom d'objet	Type	Propriété (état initial)	Propriété généralisée
X3	**tasse**	**est_sur(X3, 1a)**	**est_sur(tasse, position)**
X1	**balle**	**est_bleue (X1)**	**est_bleue(balle)**
		est_sur(X1,X3)	**est_sur(balle,tasse)**
X2	cube	est_marron(X2)	est_marron(cube)
		est_sur(X2,2b)	est_sur(cube,position)
1a	position	non_vide(1a)	non_vide(position)
1b	position	est_vide(1b)	est_vide(position)
2a	position	est_vide(2a)	est_vide(position)
2b	position	non_vide(2b)	non_vide(position)

État initial



État final



Considérons maintenant un nouveau scénario.

K met les objets sur le damier.

2.1 Types/Propriétés:

- Observez ce nouvel état. Est-ce que le tableau est toujours complet/correct? Corrigez et complétez le tableau.
  - Si le tableau est faux ou incomplet, K demande "Est-ce que tu es sûr que ..."
  - Si le tableau est incomplet, K demande "Et la position du..."/"La propriété de...?"

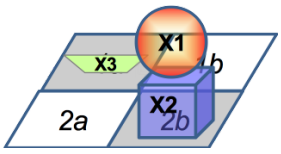
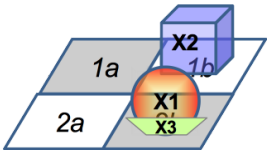
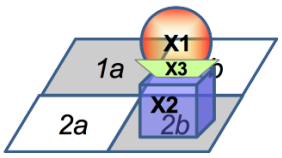
K donne à S la fiche d'action II.

2.2 Actions:

Imaginons on veut déplacer la balle vers la position 2b. Modéliser cette action comme tout-à-l'heure.

3 Sous-tâches:

- Est-ce qu'on peut utiliser cette action de déplacement?
  - Si S dit oui, K lui demande "Montrez-moi l'action de déplacement" - **Tâche 2a**

Tâche 2a : est_empilable(balle,cube)	**Tâche 2b : est_vide(position_arrivée)**	**Tâche 2c : est_empilable(tasse,cube)**
État final 	État final 	État final 

Tâche 2a : **"Montrez-moi l'action de déplacement"**

La balle ne va pas tenir sur le cube.

- Qu'est-ce qui ne va pas?
  - Si S ne trouve pas la bonne réponse, K dit "La balle ne tient pas sur le cube"

Donc, on ne peut pas utiliser l'action de déplacement dans ce cas.

K pose des questions qui guident S:

- Qu'est-ce qu'on peut ajouter pour qu'il déplace la balle seulement si elle tient sur le cube?
- Autrement dit, fais l'action seulement si la balle tient sur le cube. Donc, c'est une contrainte sur les propriétés des objets
- Il s'agit de quelle propriété entre la balle et le cube?
- Comment peut-on ajouter cette contrainte en tant que condition pour l'action?

Si S n'y arrive pas, K propose

- Peut-être on peut ajouter une précondition est_empilable de la même façon?
- En utilisant le langage avec le verbe/sujet, comment ça s'écrit?

Finalement,

- Quelle est cette condition généralisée?
    - est_empilable(balle,cube)

K explique "Donc, on a créé une action qui a deux condition pour déplacer la balle X1

1. la balle X1 est sur la position 1a
2. **la balle est empilable sur le cube"**

Transition à 2b/2c:

- On veut quand-même que la balle soit à la position 2b. Qu'est-ce qu'on peut faire?

Tâche 2b : **"Il faut déplacer le cube"**

- Quelle est la condition que l'on obtient si on déplace le cube?
    - S peut proposer "Autrement dit, la case est vide."
- Quelle est la condition que nous permet de déplacer la balle?
- Qu'est-ce qu'il faut ajouter?
- Quelle est sa formalisation généralisée?

K explique "Donc, on a créé une action qui a deux condition pour déplacer la balle X1

3. la balle X1 est sur la position 1a
4. **la position d'arrivée 2b est vide"**

! On peut enlever la condition est_sur(X2,2b) qui dit que le cube X2 est sur 2b.

Transition à 2c:

- Qu'est-ce qu'on peut faire pour que la balle soit sur la position 2b sans déplacer le cube/ **au-dessus du cube**?

Tâche 2c : **"Il faut déplacer la tasse avec la balle"**

- Pourquoi peut-on déplacer la tasse sur le cube?
- Comment représenter dans notre langage que l'on peut mettre la tasse sur le cube, mais pas la balle?
- Quelle est la condition que l'on peut ajouter?

Finalement,

- Quelle est sa formalisation généralisée?
    - est_empilable(tasse, cube)

K explique "Donc, on a créé une action qui a deux condition pour déplacer la balle X1

5. la balle X1 est sur la tasse X3
6. **la tasse X3 est empilable sur le cube"**

Transition à 2b:

- Et si on pourrait pas empiler les objets, qu'est-ce qu'on peut faire pour que la balle soit à la position 2b?

Debriefing:

- Qu'est-que vous pensez de ce langage - les types, les propriétés et leur généralisation ?
- Qu'est-que vous pensez de la représentation de l'action avec les préconditions et les effets?
- Est-ce que la représentation utilisée, vous semble correcte ?
- Est-elle claire ? Facile à comprendre ?
- Quelles améliorations apporteriez-vous ?

Niveau 1 - Enchainement des taches

Considérons une nouvelle configuration. Vous avez un état initial.

- Définissez le but.
- Quelles sont les actions pour arriver de l'état initial au but? Montrez-les sur le damier.
- Voici, les actions qui sont disponibles. Quelles sont les actions dont on a besoin?
- Remplissez les actions et leurs préconditions et effets et mettez-les dans la bonne ordre.
- Montrez le lien entre les actions (leur précondition/effets)
    - Pourquoi avez-vous choisi les deux actions?

déplacer(cube,position,position)
déplacer(X1,1a,2b)

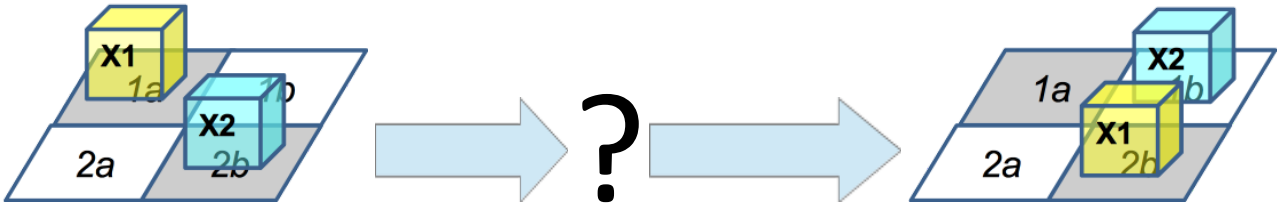
déplacer(X2,2b,1b)
déplacer(X2,2b,2a)

préconditions	action	effets
est_sur(___ , ___)	déplacer(_X2____, ___2b_, __1b_____)	est_sur(___, ___)
est_vide(___)		est_vide(___)

préconditions	action	effets
est_sur(___ , ___)	déplacer(_X2____, ___2b_, __1b_____)	est_sur(___, ___)
est_vide(___)		est_vide(___)

État initial

But



état initial		But
est_sur(X1, 1a)		est_sur(X1, 2b)
est_sur(X2,2b)		est_sur(X2,1b)
non_empilable(X1,X2)		
non_vide(1a)		est_vide(1a)
est_vide(1b)		non_vide(1b)
est_vide(2a)		est_vide(2a)
non_vide(2b)		non_vide(2b)

Niveau 2

K donne un tableau vide et repositionne les blocs sur le damier
Essayez de remplir le tableau à partir du nouveau scénario.