# Vers une gestion des croyances pour la planification Homme - Robot

Julien Guitton, Matthieu Warnier, and Rachid Alami[1,2]

[1] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France;

[2] Univ de Toulouse, LAAS, F-31400 Toulouse, France;

email: julien.guitton@laas.fr

**Résumé** : Cet article présente une extension d'une approche de planification hiérarchique pour la gestion des problèmes multi-agents et, plus spécialement, pour les problèmes d'interaction Homme-Robot dans lesquels un robot et un humain doivent coopérer pour atteindre un but commun. La méthode proposée permet de raisonner et de planifier pour des agents qui ont des croyances différentes ou incomplètes. Elle s'appuie sur une nouvelle description des croyances des agents et sur un mécanisme permettant de produire et d'insérer des actions de communication dans le plan courant. Ces actions de communication ont pour fonction de transmettre des informations d'un agent à l'autre.

## 1 Introduction

Un agent interagissant dans un environnement commun avec ses partenaires ne doit pas seulement raisonner sur ses propres capacités mais également sur les capacités des autres agents afin de réaliser des tâches de façon collaborative. Dans le contexte de l'Interaction Homme-Robot (HRI), le robot doit raisonner sur les connaissances de l'humain : le robot et l'humain peuvent ne pas avoir les mêmes informations sur l'environnement et les objets le composant. Ce raisonnement est compliqué par le fait que le robot ne connait pas parfaitement le modèle cognitif de l'homme. En effet, hormis à travers des phases de dialogue, le robot infère les connaissances de son partenaire uniquement grâce à sa capacité de changement de perspective (perspective taking).

À partir de cette connaissance, qui peut être différente de la sienne ou incomplète, le robot doit produire un plan pour lui-même et pour l'homme afin de réaliser un but commun. Ce plan doit être précis et surtout compréhensible par l'homme.

Dans nos précédents travaux, nous avons présenté le planificateur HATP (Alili *et al.*, 2008), pour Human Aware Task Planner, qui s'appuie sur une technique de planification hiérarchique, combinée avec un ensemble de règles de comportement qui guident les décisions du robot et permettent des plans dits socialement acceptables pour les êtres humains. Dans cet article, nous améliorons ce planificateur afin de traiter les problèmes d'interaction homme-robot dans lesquels le robot et l'homme peuvent avoir des croyances divergentes ou incomplètes. Nous appelons cette extension *gestion des croyances*.

Dans la partie suivante, nous présentons certains travaux existants sur la prise en compte de l'homme lors de l'élaboration d'architectures robotiques ainsi que sur planification de tâches collaboratives entre un robot et un humain. Puis, dans la section 3, nous présentons le planificateur HATP qui est le support de nos travaux. Dans la section 4, nous proposons une extension du formalisme de HATP pour la prise en compte des croyances des différents agents ainsi que l'algorithmique permettant de planifier avec ces croyances. Dans la section 5, nous présentons l'intégration de cette proposition dans notre plateforme robotique ainsi que les différents modules de cette plateforme permettant d'acquérir et de gérer les connaissances des agents. Finalement, dans la section 6, nous illustrons le processus de planification avec gestion des croyances à travers des scénarios en situation réelle.

# 2 Contexte et travaux associés

Ces dernières années, l'interaction homme-robot est devenue un champ de recherche actif dans de nombreuses disciplines et à différents niveaux. Par exemple, pour les chercheurs en sociologie, la tendance actuelle consiste à évaluer les réactions des humains interagissant avec des robots dans le but de concevoir des architectures robotiques plus conviviales (Hiolle *et al.*, 2009).

En robotique, l'humain est pris en compte à différents niveaux tels que la couche perception au travers de travaux sur le changement de perspective (Hiatt *et al.*, 2004; Trafton *et al.*, 2005; Sisbot *et al.*, 2011) ou encore la couche fonctionnelle afin d'adopter un comportement socialement acceptable durant les déplacements du robots en considérant l'humain comme étant plus qu'un obstacle à éviter (Pandey & Alami, 2009).

Une autre tendance en robotique et interaction homme-robot est de développer des architectures cognitives se rapprochant le plus possible du modèle cognitif humain (Cassimatis, 2002; Kerias & Mayer, 1997; Hiatt & Trafton, 2010). L'idée derrière ces architectures est d'embarquer à bord du robot une théorie de l'esprit (Baron-Cohen, 1991), *i.e.* fournir au robot la capacité d'inférer et de comprendre les croyances, désirs et intentions des autres agents à partir de ses observations.

Au niveau décisionnel, les travaux sur la planification pour l'interaction homme-robot ont suivi deux voies. La première approche concerne la planification par initiative mixte (Myers *et al.*, 2002; Bresina *et al.*, 2005) qui permet de placer l'humain dans la boucle : il peut contrôler la construction d'un plan tandis que le planificateur l'assiste dans sa prise de décision. L'autre approche est appelée planification continue et s'appuie sur l'idée d'acquisition active de connaissances (Knoblock, 1995) : le robot ne planifie pas seulement pour atteindre un but mais également pour acquérir les connaissances nécessaires à la réalisation de ce but. La planification continue entrelace planification et exécution pour compenser le manque d'information d'un cycle de planification à l'autre.

Dans nos travaux, nous considérons l'humain uniquement au niveau délibératif, *i.e.,* durant la planification. Contrairement à la planification continue, afin d'éviter des re-planifications et pour produire des plans compréhensibles, notre algorithme raisonne non seulement sur les croyances que le robot a sur son environnement mais également sur ses croyances concernant les croyances de l'homme. Lorsque les croyances propres du robot sont incomplètes, l'algorithme se comporte tel un algorithme de planification continue, en produisant un plan pour acquérir les informations manquantes et en essayant de résoudre à nouveau le but.

# 3 Le planificateur HATP

HATP (Human-Aware Task Planner) est un planificateur hiérarchique. Le but de la planification hiérarchique (HTN) est de décomposer une tâche de haut niveau représentant le but en sous-tâches jusqu'à atteindre un ensemble de tâches atomiques qui sont réalisables par les agents (Nau *et al.*, 2003). HATP est capable de produire des plans pour le robot aussi bien que pour les autres participants (humains ou robots). Le comportement du planificateur peut être modifié au travers du réglage des différents coûts des actions et en prenant en compte un ensemble de règles sociales. Ces coûts et règles permettent d'adapter le comportement du robot en fonction du niveau de coopération désiré.

## 3.1 Agents et séquences d'actions

Le robot ne planifie pas uniquement pour lui-même mais également pour les autres agents. Le plan produit, appelé *plan partagé*, est un ensemble d'actions formant une séquence pour chaque agent impliqué dans la réalisation du but commun. En fonction du contexte, certains plans partagés contiennent des liens causaux entre les actions des différents agents. Par exemple, le second agent peut devoir attendre la fin de la réalisation d'une action du premier agent avant de pouvoir commencer sa propre action. Lorsque le plan est exécuté, les liens causaux induisent des synchronisations entre agents. La figure 1 illustre un plan avec deux séquences d'actions.

## 3.2 Coûts des actions et règles sociales

À chaque action est associée une fonction de coût et une fonction de durée. La fonction de durée produit un intervalle temporel pour l'exécution de l'action et est utilisée, d'une part, comme une ligne de temps

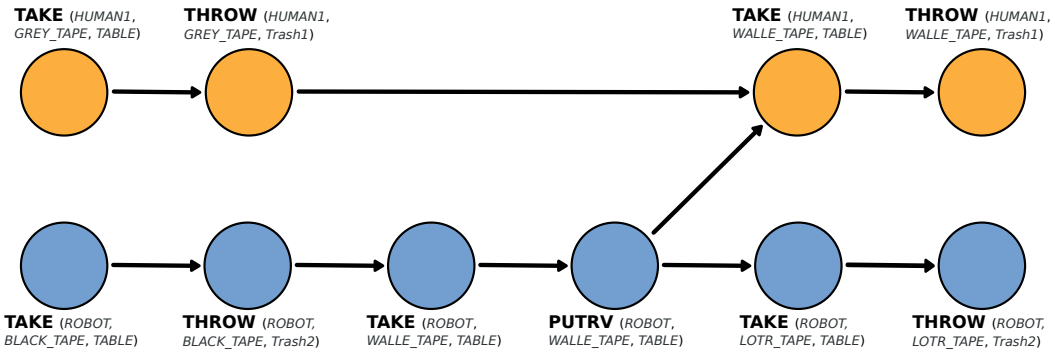



FIG. 1 – Un plan à 2 séquences d'action produit par HATP

pour ordonner les séquences d'actions des différents agents et, d'autre part, comme fonction de coût additionnelle. En plus de ces coûts, HATP prend en entrée un ensemble de règles sociales. Ces règles sont des contraintes qui ont pour but de guider la construction du plan vers le meilleur plan selon les préférences de l'homme. Les principales règles sociales que nous avons définies sont :

- état indésirable. Pour éviter un état dans lequel l'humain peut se sentir en position inconfortable ;
- séquence indésirable. Pour éliminer les séquences d'actions qui peuvent être mal interprétées ou rejetées par l'homme ;
- balance d'effort. Pour répartir et ajuster l'effort de travail entre les agents ;
- temps morts. Pour éviter les temps morts entre les actions de l'homme ou d'un autre agent ;
- liens intriqués. Pour limiter les dépendances entre les actions des différents agents.

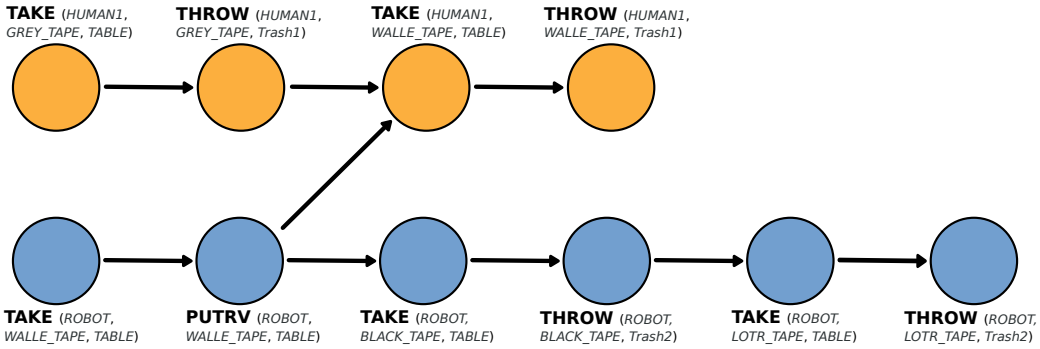



FIG. 2 – Le même plan avec la règle sociale des temps morts

La figure 2 illustre un plan alternatif au plan précédent (figure 1) lorsque la règle sociale concernant les temps morts est utilisée. Le plan produit est le meilleur plan en accord avec une évaluation globale des différents critères de coût. On remarque que pour appliquer cette règle, le planificateur a dû modifier l'ordre des actions du robot.

## 3.3 Différents niveaux de coopération

En réglant les fonctions de coûts et en appliquant les règles sociales, HATP peut être utilisé pour calculer de nombreux plans alternatifs. Ces plans peuvent être catégorisés en différents niveaux de coopération :

- aider l'humain à accomplir son objectif en agissant à sa place ;
- partager des ressources concrètes en donnant des objets à l'humain ;
- coopérer avec l'humain en coordonnant ses actions avec les actions du robot dans le but d'atteindre un but commun.

## 3.4 Modélisation du domaine

HATP utilise son propre langage orienté objet pour la modélisation du domaine. Ce langage a au moins le même pouvoir d'expressivité et les mêmes fonctionnalités que SHOP2 (Nau *et al.*, 2003).

### 3.4.1 Représentation orientée objet

Le monde est représenté par un ensemble d'entités. Chaque entité est unique et est définie par un ensemble d'attributs. Les attributs sont soit statiques (*static*), soit dynamiques (*dynamic*) et ont le type *atom* ou *vector*. Un attribut statique représente une information non modifiable tandis qu'un attribut dynamique peut être mis à jour. Un attribut atomique (*atom*) ne peut contenir qu'une seule valeur alors qu'un attribut de type *vector* est utilisé pour stocker des listes de valeurs.

### 3.4.2 Représentation des agents

Dans HATP, les agents sont considérés comme étant des objets. Cependant, comme la sortie du planificateur est un plan sous la forme de séquences d'actions par agent, le type d'entité *Agent* est prédéfini et au moins une entité *Agent* doit être initialisée.

### 3.4.3 Définition du domaine

Le domaine de planification, appelé base de faits (*fact database*) dans HATP, est défini en quatre étapes telles qu'illustrées par la figure 3. Premièrement, les différents types d'entités sont définis (excepté pour le type *Agent* qui est implicite). Puis, les attributs de chaque entité sont définis. Dans la troisième étape, les objets et agents présents dans l'environnement sont créés. Finalement, des valeurs initiales sont attribuées aux attributs de chaque entité créée.

```
factdatabase {
  //step 1: Definition of entity types
  define entityType Container;
  define entityType GameArtifact;
  //step 2: Definition of attributes
  define entityAttributes Agent {
    static atom string type;
    dynamic atom GameArtifact hasInRightHand;
  }
  define entityAttributes Container {
    dynamic set Agent isReachableBy;
  }
  define entityAttributes GameArtifact {
    dynamic set Agent isReachableBy;
    dynamic atom Container location;
  }
  //step 3: Creation of entities
  JIDO = new Agent;
  PINK_TRASHBIN = new Container;
  WHITE_TAPE = new GameArtifact;
  //step 4: Attributes initialization
  JIDO.type = "robot";
  PINK_TRASHBIN.isReachableBy <<= JIDO;
  WHITE_TAPE.isReachableBy <<= JIDO;
  WHITE_TAPE.location = PINK_TRASHBIN;
}
```

FIG. 3 – Exemple de définition d'un domaine de planification

# 4 Gestion des croyances

La description du monde telle qu'elle a été précédemment présentée suppose que l'état courant est entièrement connu et que tous les agents partagent la même vision du monde. Pour les applications réelles, et plus particulièrement pour les problèmes d'interaction homme-Rrbot, ces assertions peuvent mener à des solutions impossibles ou des plans incompréhensibles pour le partenaire humain.

Afin d'éviter ce genre de problèmes, nous proposons d'étendre cette représentation en ajoutant, d'une part, la possibilité de modéliser des croyances différentes pour chaque agent et, d'autre part, la possibilité de considérer qu'un agent connait ou ne connait pas certaines informations.

## 4.1 Modélisation des états de croyances

Dans nos expérimentations d'interaction homme-robot, la solution à un but donné est entièrement calculée par le robot. La base de faits doit modéliser l'état du monde du point de vue du robot ainsi que les croyances du robots sur les croyances de l'homme (et plus généralement, des autres agents).

### 4.1.1 L'agent myself

Pour spécifier quel agent est le robot, *i.e.,* l'agent pour lequel le système planifie, nous utilisons le mot-clé **myself** au lieu de déclarer un nouvel agent. Par exemple, si JIDO est le robot et ACHILE est un humain, l'initialisation des agents sera :

```
JIDO = myself;
ACHILE = new Agent;
```

### 4.1.2 Représentation des croyances

Afin de modéliser des croyances différentes pour les agents, le formalisme utilisé par HATP est étendu en utilisant la notion de variable d'état à valeurs multiples (MVSV). Une variable d'état à valeurs multiples $V$ est instanciée à partir d'un domaine $Dom$ et pour chaque agent $a \in A$ la variable V a une instance $V(a) \in Dom$. Par exemple, si les agents ont une croyance différente de la position de l'objet WHITE_TAPE :

```
WHITE_TAPE(JIDO).location = PINK_TRASHBIN;
WHITE_TAPE(ACHILE).location = BLUE_TRASHBIN;
```

Par défaut, afin de clarifier le domaine de planification, seuls les attributs des entités pour lesquelles les agents ont une croyance divergente sont modélisés en utilisant le formalisme MVSV.

### 4.1.3 Informations connues et inconnues

Le modèle de croyances des agents inclut les notions d'information connue et d'information inconnue. Lorsqu'un agent n'a pas d'information sur la valeur d'un attribut, la variable associée reçoit l'instance *unknown*. Lorsqu'un agent différent de l'agent **myself** connait une information qui est inconnue du robot, la valeur de l'attribut correspondant est *known*.

La représentation précédente des croyances des agents est augmentée afin de prendre en compte ces valeurs spécifiques :

$$V(a) \in \begin{cases} Dom_v \sqcup \{unknown\} & \text{si} \quad a = \text{myself} \\ Dom_v \sqcup \{unknown\} \sqcup \{known\} & \text{autrement} \end{cases}$$

Par exemple, si l'homme ne connait pas la position de l'objet WHITE_TAPE :

```
WHITE_TAPE(JIDO).location = PINK_TRASHBIN;
WHITE_TAPE(ACHILE).location = unknown;
```

Lorsqu'un agent n'a aucune information sur un objet, tous les attributs de l'entité correspondante devraient prendre la valeur *unknown*. Nous simplifions cette représentation par :

```
WHITE_TAPE(ACHILE) = unknown;
```

### 4.1.4 Consistance des croyances

Pour être consistante, une variable d'état $V$ requiert que l'union des croyances des agents sur la propriété associée forme un ensemble de dimension 1. C'est-à-dire, une croyance est consistante si l'ensemble des agents ont la même croyance sur la propriété du monde concernée.

$$\| \bigcup_{\forall a \in Ag} V(a) \| = 1$$

Cette formule est utilisée durant le processus de planification afin d'assurer que le plan est correct du point de vue des croyances des agents. Ainsi, à la fin de l'exécution du plan, les agents doivent avoir les mêmes croyances sur les objets manipulés.

## 4.2 Mise à jour des croyances et communication

En planification classique, une action est définie par un ensemble de préconditions représentant les conditions nécessaires à sa réalisation et, un ensemble d'effets modélisant les changements du monde résultant de l'exécution de l'action.

Planifier pour plusieurs agents qui ont leur propres croyances soulève un ensemble de questions :
- les croyances de quel agent doit utiliser le système, surtout dans le cas d'une action jointe ?
- Les croyances des agents doivent-elles être consistantes avant l'exécution d'une action ? comment ?
- Comment évoluent les croyances des agents impliqués dans la réalisation d'une action jointe ?
- Comment évoluent les croyances des autres agents ?

### 4.2.1 Croyances et préconditions des actions

Afin de réaliser une action jointe, tous les participants doivent avoir les mêmes croyances sur les objets manipulés durant cette action. Pour assurer cette consistance, le planificateur produit des actions de communication entre l'agent principal (l'agent **myself**) et les autres participants à l'action.

### 4.2.2 Croyances et effets des actions

Comme les croyances des agents sur les objets manipulés doivent être consistantes avant d'exécuter correctement une action, les effets des actions sont appliqués sur les croyances de tous les participants. C'est-à-dire que les croyances des agents sur les objets manipulés restent consistantes après l'exécution de l'action. Concernant les croyances des agents ne participant pas à l'action mais pouvant être présent dans la scène, la mise à jour de leurs croyances n'est pas automatique et est de la responsabilité du concepteur du domaine de planification.

### 4.2.3 Actions de communication

Une action de communication est une action particulière qui a comme paramètres deux agents, l'émetteur et le récepteur, et un sujet qui est représenté par une entité et un attribut. Ainsi, le prototype d'une action de communication est le suivant :

```
commAction name(Agent A, Agent B, Entity E, Attribute T){
  preconditions {...};
  effects {...};
  cost{...};
  duration{...};
}
```

Le but d'une action de communication est de transmettre la valeur d'un attribut d'une entité de l'agent émetteur à l'agent récepteur. Ce qui correspond à l'effet suivant :

```
E(B).T = E(A).T;
```

Cet effet est implicite pour ce type d'action, *i.e.,* le concepteur du domaine de planification n'a pas besoin de le spécifier pour chaque action de communication.

| myself | Agent$_B$ | type de communication |
|---|---|---|
| $v$ | $unknown$ | information |
| $v$ | $v'$ | contradiction |
| $unknown$ | $known$ | question |

TAB. 1 – Types de communication en fonction des croyances

De la même façon qu'une action classique, une action de communication est définie par un ensemble de préconditions exprimant les conditions nécessaires à la communication (*e.g.,* les agents doivent être dans la même pièce), et un ensemble d'effets additionnels à l'effet implicite résultant de la communication. Avec ces effets, il est par exemple possible de modéliser le concept de co-présence, *i.e.,* la communication n'affecte pas seulement les croyances de l'agent récepteur mais également les croyances de tous les agents présents aux alentours.

Afin d'être cohérent avec la modélisation du domaine, pour transmettre l'ensemble des informations concernant une entité de l'agent A à l'agent B, le paramètre attribut peut prendre la valeur *null*. Dans tous les autres cas, seule la valeur de l'attribut spécifié est transmise.

#### 4.2.4 Types de communication

En fonction des croyances des agents, les actes de communication ne seront pas traités de la même manière au moment de l'exécution. Nous choisissons de faire cette distinction dès la planification par la définition de trois types d'actions de communication : *information*, *contradiction* et *question*.

Les actions de communication de type *information* ont pour objectif de transmettre une information de l'agent **myself** à un autre agent lorsque celui-ci ne connait pas cette information, *i.e.,* la valeur de l'attribut correspondant est *unknown* dans sa base de croyances.

Quand la valeur associée à un attribut pour un agent est différente de la valeur du même attribut pour l'agent **myself**, le planificateur produit une action de communication de type *contradiction*.

Le type *question* est utilisé lorsque l'agent **myself** ne connait pas une donnée et qu'il existe un autre agent qui a cette connaissance (modélisée par la valeur *known*).

Le tableau 1 résume ces différents types de communication en fonction des croyances d'un agent comparées aux croyances de l'agent **myself**.

### 4.3 Mise en œuvre et adaptation du planificateur

Pour pouvoir gérer les croyances des agents, l'algorithme de planification doit être adapté afin de prendre en compte le nouveau formalisme pour la modélisation du domaine ainsi que les actions de communication.

#### 4.3.1 Bases de faits

Afin de stocker les croyances des différents agents, nous créons une base de faits pour chaque agent. Durant la phase d'initialisation, les entités représentant les agents et objets présents dans l'environnement sont créées et stockées dans la base de faits de l'agent **myself**. Pour les autres agents, afin de limiter l'espace mémoire utilisé, nous décidons de stocker dans les bases de faits additionnelles uniquement les valeurs qui sont inconsistantes avec les croyances de l'agent principal.

#### 4.3.2 Méthode générale de communication

Pour permettre au concepteur du domaine de planification de nommer les actions de communication de la même manière qu'il le ferait pour les actions classiques, celles-ci sont liées aux concepts *information*, *contradiction* et *question* au travers d'une méthode générale de communication appelée **beliefManagement**.

```
beliefManagement {
  information { GiveInformationAbout; };
  contradiction { ForceInformation; };
  question { AskForInformation; };
}
```

Dans cet exemple, le type de communication *information* se réfère à l'action de communication appelée GiveInformationAbout par le concepteur du domaine. Chaque action de communication doit être définie précédemment dans le domaine de planification.

### 4.3.3 Processus de planification

L'algorithme principal d'un planificateur HTN consiste à développer l'arbre de planification, *i.e.,* à décomposer des tâches complexes en sous-tâches jusqu'à obtenir une séquence d'actions primitives exécutables par les agents. Le développement de l'arbre s'effectue par une recherche en profondeur d'abord et s'arrête lorsque la liste des tâches à décomposer est vide.

Cet algorithme est augmenté par le processus de gestion des croyances de la manière suivante :

```
1  Tree_develop(T):
2  t0 = T[0];
3  if(t0 is a primitive task) {
4    classical = false;
5    agent = the (main) agent achieving the action;
6    v(agent)= value of task arguments and preconditions;
7    forall(v(agent)) {
8      if(agent ≠ myself) {
9        if(v(agent) = unknown) {
10         c = make_action(beliefManagement.information);
11         T[0] = c;
12         Tree_develop(T);
13       }
14       else if(v(agent) ≠ v(myself)) {
15         c = make_action(beliefManagement.contradiction);
16         T[0] = c;
17         Tree_develop(T);
18       }
19       else {
20         classical = true;
21     }
22     else if(v(myself)=unknown and v(other agent)=known) {
23       c = make_action(beliefManagement.question);
24       T = empty;
25       plan = c;
26     }
27     else {
28       classical = true;
29     }
30   }
31   if(classical=true) {
32     // do the classical HTN treatment for primitive task
33   }
34 }
35 else {
36   // do the classical HTN treatment for compound task
37 }
```

Dans cet algorithme, si la tâche courante correspond à une action (l.3), les valeurs de chaque attribut de chaque entité liée à la tâche sont vérifiées. Si l'agent réalisant cette action n'est pas l'agent **myself** (l.8) et si la valeur d'un attribut est *unknown* (l.9) dans la base de faits de cet agent, alors une action de communication de type *information* est produite (l.10). Cette action de communication est insérée au début de la liste des tâches (l.11) et sera raffinée durant le prochain appel récursif de l'algorithme de planification (l.12).

De la même façon, si la valeur d'un attribut est différente de la valeur correspondante dans la base de faits de l'agent **myself** (l.14), une action de communication de type *contradiction* est produite (l.15) et insérée

au début de la liste des tâches.

Si l'agent réalisant l'action est l'agent **myself** et si la valeur de l'attribut considéré est *unknown* (l.22), l'algorithme vérifie qu'il existe un autre agent connaissant cette valeur. Si c'est le cas, l'algorithme de planification produit une action de communication de type *question* (l.23) et replace toutes les tâches en attente dans la liste des tâches par cette action (l.24 et l.25). En effet, seulement après l'exécution de cette action de communication, la base de connaissance de l'agent principal sera mise à jour et, donc, le planificateur pourra produire le reste des actions permettant d'atteindre le but courant.

# 5 Integration dans l'architecture robotique

HATP avec gestion des croyances a été intégré et testé dans notre architecture robotique. Dans cette section, nous donnons un aperçu de cette architecture et de certains des modules la composant.

## 5.1 Aperçu de l'architecture

Le robot est contrôlé par une architecture trois-tiers (Alami *et al.*, 1998). La figure 4 illustre la couche décisionnelle de cette architecture. Le cadre décisionnel proposé est composé de différents modules, chacun ayant un rôle spécifique et qui peuvent être liés aux trois activités principales du contrôleur du robot : 1. évaluation de la situation et gestion du contexte d'exécution, 2. gestion des buts et des plans, 3. actions, exécution et monitoring.

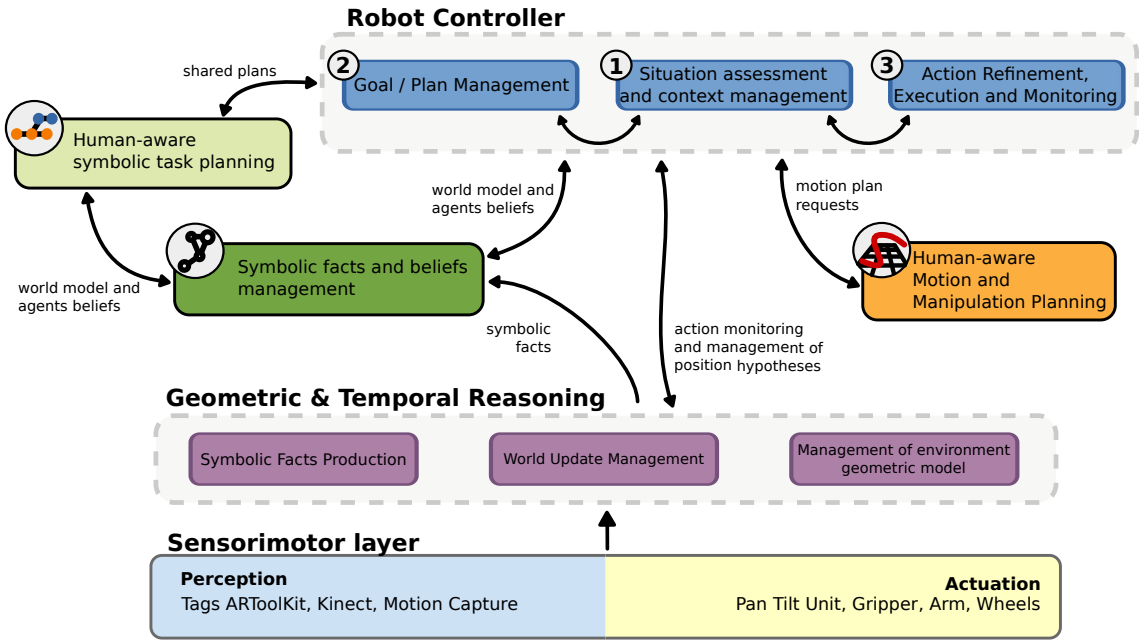



FIG. 4 – La couche décisionnelle de l'architecture robotique

Dans les prochains paragraphes, nous présentons certaines parties de ces activités qui permettent à HATP de recueillir les informations nécessaires pour produire un plan solution et, comment ce plan est exécuté.

## 5.2 Acquisition des connaissances

Le module de raisonnement géométrique que nous utilisons est appelé SPARK (pour SPAtial Reasoning and Knowledge) (Sisbot *et al.*, 2011). Ce module a la responsabilité de recueillir les informations géométriques sur l'environnement et, embarque un ensemble d'activités décisionnelles permettant des capacités d'abstraction et d'inférence à partir de raisonnements géométriques et temporels. SPARK maintient l'ensemble des positions et configurations géométriques des agents et objets à partir des données reçues du module de perception et à partir des connaissances précédentes ou *a priori*. L'état géométrique de l'environnement est abstrait en un ensemble de faits symboliques qui peut être directement utilisé par HATP.

Ces faits produits sont stockés dans une base de connaissances symboliques centrale, appelée ORO (Lemaignan *et al.*, 2010). ORO gère ces données dans des bases de connaissances distinctes pour chaque agent. Ainsi les croyances de chaque agent peuvent être stockées dans le modèle correspondant. Chacun de ces modèles est indépendant et logiquement consistant, permettant ainsi de raisonner à partir de différentes perspectives du monde qui seraient autrement considérées comme étant globalement inconsistantes (par exemple, un objet peut être visible par un agent mais pas par les autres).

### 5.3 Traitement d'un but, planification et exécution

L'objectif à atteindre est donné par le partenaire humain. Lorsqu'un évènement annonçant un nouveau but est détecté par le contrôleur du robot, la validité de ce but est tout d'abord testée : correspond-il aux capacités des agents ? N'a-t-il pas déjà été achevé ?

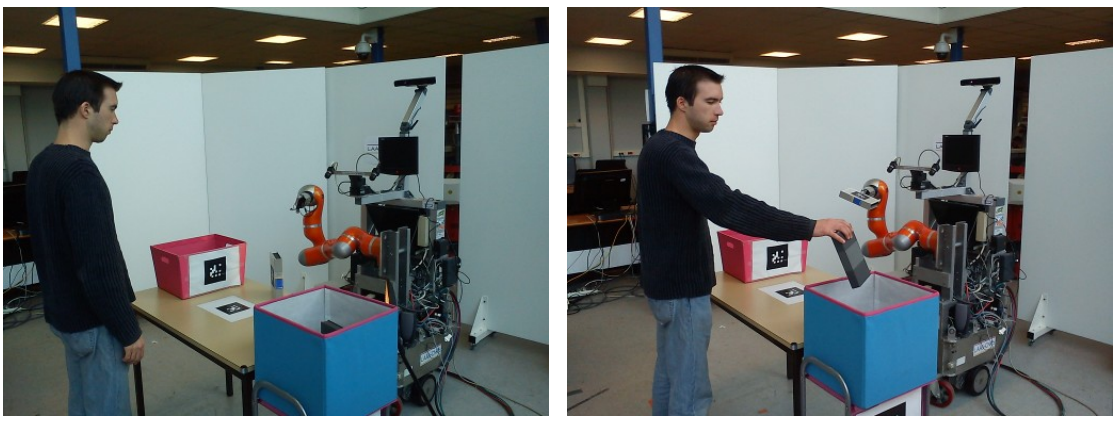
Puis ce but est envoyé à HATP qui acquiert l'état courant du monde, c'est-à-dire les croyances de chaque agent, à partir du module ORO et produit un premier plan permettant d'atteindre le but. En cas de replanification suite à une erreur d'exécution, ce but est considéré comme faisable tant que le planificateur produit un plan valide ou tant que l'exécution du plan n'est pas abandonnée par l'homme.

L'exécution du plan consiste en l'exécution de chaque action de ce plan. La gestion d'une action se déroule en trois étapes : Tout d'abord, les préconditions de l'action sont testées par le module de supervision. Puis l'action est exécutée et contrôlée ou seulement contrôlée s'il s'agit d'une action de l'homme. Finalement, les effets attendus sont vérifiés afin acquiter de la bonne exécution de l'action. En cas d'erreur d'exécution d'une action, détectée par le monitoring ou la vérification des effets, un nouveau plan pour le but courant est demandé puis exécuté à partir de la situation courante.

## 6 Trois exemples illustratifs

Afin d'illustrer le fonctionnement d'HATP avec gestion des croyances, nous détaillons trois exemples de réalisation d'une tâche jointe pour laquelle les croyances des agents sont incomplètes ou divergentes.

Pour les deux premiers exemples, deux agents, un robot (nommé JIDO_ROBOT) et un humain (nommé HERAKLES_HUMAN), doivent coopérer dans le but de nettoyer une table (EXP_TABLE). L'objectif est de mettre les deux cassettes (BLACK_TAPE et GREY_TAPE) dans la corbeille rose (PINK_TRASHBIN). La cassette grise est sur la table tandis que la cassette noire se trouve dans la corbeille bleue (BLUE_TRASHBIN). La figure 5(a) illustre la situation initiale.



(a) État initial de l'expérimentation (b) Durant l'exécution du plan

FIG. 5 – Représentation de l'environnement des exemples illustratifs

Dans ces deux scénarios, la corbeille rose est accessible par les deux agents, la corbeille bleue est accessible uniquement par l'homme et la cassette grise uniquement par le robot. L'accessibilité de la cassette noire est déduite de l'accessibilité de la corbeille bleue. Tous ces faits sont calculés automatiquement par le module SPARK et stockés dans ORO. L'état initial est défini de la façon suivante (excepté pour la cassette noire) :

```
BLUE_TRASHBIN.isReachableBy <<= HERAKLES_HUMAN;
PINK_TRASHBIN.isReachableBy <<= HERAKLES_HUMAN;
PINK_TRASHBIN.isReachableBy <<= JIDO_ROBOT;
GREY_TAPE.isVisibleBy <<= JIDO_ROBOT;
GREY_TAPE.isVisibleBy <<= HERAKLES_HUMAN;
GREY_TAPE.isReachableBy <<= JIDO_ROBOT;
GREY_TAPE.isOn = EXP_TABLE;
```

## 6.1 Premier exemple : Inconnu pour l'homme

Dans ce premier scénario, l'environnement a été mis en place durant l'absence de l'agent Herakles. Nous faisons l'hypothèse que, de sa position, l'homme ne peut pas voir le contenu de la corbeille bleue et donc la cassette noire, impliquant le fait que la position de cette cassette est inconnue pour lui. L'état initial est complété par (la valeur unknown est entrée manuellement dans ORO) :

```
BLACK_TAPE(JIDO_ROBOT).isIn = BLUE_TRASHBIN;
BLACK_TAPE(HERAKLES_HUMAN).isIn = unknown;
```

La figure 6 illustre le plan produit par HATP pour ce scénario. La première action est une action de communication de type *information*. Le robot informe l'humain que la cassette noire se trouve dans la corbeille bleue. Puis l'homme prend cette cassette et la jette dans la corbeille rose. De son côté, le robot doit saisir la cassette grise et la jeter.

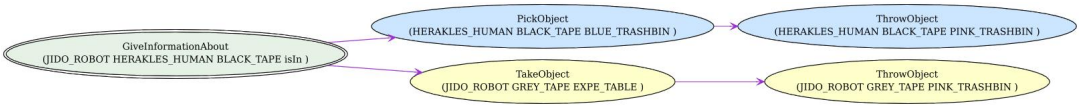



FIG. 6 – Plan produit pour le premier scénario. Le robot donne l'information de position de la cassette noire à l'homme.

## 6.2 Second exemple : Inconnu pour le robot

Dans cette seconde expérimentation, le système de perception du robot a été désactivé durant la mise en place de l'environnement. Pour Jido, la position de la cassette noire est inconnue. Herakles est le seul à savoir où se trouve cette cassette. L'état initial est complété par les faits (ajoutés manuellement) :

```
BLACK_TAPE(JIDO_ROBOT).isIn = unknown;
BLACK_TAPE(HERAKLES_HUMAN).isIn = known;
```

Concernant la cassette noire, comme le robot sait uniquement que l'homme sait ou elle se trouve, il ne peut pas produire de plans complets pour le but courant. En effet, l'information de position de la cassette est nécessaire à la validation des préconditions de l'action PickObject. Dans ce cas, HATP produit un plan contenant une seule action (figure 7) : une action de communication de type *question* permettant au robot d'obtenir la donnée manquante.


FIG. 7 – Le plan produit par HATP contient uniquement une action de communication permettant au robot d'obtenir l'information manquante.

Une fois que le robot a obtenu l'information sur la position de la cassette noire, un nouveau plan pour résoudre le but courant est demandé à HATP. La figure 8 illustre la sortie produite par HATP permettant de réaliser l'objectif.

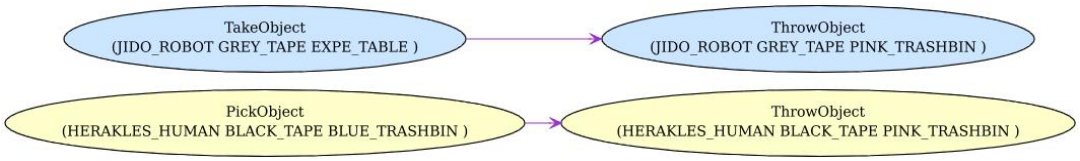



FIG. 8 – Plan produit par HATP après acquisition de l'information et re-planification.

## 6.3 Troisième exemple : Un scénario plus complexe

Dans ce troisième et dernier scénario, les conditions expérimentales sont légèrement différentes des expérimentations précédentes. Les deux agents doivent jeter les trois cassettes (BLACK_TAPE, GREY_TAPE et WHITE_TAPE) se trouvant sur la table, dans la corbeille rose. La cassette blanche est accessible uniquement par le robot tandis que les deux autres cassettes (grise et noire) sont accessibles uniquement par l'homme mais invisibles pour celui-ci, car cachées derrière des boites. Comme la corbeille rose est accessible uniquement par l'homme, celui-ci a la charge de jeter les trois cassettes.

La figure 9 est une capture d'écran du module SPARK et illustre l'état initial de ce scénario, c'est-à-dire la représentation de l'environnement du point de vue du robot.

L'agent humain, Herakles, n'a aucune information sur la position de la cassette noire et croit que la cassette grise se situe derrière la boite centrale (BOX1). Cette croyance est représentée par la boule verte. En réalité, la cassette grise est positionnée derrière l'autre boite (BOX2), tel que représenté sur la figure 9.

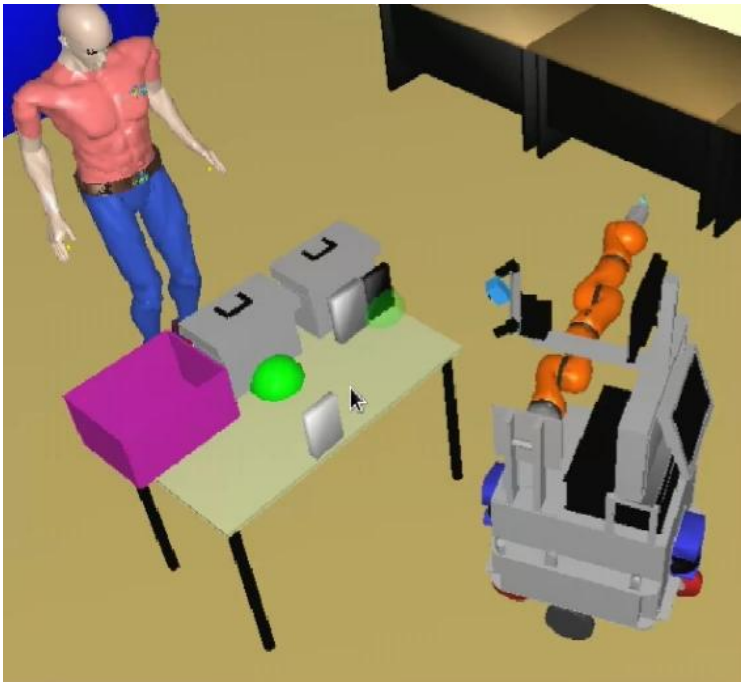


FIG. 9 – Vue de l'état initial calculée par le module SPARK.

Le plan produit par HATP correspond à celui donné par la figure 10. Ce plan contient deux actions de communication : une information sur la position de la cassette noire et une contradiction sur la position de la cassette grise.

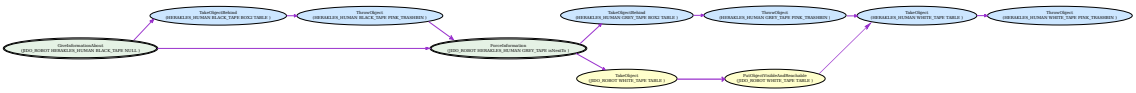
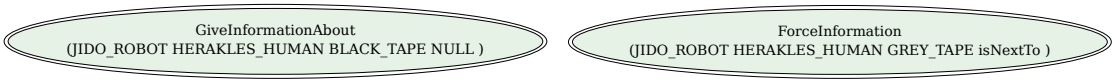


FIG. 10 – Plan produit par HATP pour le troisième scénario. Ce plan contient 2 actions de communication.




(a) (b)

FIG. 11 – Zoom sur les deux actions de communication produites.

On remarque que pour la communication d'information, l'attribut sur lequel s'effectue la communication a la valeur NULL. En effet, l'homme n'a aucune information sur la cassette grise et l'ensemble des valeurs caractérisant cet objet doit lui être transmis. Tandis que pour la contradiction, seule la position de la cassette relativement aux boites (isNextTo) doit être mise à jour.

# 7 Conclusion et perspectives

Dans cet article, nous avons présenté une première approche permettant à un planificateur hiérarchique de produire des plans valides et compréhensibles pour des agents ayant des croyances différentes ou incomplètes. Ce travail est basé sur une extension du formalisme de HATP permettant d'exprimer des croyances différentes, inconnue ou connue pour chaque agent et, sur la définition d'actions particulières : les actions de communication. Durant le processus de planification, lorsque les croyances d'un agent sont inconsistantes ou lorsque que celui-ci n'a pas les informations nécessaires à la réalisation d'une action, une action de communication de type contradiction, information ou question est produite et insérée dans le plan courant. Ces travaux ont été mis en œuvre au sein de l'architecture décisionnelle de l'un de nos robots assistants et testés sur des scénarios simples en conditions réelles.

Par la suite, nous allons travailler sur la déduction et la production automatique des faits *known* et *unknown* dans le module de reconnaissance de situations. Ces travaux nécessitent l'étude et la mise en place de modules de raisonnements géométriques et temporels additionnels. L'homme a-t-il vu le changement de l'environnement ? Que savait-il avant de quitter la pièce ? Quelles informations lui manquent-il en revenant dans la scène ? Par ailleurs, les actions de communications sont, pour l'instant, exécutées de façon orale. Nous aimerions étudier la possibilité d'utiliser d'autres modalités de communication (regard, geste, ...). Finalement, les humains ont une tendance à oublier ou ne pas accepter ce qui leur sont donnés comme vérités. Il serait intéressant d'étudier l'influence de ce phénomène sur la planification.

# Remerciements

This work has been conducted within the EU SAPHARI project (http ://www.saphari.eu/) funded by the E.C. Division FP7-IST under Contract ICT-287513.

# Références

ALAMI R., CHATILA R., GHALLAB M. & INGRAND F. (1998). An architecture for autonomy. *Int. Journal of Robotics Reasearch*, **17**(4), 315–337.

ALILI S., MONTREUIL V. & ALAMI R. (2008). HATP : Task planner for social behavior control in autonomous robotic systems for HRI. In *9th Int. Symposium on Distributed Autonomous Robotic Systems*.

BARON-COHEN S. (1991). Precursors to a theory of mind : understanding attention in others. *Natural theories of mind : Evolution, development and simulation of everyday mindreading*, p. 233–251.

BRESINA J., JONSSON A., MORIS P. & RAJAN K. (2005). Mixed-initiative activity planning for mars rovers. In *19th international conference on Artificial Intelligence*.

CASSIMATIS N. (2002). *A cognitive architecture for integrating multiple representation and inference schemes*. PhD thesis, MIT.

HIATT L. & TRAFTON J. (2010). A cognitive model of theory of mind. In *Proceedings of the 10th International Conference on Cognitive Modeling*, p. 91–96.

HIATT L., TRAFTON J., HARRISON A. & SCHULTZ A. (2004). A cognitive model for spatial perspective taking. In *the sixth International Conference on Cognitive Modeling*, p. 354–355.

HIOLLE A., BARDE K. & CANAMERO L. (2009). Assessing human reactions to different robot attachment profiles. In *18th IEEE int. symposium on Robot and Human Interactive Communication*, p. 251–256.

KERIAS D. & MAYER D. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, **12**, 391–438.

KNOBLOCK C. (1995). Planning, executing, sensing and replanning for information gathering. In *14th international joint conference on artificial intelligence*.

LEMAIGNAN S., ROS R., MOSENLECHNER L., ALAMI R. & BEETZ M. (2010). ORO, a knowledge management module for cognitive architectures in robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

MYERS K., TYSON W., WOVERTON M., JARVIS P., LEE T. & DESJARDINS M. (2002). PASSAT : A user-centric planning framework. In *Third international workshop on planning and scheduling for space*.

NAU D., AU T.-C., ILGHAMI O., KUTER U., MURDOCH J. W., WU D. & YAMAN F. (2003). SHOP2 : An HTN planning system. *Journal of Artificial Intelligence Research*, **20**, 380–404.

PANDEY A. & ALAMI R. (2009). A framework for adapting social conventions in a mobile robot motion in a human-centered environment. In *International Conference on Advanced Robotics*, p. 1–8.

SISBOT E., ROS R. & ALAMI R. (2011). Situation assessment for human-robot interaction. In *20th IEEE International Symposium in Robot and Human Interactive Communication*.

TRAFTON J., CASSIMATIS N., BUGAJSKA M., BROCK D., MINTZ F. & SCHULTZ A. (2005). Enabling effective human-robot interaction using perspective-taking in robots. *IEEE transactions on Systems, Man, and Cybernetics*, **35**(4), 460–471.