Modèles de tâche

Introduction : tâche, activité.

“Analyse de tâches”, “modèle de tâche”, “activité” sont des termes issus de l’ergonomie qui se sont propagés dans de nombreux domaines, du génie logiciel, de l’intelligence artificielle, de la robotique, pour aboutir au domaine du dialogue homme machine [Pierrel, 87]. Si dans leurs utilisations ces termes ne recouvrent pas toujours exactement les mêmes notions, l’objectif de base est cependant d’analyser, de représenter et de contrôler la suite des actes d’un utilisateur. Cette suite d’actes, réellement effectuée par l’utilisateur pour résoudre un problème, sera appelée *activité.* Elle n’est de fait connue et explicable qu’a posteriori, une fois le but atteint et vis-à-vis de l’objectif réellement visé.

Par convention nous distinguons :

- un *acte*, il est produit par un utilisateur. Il a pour effet de modifier le monde ou de changer un état mental,
- une *action*, elle est produite par la machine. Elle se décompose en opérations élémentaires relativement au programme implémenté. Il est évident que les actions définies en machine conditionnent les actes permis à l’utilisateur et leur donnent leur granularité.

La planification cognitive

Les recherches sur la planification ont été nourries par la psychologie cognitive et l’intelligence artificielle, les deux disciplines s’influençant d’ailleurs très fortement. Pour l’humain, il y a plusieurs façons de planifier son action pour résoudre un problème [Hoc, 89] :

- de manière ascendante,
- de manière descendante.

(a) la planification ascendante part de l’analyse détaillée de la situation et tente d’évoquer des plans déjà connus en les adaptant à cette situation. L’évocation des plans peut se faire à partir d’indices ou par analogie. Dans ABSTRIPS [Sacerdoti, 74] le robot apprend à hiérarchiser les détails des situations relevées au cours de son déplacement. Ces détails serviront plus tard d’indices de déclenchement du plan lorsqu’une situation identique se re-présente. Dans la 2ème méthode, l’analogie est suggérée par une mise en relation de SRT (Systèmes de Représentation des Traitements) dans des situations différentes.

(b) La planification descendante détaille un plan pour l’appliquer à une situation donnée, connaissant le but à atteindre. L’adaptation du plan à la situation peut se faire en particularisant certaines variables du plan. On distingue plusieurs variantes

• par décomposition du but et hiérarchisation en sous buts (à chaque sous but est associé un plan de résolution),

• en particularisant un but général à un but spécifique,

• en traitant les interférences entre buts lorsqu'ils ne peuvent être décomposés indépendamment.

La planification chez l'humain dépend de la nature du problème à résoudre. On distingue trois grands types de problèmes :

• les problèmes de *transformation d'états* : chaque fois qu'un problème peut se décrire comme une succession d'états déterminés à parcourir,

• les problèmes *d'induction de structures* : se ramènent à une recherche de relations dans un ensemble (par ex. compléter les termes d'une série connaissant les premiers éléments),

• les problèmes *de conception* : dans ce type de problème le but est imprécis voire complètement indéfini ou défini par un ensemble de contraintes (par ex. en conception architecturale, obtenir une pièce lumineuse et agréable).

Les stratégies de recherche ou de développement du plan dépendent de ces grands types de problèmes. On distingue :

• la stratégie *essais-erreurs* qui consiste à explorer le graphe d'états au moyen d'heuristiques ou avec des techniques d'optimisation, pour trouver le chemin (si possible le meilleur) qui permet d'atteindre le but,

• la stratégie *fins-moyens* avec laquelle on évalue à chaque pas, l'écart entre la situation courante et l'état but pour déterminer les opérateurs à mettre en œuvre pour réduire cet écart (c'est le GPS, General Problem Solver de Newell [Newell, 1959]),

• la stratégie de *test d'hypothèse* (GRI, General Rule Inducer de Simon et Lea [Simon, 1974]) dans laquelle des schémas d'hypothèses sont testés, dans l'ordre du général au particulier, selon une organisation en "espaces d'exemples", c'est-à-dire par focalisation puis par balayage (autrement dit en largeur d'abord puis en profondeur),

• la stratégie *opportuniste*, qui consiste à tenter des actions selon la situation de manière à se rapprocher de situations connues, si possible en se ramenant à un problème de transformation d'états. pour lesquelles on dispose déjà de solutions.

En IA, la représentation des plans se fait à l'aide d'arbres et/ou ou de graphes, de schéma [Bobrow et Norman, 1975] ou de *frame* [Minsky, 1975], ou enfin à l'aide de scripts [Shank et Abelson, 1977]. Il existe une abondante littérature en IA sur la planification. Hayes-Roth définit l'ensemble des éléments nécessaires à la planification que nous retiendrons en termes de synthèse :

(1) les unités d'exécution (procédures, agents) et les unités de décision (focalisation, contrôle),

(2) les abstractions de plan (intention, but, schémas, objets, stratégies, tactiques),

(3) l'unité de méta-planification (type de problème, contraintes sur le plan, heuristiques, critères d'évaluation des hypothèses),

(4) les unités de raffinement (démarche globale, but final, macro-instructions, actions),

(5) les connaissances du domaine.

La planification dans le dialogue

Le problème en DHM est idéalement de prédire les actes ou au moins de les situer dans un cadre référentiel précis pour orienter le dialogue, voire le planifier. Pour cela on parle, peut-être abusivement, de *modèle de tâche* (il serait peut-être mieux de parler de modèle d'activité sachant d'autre part que ce modèle dépend de l'utilisateur), pour représenter l'ensemble des connaissances nécessaires à la planification des actions de la machine.

Une *tâche* se définit comme la réalisation d'un but dans un contexte et selon une procédure dont la représentation et la précision varient en fonction des objectifs dirigeant l'analyse. C'est en effet du ressort de l'analyste de dire ce qu'il entend par tâche. Souvent il adopte un point de vue fonctionnaliste : c'est une représentation plus ou moins abstraite des actes potentiels d'un utilisateur. Le degré de granularité des actes est très variable, leur décomposition ou leur structuration est souvent arbitraire. La complexité des tâches dépend bien sûr des domaines d'application : elle va de la routine (contrôle de processus) à l'innovation (problèmes de conception) en passant par tous les stades de complexité intermédiaires. Dans le premier cas on conçoit bien qu'elle soit fortement structurée et que l'activité soit planifiable a priori, dans l'autre elle ne l'est pas du fait de l'imprévisibilité même des processus de création. Il est alors tentant pour le concepteur du dialogue de représenter des formes de comportements types pour canaliser et interpréter l'activité de l'utilisateur. Remarquons cependant que dans de nombreuses situations l'utilisateur ne suivra pas nécessairement l'ordonnancement prévu dans le modèle de tâche — même pour des tâches à risque et/ou très contraintes [Visser, 93]. En fait, ce modèle n'est donc utile à la machine que pour tenter de se représenter le sens des actions d'un utilisateur typique et planifier son propre comportement.

L'intérêt de la notion de tâches en DHM est double : pour l'analyse et pour la modélisation. Pour l'analyse cet intérêt réside dans la conception et l'évaluation des systèmes informatiques et de leurs interfaces. L'intérêt de modéliser la tâche est lui aussi double. Il est de servir de cadre à la conduite du dialogue (perspective inter-actionnelle) et d'aider à la résolution des problèmes de référence (en replaçant les énoncés dans un cadre pragmatique). Au niveau représentationnel en effet, il apparaît que le modèle de tâche peut servir de support à l'expression et à la représentation de la dynamique de l'interaction, comme base du modèle de dialogue. En particulier, le modèle de tâche est un support intéressant pour la représentation et la mise en œuvre des contraintes du dialogue issues du séquencement entre tâches et des conditions d'activation des tâches, ainsi que pour la gestion du partage des tâches entre le système et l'opérateur.

Pour le DHM nous pouvons retenir que :

(a) la tâche est une représentation abstraite des opérations logiquement possibles pour atteindre un objectif,

(b) l'activité est la séquence d'actes réellement effectués dans le contexte d'une tâche donnée,

(c) l'acte est effectué par l'utilisateur en vue de produire un effet dans le monde référentiel de la machine en réponse à (ou pour provoquer) des actions de la machine.

Enfin rappelons les présupposés énoncés dans le chapitre 1, paragraphe "Apprentissage, langue et dialogue : que retenir ?" qui visent à:

• rendre la machine apte à acquérir de nouvelles actions (ou scénarios),

• permettre une grande variété d'activités différentes pour une même tâche (pour respecter le principe d'ergonomie).

En pratique, on distingue deux types de modèles de tâche : les modèles explicites et les modèles implicites. Nous examinons ci-après leurs capacités respectives à satisfaire les contraintes ci-dessus, puis nous proposons un modèle possible.

Les modèles de tâche explicites

Un modèle de tâche explicite, appelé aussi *plan hiérarchique*, décrit de manière explicite la succession des actions (en termes d'effets, de conditions, d'ordonnancement) conduisant à un but donné. Ce modèle convient bien aux tâches dites de *routine*, car les tâches étant définies une fois pour toutes, le système peut parfaitement contrôler le déroulement d'un plan.

Classiquement les tâches sont hiérarchisées en plans et/ou sous plans ou sous tâches, scénarios, etc., jusqu'aux scripts qui instancient les actions élémentaires. Le script étant une structure rattachée à un plus haut niveau à un scénario et ainsi de suite jusqu'à la racine de la structure qui est la tâche. On peut ainsi obtenir une certaine récursivité des représentations. Le nombre de niveaux de décomposition peut être très variable d'une application à une autre. Pour Valot [Valot et al., 91], les scripts sont des schémas dont les attributs décrivent l'ensemble des informations nécessaires à l'exécution d'une tâche élémentaire (ouvrir un fichier, déplacer un objet…). Les scénarios sont définis comme des séquences probables de scripts. Ils sont décrits à l'aide de scores que l'on attribue aux différentes successions possibles de scripts au cours du dialogue. De la même façon, les plans et/ou les sous plans sont définis en termes de probabilités attachées aux diverses séquences de scénarios. Plusieurs formes de représentation sont possibles : arbre ET/OU, réseau ATN, réseau de Pétri, CLG [Moran, 81], GOMS [Card, 83], TAG [Payne, 86], MAD [Pierret-Golbreich, 89], ETAG [Tauber, 90], TKS [Johnson, 91], UAN [Hartson, 92], etc. A ces formes sont associées des méthodes de parcours qui conditionnent le processus de planification.

Le mode de représentation le plus habituel est statique : il décrit la combinatoire des actions qu'il est *théoriquement* possible d'enchaîner pour exécuter une tâche avec succès. Du côté formel, on utilise les notations suivantes pour écrire les relations entre actions :

- Succession fortuite dans le temps de deux actions

  $a_i > a_j$ (se lit a_j suit fortuitement a_i), par exemple : planter > pleuvoir,
- Succession finalisée de deux actions (l'une est le but de l'autre),

  $a_i >> a_j$ (se lit A_j est la raison de a_i), par ex. : planter >> récolter,
- Succession motivée ou entraînée par la situation,

  $a_j * a_i$ (se lit a_j accompli pour réaliser a_i), par ex. : (faire un trou) $*$ (planter),

  cette relation se lit aussi comme : a_j dérive de a_i si a_i est de plus haut niveau,
- Causalité (provoqué par les lois du monde ou par volonté),

  $a_i \bullet a_j$ (se lit a_i est cause de a_j), par ex. : faner • tomber)

Voici par exemple un modèle de tâche explicite (fig. 1) représenté par un graphe ET/OU :

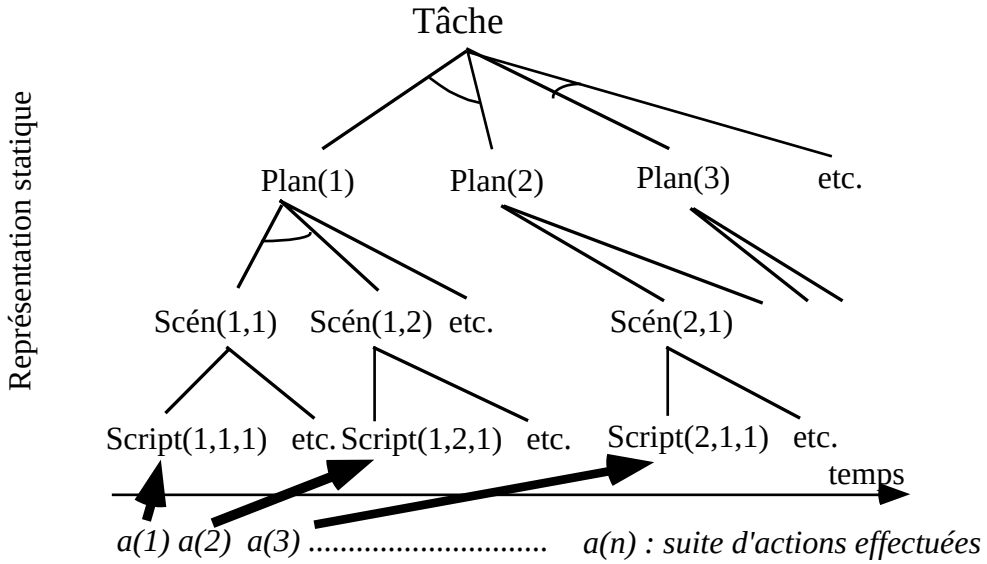


Fig. 1 : Décomposition arborescente de tâches par arbre ET/OU. Cette représentation se lit : pour faire la tâche il faut exécuter le Plan(1) ET le Plan(2), OU exécuter le plan(3). De même au niveau inférieur, pour exécuter le Plan(1) il faut exécuter les scénarios Scén(1,1) ET Scén(1,2), et ainsi de suite.

Cette représentation définit une action élémentaire a(n) comme une structure spécialisée instanciée (↵) à partir d'un script σ et effectuée à l'instant n. Cette action normalement appliquée avec succès à la situation $\xi(n-1)$, la fait passer à l'état $\xi(n)$ (la relation • se lit *cause-de*), ce qui se formalise par :

$$\forall\, a(n)\ \exists \sigma : a(n) \hookleftarrow \sigma,\ a(n) \bullet \xi(n)$$

Chaque action est activée par un ou plusieurs actes de l'utilisateur (verbal, gestuel, etc.). Il existe une pragmatique P qui associe à chaque action a une suite d'actes $\{\alpha\}$ de communication qui déclenche a :

$$\forall\, a\ \exists\, P : \{\alpha\} \bullet a$$

Tous les actes ne sont pas opérants vis-à-vis d'une action — il peut y avoir par exemple des actes méta-discursifs. Ils sont tels que : $\exists\alpha : \alpha \bullet ø$, où ø est une action nulle qui par définition appartient à $A = \{ a \}$

L'ensemble des actes ↵ de l'utilisateur est théoriquement illimité. On définit sur l'ensemble des actes ↵ la relation d'ordre partiel de séquentialité (≤) qui confère à toute sous séquence d'actes, une structure de demi-treillis inférieur. Cette relation (≤) permet de définir un ordonnancement des actes et donc, corrélativement, des actions déclenchées. Dans l'exemple de la fig. 2 nous avons :

$\{\alpha 1\}$• a(1) • ξ(1) avec a(1) ↵ σ6 ↵ s5

$\{\alpha 2 \leq \alpha 3\}$ • a(2) • ξ(2) avec a(2) ↵ σ5 ↵ s4

s5 ET s4 ∗ S2

$\{\alpha 5\}$• ø

$\{\alpha 4 \leq \alpha 6\}$ • a(3) • ξ(3) avec a(3) ↵σ3

$\{\alpha 7\}$•a(4) • ξ(4) avec a(4) ↵ σ2 ↵ s2

σ3 ET s2 ∗S1

S2 ET S1 ↵ P1

Si ξ(4) est la situation correspondant au but qu'il fallait atteindre, alors ce but est non seulement atteint mais il est satisfait. En conséquence on s'aperçoit que le locuteur a déroulé le plan P1 pour l'atteindre.

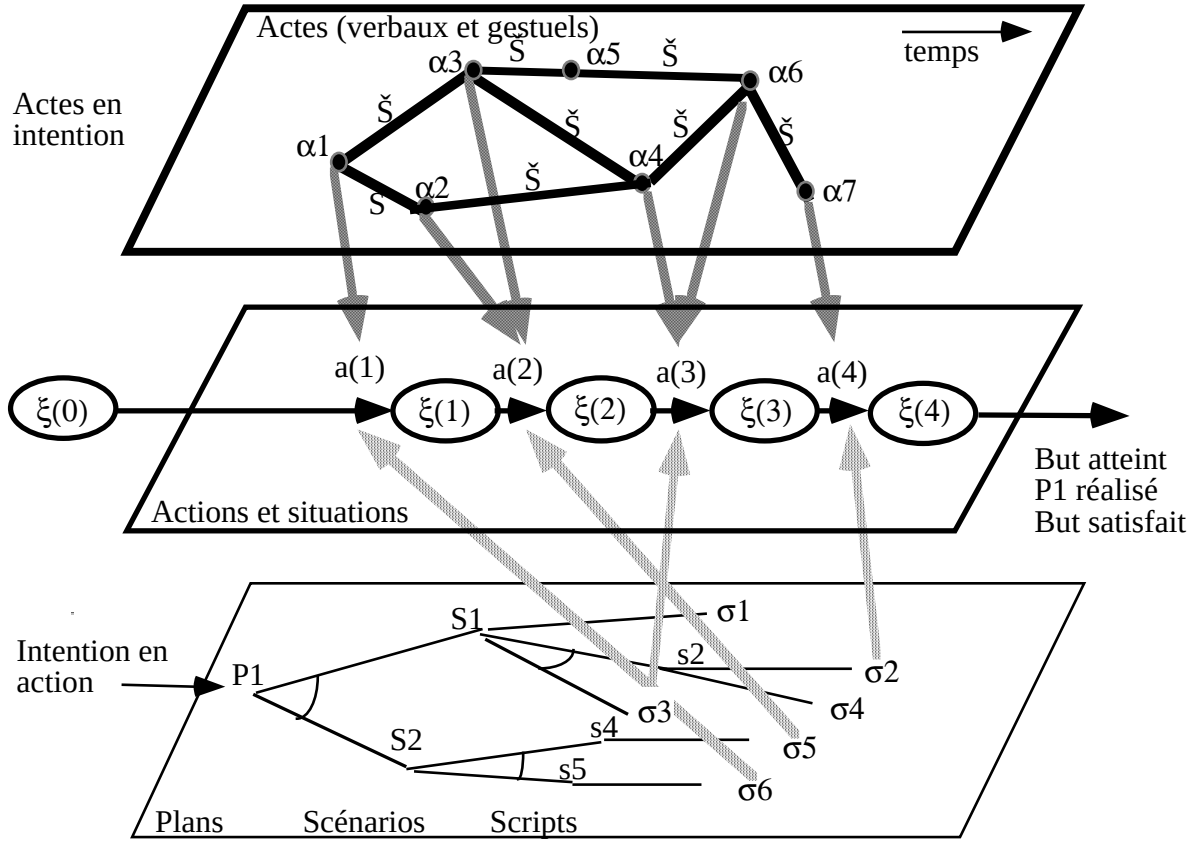


Fig. 2 : dans cet exemple le plan est réalisé par instanciation des scripts σ6, σ5, σ3, σ2, c'est-à-dire après exécution des actions {α1} • a(1), {α2 ≤ α3} • a (2), {α5} • ø , {α4 ≤ α6} • a (3), {α7}• a (4)

Ces modèles classiques peuvent être décorés d'attributs référençant des tâches, des sous tâches ou des actions élémentaires, ce qui fournit les moyens à l'exécution de reconnaître les tâches faites par l'opérateur. Dans cette approche cependant, une partie des connaissances est laissée de côté, notamment les dépendances et les contraintes entre les plans, les conditions portant sur le déclenchement ou l'activation des plans ; ces contraintes sont soit mises en œuvre au niveau des appels vers le noyau fonctionnel (tests pour vérifier la validité d'un appel), soit diluées dans les règles de syntaxe : on en perçoit une manifestation de surface au niveau de la syntaxe d'interaction autorisée mais on a perdu l'accès aux "causes" profondes. Pour lever ce type de limitation, les modèles fonctionnels [Foley, 88] présentent des modèles de tâche à deux niveaux de profondeur : la surface de l'activité et l'enchaînement causal.

Nous ne détaillerons pas davantage ce type de modèle, l'essentiel étant de comprendre ses mécanismes pour le problème de l'apprentissage qui nous motive. Nous voyons que un modèle explicite nous ne pouvons intervenir que sur le niveau pragmatique, c'est-à-dire en termes formels, sur l'application P qui associe p(↵) (l'ensemble des parties de l'ensemble des actes) à A (l'ensemble fini des actions). Ce modèle est donc insuffisant pour notre propos puisque l'ensemble A ne peut évoluer dans le temps.

Les modèles de tâche implicites

Un modèle de tâche implicite ne décrit pas la succession des actions mais seulement le but à atteindre et des moyens (ou contraintes) pour l'atteindre (le cheminement n'est pas explicité). Ce modèle convient aux tâches dites *innovantes* ou de conception dans lesquelles l'espace de recherche n'est pas complètement connu et où les connaissances du domaine ne sont pas clairement exprimables.

Durant la dernière décennie, différents mécanismes issus de l'IA ont été proposés pour simuler le comportement des concepteurs :

- l'exploration d'arbres de buts [Mahler, 85], [Mittal, 85],
- la sélection et l'expansion de plans de conception [Brown, 85],
- la proposition et la révision d'hypothèses fondées sur la satisfaction et la propagation des contraintes [Marcus, 86],
- la modification de solutions presque bonnes [Stallman, 77], [Howe, 88].

On distingue souvent trois grandes catégories de méthodes d'innovation :

- les méthodes analogiques qui s'appuient sur les ressemblances qui peuvent exister entre le nouveau problème et d'autres problèmes connus — l'objectif consiste à explorer les concepts qui sont en relation avec le concept de départ —,
- les méthodes antithétiques qui s'appuient sur les oppositions qui peuvent exister entre le nouveau problème et d'autres problèmes connus — l'objectif consiste à explorer des concepts contraires au concept de départ —,
- les méthodes aléatoires qui s'appuient sur le hasard pour explorer des concepts qui semblent sans aucun rapport avec le problème posé.

Cela conduit à représenter le modèle de tâche par :

- le but B à atteindre,
- des moyens ou des contraintes pour atteindre le but,
- et des algorithmes de *mutation* de scripts prototypes permettant d'ajuster, de modifier, de généraliser ou de spécialiser des scripts ayant déjà été utilisés dans d'anciennes situations.

Nous allons étudier plus avant une représentation adéquate pour nos objectifs à partir de ce deuxième modèle de tâche. Notons cependant que les deux types de représentation, explicite et implicite, seront utiles pour modéliser l'apprentissage de la tâche. Dans un premier temps un modèle implicite permet de prendre en compte de nouveaux plans, qui ajoutés à un modèle explicite incrémental permet de recomposer ou de contrôler le déroulement de tâches itératives.

L’apprentissage de la tâche

Comme nous l’avons annoncé ci-dessus, si nous sommes intéressés par la modélisation du dialogue dans les situations de conception pour lesquelles il n’est pas possible de prévoir toutes les activités à l’avance, puisque par définition la conception est une activité créatrice. Cependant on s’aperçoit — chez les architectes par exemple — que l’acte de conception passe souvent par une réutilisation de plans architecturaux anciens, simplement réorganisés. Dans ces conditions on a intérêt à apprendre un *plan* (pour faire une tâche ou résoudre un problème) en le construisant pendant le déroulement du dialogue à partir des buts poursuivis par l’architecte. L’apprentissage du plan permettra ultérieurement de l’assister de manière coopérative ou même d’anticiper de manière constructive. Au-delà de ce simple exemple notons également l’importance de l’apprentissage par l’action en psychologie du comportement [Piaget 64].

Les problèmes soulevés à propos de l’apprentissage de la tâche sont :

- l’acquisition d’un nouveau plan d’action, pertinent pour la tâche en question,
- sa caractérisation par rapport aux autres tâches,
- l’assignation d’un but dans le contexte du dialogue.

Puis, dans un deuxième temps, pour l’assistance à l’utilisateur, les problèmes sont :

- la reconnaissance du but poursuivi à partir des actions observées, pour inférer la logique d’action générale de l’utilisateur et identifier son plan d’action,
- l’activation pertinente des stratégies d’assistance à l’utilisateur dans le cours du dialogue.

L’acquisition du plan peut être réalisée en utilisant des sous dialogues adaptés [Caelen 95a]. Par exemple :

U : Dessine une maison

M : Qu’est-ce qu’une maison ? Montre-moi… en la dessinant…

U : "*série d’actes pour faire le dessin*" par exemple

(dessiner(carré) < dessiner(triangle)) • dessiner(maison)

avec (activer(carré) < poser(carré)) ∗dessiner(carré)

et ((sélect(triangle) ∗ dupliquer(triangle)) < déplacer(triangle)) • dessiner(triangle)

M : Il s’agit bien d’une maison n’est-ce pas ?

U : Oui.

Dans cette courte séquence dialogique menée avec une stratégie directive de la part de la machine, les actions ont toutes rapport au même but B = dessin(maison). La confirmation permet de s’assurer que le but est bien satisfait. Il y aurait de nombreuses autres stratégies possibles, ceci n’est qu’un exemple.

Au départ, on suppose que l’on dispose d’actions élémentaires bien définies (que l’on

trouve généralement dans les éditeurs graphiques). Puis, après que l'utilisateur a posé un but, il s'agit pour la machine d'*observer*, d'*ordonner* et d'*associer* cette séquence d'actions au but posé. Nous sommes donc ici dans un contexte *maître apprenant* où le maître est l'utilisateur et l'apprenant est la machine.

- pendant la phase d'observation la machine enregistre la série d'actes effectués par l'utilisateur, c'est-à-dire son activité ;
- à partir de cet enregistrement, la machine tente de distinguer des relations entre les actes, par exemple, la succession dans le temps ou la causalité ; pour établir un ordre partiel entre les actes ; et
- dans la phase d'association, on attache le plan obtenu au but poursuivi par l'utilisateur.

A la fin de cette étape, la machine est capable de recommencer la tâche à partir d'une demande de l'utilisateur. Mais l'apprentissage ne s'arrête pas ici. La nouvelle tâche apprise doit être généralisée, spécialisée ou intégrée comme une sous tâche à une tâche plus complexe. Cette *assimilation* permet la formation d'une abstraction liée au but posé par l'utilisateur. De cette façon, la connaissance acquise évolue pour se rapprocher du concept réel détenu par l'utilisateur.

Représentation du plan

Une difficulté majeure est la manière de représenter les connaissances pour supporter cet apprentissage, car on fait l'hypothèse que les connaissances acquises par la machine sont incomplètes et dynamiques et qu'elles doivent rendre explicites les différents rapports entre les actions. La modélisation d'un monde qu'introduit le concept d'action exige la prise en compte de connaissances temporelles. Diverses approches ont été proposées en Intelligence Artificielle (IA) pour tenter une modélisation des notions temporelles comme l'antériorité, le recouvrement, la simultanéité entre actions, leur causalité, etc. Différents travaux en *raisonnement temporel* ont été proposés, parmi lesquels : le calcul situationnel de [McCarthy, 69], la logique temporelle de [MacDermott 82], le calcul d'intervalles de [Allen 84] et la logique d'intervalles de [Shoham 87].

Les idées ci-après sont basées sur le travail de Shoham qui définit d'une manière générale une proposition temporelle <*i, p*> laquelle associe une assertion logique *p* à un intervalle de temps *i.* La syntaxe et la sémantique de cette logique sont définies dans [Shoham 87] et [Dean, 90].

Pour parler de changements du monde, on utilise la notion d'*événement*, c'est-à-dire, la transition d'une proposition d'un état à un autre. En effet, considérons deux propositions temporelles <(t_1, t_2), *p*> et <(t_4, t_5), $\neg p$> telles que t_2 précède t_4 ($t_2 \leq t_4$) ; la transition de *p* à $\neg p$ qui se produit dans un instant t_3 compris entre t_2 et t_4 est déterminée par un *événement*. La proposition temporelle <*i, p*> s'écrit *HOLDS(i, p)* et l'intervalle *i* est le couple $t_a \backslash t_b$, où $t_a \leq t_b$; mais pour souligner le rôle d'événement d'une expression temporelle nous l'écrirons *OCCURS(i, p)*, l'intervalle *i* où l'événement se produit est

l'instant $t_a \backslash t_b$, où $t_a = t_b$[1]. Par exemple :

OCCURS(t_1, dessiner(triangle, x_1, y_1)) ET OCCURS(t_2, déplacer(triangle, x_2)) ET

OCCURS(t_3, dessiner(carré, x_3, y_2)) ET OCCURS(t_4, déplacer(carré, x_4))

décrit la séquence d'actions que l'utilisateur a effectué pour dessiner une maison et dans laquelle *dessiner(triangle, x, y)* représente l'action de dessiner un triangle à la position *x* (où *x* est un vecteur) et de taille *y*. Chaque événement correspond à une action de base. Chaque action de base est définie par une règle *à la STRIPS* qui permet de récupérer l'ensemble des propositions temporelles affectées par l'exécution de l'action. Par exemple, la règle simplifiée suivante établit les conditions et les effets pour l'action déplacer :

HOLDS($t_0 \backslash t_1$, position(carré, x_1)) ET

OCCURS(t_1, déplacer(carré, x_2)) ET HOLDS($t_1 \backslash t_2$, position(carré, x_2))

A partir de ces règles, on introduit une relation de précédence temporelle entre les propositions. Par exemple, après la séquence d'actions pour dessiner la maison, on récupère les propositions suivantes :

HOLDS($t_0 \backslash t_1$, $\neg$dessiné(triangle)) ET

HOLDS($t_1 \backslash t_a$, dessiné(triangle)) ET

HOLDS($t_1 \backslash t_2$, position(triangle, x_1)) ET

HOLDS($t_2 \backslash t_b$, position(triangle, x_2)) ET

HOLDS($t_0 \backslash t_3$, $\neg$dessiné(carré)) ET

HOLDS($t_3 \backslash t_c$, dessiné(carré)) ET

HOLDS($t_3 \backslash t_4$, position(carré, x_3)) ET

HOLDS($t_4 \backslash t_d$, position(carré, x_4))

La figure 3, une adaptation d'un *time map* [Dean 87], montre les propositions et leur précédence temporelle. Les lignes noires représentent les intervalles de temps et les lignes pointillées la précédence temporelle. Cette relation entre les propositions est déduite d'une part, par les règles associées à chaque action et d'autre part, par les durées de réalisation de la tâche. Le premier cas est une relation forte qui oblige une précédence temporelle *nécessaire*. On n'est pas capable de changer l'ordre d'exécution (avant de déplacer l'objet, il doit exister). Le deuxième cas entraîne une relation temporelle *fortuite* i. e. $t_2 \leq t_3$ (le dessin du triangle avant le carré ne change rien au résultat final). Ces deux types de relation permettent de mieux appréhender la combinaison temporelle des actions, et donc de mieux faire ressortir les parallélismes possibles et les séquencements obligatoires. De cette façon, on peut donc obtenir un plan, non pas comme un ensemble de séquences d'actions, mais comme un ensemble

---

[1] [Shoham 87] présente une catégorisation des propositions temporelles. Par rapport à celle-ci, une proposition ***HOLDS(i, p)*** est équivalente à une expression du type *liquide* et ***OCCURS(i, p)*** à une expression du type *gestalt*.

d’actions partiellement contraintes dans le temps les unes par rapport aux autres et par rapport à certains faits.

La distinction entre ces deux types de relation nous permet de relier une série d’actions qui sont dépendantes entre elles. Une action dépend d’une autre si ses effets sont des conditions par l’exécution de l’autre, c’est-à-dire, s’il existe une relation temporelle nécessaire entre elles. On considère cette série d’actions comme une seule action. Cette action *composée* est liée aux autres actions, aussi probablement composées, à travers une relation temporelle fortuite. Après avoir distingué les actions composées, on peut appliquer des opérations de réduction dans chacune pour éliminer les propositions redondantes et simplifier le plan.

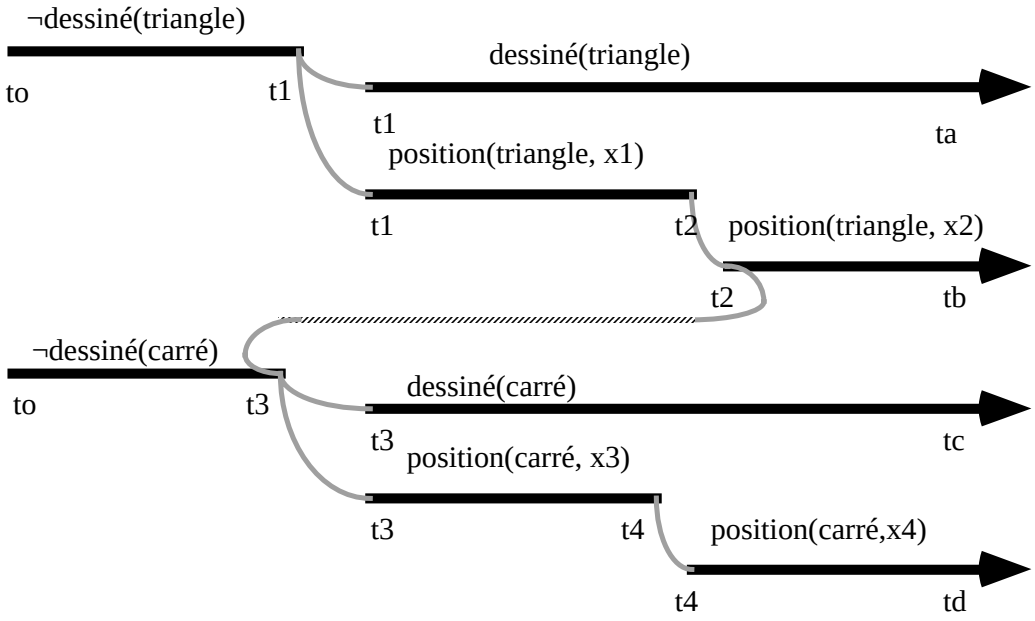


Fig. 3. Les propositions et leur précédence temporelle.

On peut réduire l’action composée pour dessiner le carré au schéma suivant (fig. 4) :

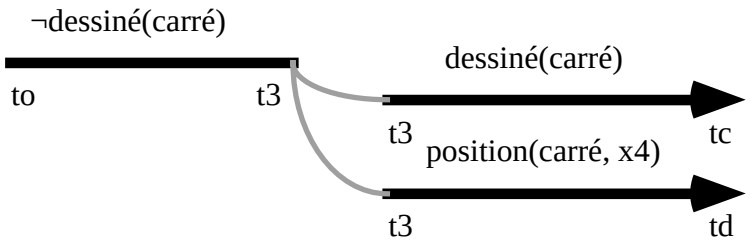


Fig. 4 : Action composée après réduction.

cette réduction entraîne l’élimination de l’action *déplacer* et la modification de la création

du carré directement à la position x_4.

Autre opération importante, que l'on peut tenter après la séparation des actions composées est la recherche de relations spatiales entre les objets sur lesquels elles sont centrées. L'établissement des relations spatiales peut être réalisé à travers les déductions réalisées par la machine et/ou à travers l'aide directe de l'utilisateur [Li 94].

L'objet comme résultat de l'interaction

À travers ce formalisme pour la représentation des connaissances, on est donc capable de décrire la séquence d'actions pour faire la tâche mais aussi de produire des objets comme résultats de ces actions. Le but de l'utilisateur est vu comme l'ensemble d'effets produits par la réalisation des actions. But et objet sont donc deux notions complètement liées. Pour décrire les objets, nous allons focaliser notre attention sur la séquence d'actions observées et sur le résultat des actions, dans le contexte du but à atteindre. Par exemple, les objets « maison » et « lampe » seront différents plus parce qu'ils ont été dessinés avec des actions différentes que parce que leurs formes sont différentes. Peut-être que « maison » a été dessinée en positionnant le triangle sur le rectangle alors que « lampe » a été dessinée en posant le triangle sur un cercle.

Le processus d'assimilation ainsi que la reconnaissance des intentions prennent aussi l'action comme élément de base. L'utilisateur peut montrer deux séquences différentes pour faire la même tâche, mais le processus d'assimilation nous permet, à partir des actions réalisées, de former une abstraction plus générale. Ainsi, la connaissance de la machine évolue pour se rapprocher du concept exprimé par l'utilisateur. De la même façon, le processus de reconnaissance des intentions prend l'action comme élément de base. C'est à partir des actions observées que la machine tente d'inférer le but de l'utilisateur. Néanmoins, comme nous l'avons vu, il faut exprimer une séquence d'actions par ses effets pendant la phase d'ordonnancement. Cette phase trouve, à partir des effets, des dépendances entre les actions et permet d'éliminer les actions redondantes (cette phase complète la phase de réduction).

L'assimilation de la tâche

Après avoir appris une tâche, une autre considération importante est l'assimilation de celle-ci par rapport aux tâches déjà apprises. Le résultat de cette assimilation est une abstraction du plan pour atteindre le but poursuivi par l'utilisateur. Depuis plusieurs sessions de travail, la connaissance de la machine évolue pour se rapprocher du *concept* exemplifié par l'utilisateur. L'apprentissage dans cette étape, est le perfectionnement des connaissances, l'amélioration des performances au cours de l'expérience. On part de l'hypothèse que les connaissances incorporées par le système sont incomplètes et qu'elles évoluent à chaque fois que l'utilisateur montre un nouvel exemple d'un même concept.

Il y a différents paradigmes pour l'apprentissage automatique [Carbonell, 89]. Les quatre paradigmes d'apprentissage principaux sont : l'inductif, l'analytique, le génétique et le connexionniste. Bien que ces paradigmes aient le même objectif, ils apprennent de façon très différente. Toutefois, tous les quatre voient le processus d'apprentissage comme une modification de la connaissance courante de l'apprenant par interaction avec une

source d'information extérieure. Dans notre cas, où l'utilisateur tente de montrer son savoir-faire à la machine, et où la machine et l'utilisateur doivent partager des ressources et faire des raisonnements collectivement, les connaissances doivent être codifiées de manière compréhensible par tous les deux [Carbonell, 83]. Les paradigmes "génétique" et "connexionniste" ne représentent pas leurs connaissances de façon simple à interpréter par l'utilisateur. Il semble donc qu'il soit plus difficile de les utiliser ici.

Pour la formation d'un concept, notre travail s'appuie sur le paradigme inductif en utilisant des stratégies d'apprentissage de " concepts par l'exemple ". L'idée principale est l'adaptation de la description d'une tâche à chaque fois que l'utilisateur donne une instance de cette tâche. Dès que l'on apprend une nouvelle tache, le concept créé est la première approximation de la description de celle-là. Lorsque l'utilisateur présente une nouvelle instance de la tâche, un processus de *généralisation* adapte l'approximation actuelle du concept pour inclure les caractéristiques propres de l'instance montrée. Le résultat de la généralisation, entre l'approximation courante et l'instance actuelle, est une nouvelle approximation plus complète. Cependant, dès la première approximation, bien qu'inexacte et/ou incomplète, on peut l'utiliser pour reproduire la tâche.

Dans notre cas, l'apprentissage est caractérisé par :

a) La formation d'un concept, il est (i) empirique — au départ, on ne connaît rien sur la tâche —, et il (ii) procède de façon incrémentale — la description d'une tâche est ajustée à chaque occurrence d'une nouvelle instance de cette tâche.

b) Les tâches sont classifiées avec l'aide et sous le contrôle de l'utilisateur. L'utilisateur guide la formation de nouveaux concepts, ainsi que l'affinage de concepts déjà appris.

c) La description d'une tâche doit permettre sa reconnaissance ; dans une situation ultérieure, on tente d'inférer la tâche à partir des actions observées.

Il y a différentes méthodes pour réaliser des généralisations. Des opérations comme *la restriction de conjonction* ou *le remplacement de constantes par des variables* sont utilisées pour dériver des généralisations [Michalski 83].

Les processus de généralisation ont été amplement explorés, Plotkin a été le premier à analyser rigoureusement cette notion. La *généralisation la plus spécifique* (basée sur la θ*-subsumption*) est la première solution proposée par Plotkin [Buntime, 96]. L'idée est de trouver entre deux termes la généralisation minimale possible. C'est-à-dire, une expression qui conserve le maximum de caractéristiques communes aux expressions de départ.

La généralisation la plus spécifique de deux termes T_1 et T_2, dénoté $g(T_1, T_2)$, est définie de la manière suivante :

$g(T_1, T_2)$ est une généralisation de T_1 et T_2 s'il existe deux substitutions σ_1 et σ_2 telles que :

$g(T_1, T_2).\ \sigma_1 = T_1$ et $g(T_1, T_2).\ \sigma_2 = T_2$

$g(T_1, T_2)$ est plus spécifique que toute autre généralisation selon l'ordre partiel sur les termes :

$T \varepsilon T'$, T est plus spécifique que T', si et ssi il existe une substitution σ telle que $T'.\sigma = T$

Pour faire le calcul de la généralisation la plus spécifique de deux termes T_1 et T_2, on a :

- si $T_1 = T_2$, alors $g(T_1, T_2) = T_1$
- si $T_1 = f(s_1, s_2 \ldots s_n)$ et $T_2 = f(e_1, e_2 \ldots e_n)$,

  alors
  $g(T_1, T_2) = f(g(s_1, e_1), g(s_2, e_2) \ldots g(s_n, e_n))$
- si T_1 et T_2 sont différents (c'est-à-dire des fonctions ou des variables différentes, ou bien l'un est une fonction et l'autre une variable) alors $g(T_1, T_2)$ est une variable v, où la même variable est partout utilisée comme la généralisation de ces deux termes.

Par exemple :

$T1 = \text{dessiner}(\text{obj1}, x_1, y_1)$

$T2 = \text{dessiner}(\text{obj2}, x_2, y_1)$

$g(T1, T2) = \text{dessiner}(X, Y, y_1)$

Dans le cas le plus simple, la généralisation la plus spécifique est exactement l'opération duale de l'unification. Le résultat du processus de généralisation est une hiérarchie qui exprime la relation de généralisation/spécialisation. Pour l'exemple ci-dessus, on a la hiérarchie suivante :

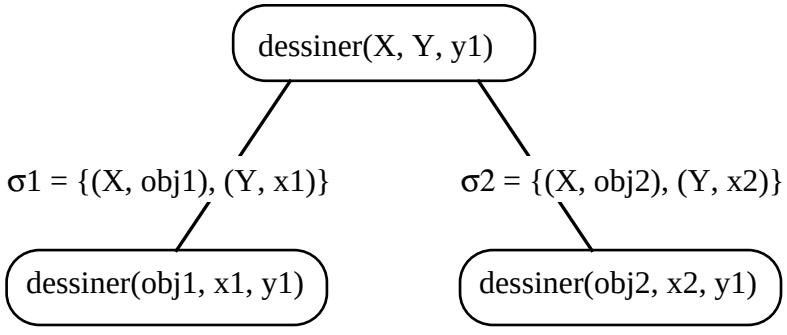


Fig. 5 : Relation de généralisation/spécialisation.

Les substitutions σ_1, σ_2 nous permettent de revenir aux instances du départ. Cette spécialisation du concept général est nécessaire pour reproduire la tâche.

A partir de ces idées, on obtient une approximation du concept comme un ensemble d'actions qui décrivent la tâche. Cependant, il reste à caractériser le concept, c'est-à-dire, à trouver des associations parmi les éléments descriptifs du concept. La caractérisation d'un concept entraîne donc la recherche des relations inhérentes entre ses éléments, à partir des valeurs observées dans les exemples [Li, 94 ; Bratko, 91]. Dans notre cas, pour capturer ce type de relation, on utilise une proposition, dénommée *corrélation*, qui regroupe les associations trouvées.

Par exemple, l'utilisateur dessine une maison, ce premier dessin devient la première approximation du concept de « maison » :

→ maison1 (avec fenêtre) 

OCCURS(t_1, dessiner(triangle1, x_1, y_1)) ET

OCCURS(t_2, dessiner(rectangle1, x_2, y_2)) ET

OCCURS(t_3, déplacer(rectangle1, x_3)) ET

OCCURS(t_4, dessiner(rectangle2, x_4, y_3)) ET

OCCURS(t_5, changer_taille(rectangle2, y_4)) ET

OCCURS(t_6, déplacer(rectangle2, x_5))

après la phase de réduction on obtient :

OCCURS(t_1, dessiner(triangle1, x_1, y_1)) ET

OCCURS(t_2, dessiner(rectangle1, x_3, y_2)) ET

OCCURS(t_4, dessiner(rectangle2, x_5, y_4)) ET

HOLDS($t_4 \backslash t_b$, corrélation($\{x_1, x_3, x_5\}$, $\{y_1, y_2, y_4\}$)))

Supposons que l'utilisateur montre un deuxième dessin de « maison » à travers la séquence suivante :

→ maison2 (sans fenêtre) 

OCCURS(t_1, dessiner(triangle1, x_4, y_1)) ET

OCCURS(t_2, dessiner(rectangle1, x_3, y_2)) ET

HOLDS($t_2 \backslash t_b$, corrélation($\{x_4, x_3\}$, $\{y_1, y_2\}$))

Le résultat de l'assimilation de cette deuxième instance par rapport à la première approximation sera le suivant :

concept *maison* = *g*(maison1,maison2)

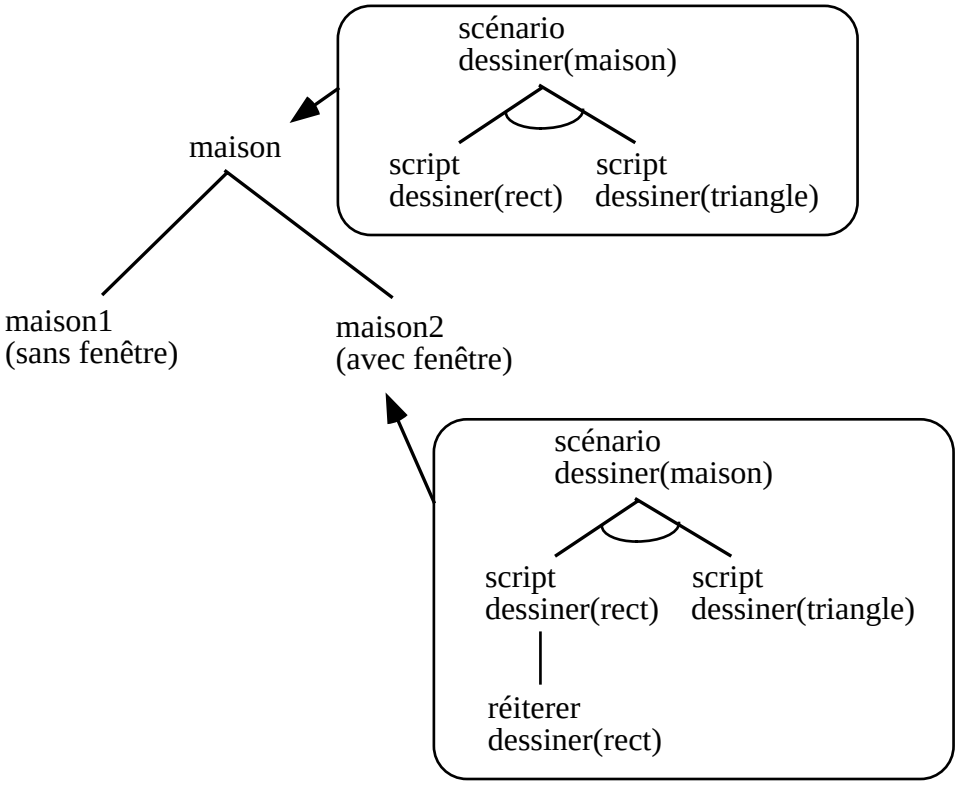
OCCURS(t_1, dessiner(triangle1, Xt, Yt)) ET

OCCURS(t_2, dessiner(rectangle1, Xr, Yr)) ET

HOLDS($t_2 \backslash t_b$, corrélation({Xt, Xr}, {Yt, Yr}))

La dernière proposition *HOLDS($t_2 \backslash t_b$, corrélation({Xt, Xr}, {Yt, Yr}))*, tente de capturer les caractéristiques inhérentes au concept *maison* qui sont satisfaites par les deux instances, par exemple, le fait que le triangle est au-dessus du carré et solidaire par rapport à lui dans les déplacements.

Nous obtenons finalement une généralisation du concept de « maison » associé aux actions permettant de l'instancier :



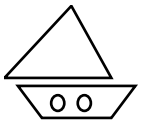
L'idée de la généralisation plus spécifique peut être étendue pour prendre en compte des connaissances d'arrière-plan et tenter d'obtenir de meilleures conclusions, par exemple, par *subsumption généralisée* [Buntime, 88]. Une autre possibilité est l'utilisation des idées proposées par la programmation logique inductive [Berdagano, 96].

Reconnaissance du but

La reconnaissance du but peut être vue comme le problème inverse de l'apprentissage : les notions résultantes de l'apprentissage dirigent, pendant le déroulement du dialogue, la reconnaissance du but poursuivi par l'utilisateur, c'est-à-dire, qu'à partir des concepts appris, on tente de retrouver le but de l'utilisateur. Ainsi, la machine sera capable de coopérer lorsqu'une *situation*, analogue à une autre déjà vécue, est rencontrée. Dans ce cas-là, après la confirmation de l'utilisateur et éventuellement un sous dialogue de particularisation de la tâche, le plan pourra être exécuté. De cette manière, en s'appuyant sur une stratégie *dirigée par les intentions*, la machine établit un dialogue qui converge plus rapidement vers la réalisation de la tâche.

La recherche du but utilise un processus de raisonnement pour déduire le but de l'utilisateur à travers la succession d'actions observées. Comme nous l'avons évoqué, l'action est l'élément principal, c'est elle qui permet de caractériser un concept ou une situation (le dessin d'une maison et d'une lampe sont différents par les actions réalisées et non pour ce que cela représente pour l'utilisateur). La machine n'a évidemment pas de connaissances d'arrière-plan ici. Pour reconnaître le but de l'utilisateur on doit enregistrer et analyser les événements observés, identifier les actions et ensuite vérifier si elles font partie d'un concept connu.

Le système de reconnaissance tente donc d'établir un lien entre une séquence répertoriée et la séquence d'actions observées. Cette opération est réalisée de manière incrémentale. Au fur et à mesure de l'occurrence des événements une description *évolutive* est construite. Lorsque cette description peut s'apparier avec un sous-ensemble d'actions d'un concept connu, la machine a « reconnu l'intention » de l'utilisateur. Cela peut ensuite être mis à profit dans le dialogue pour demander une confirmation du concept ou donc continuer la tâche après la validation de l'utilisateur. Dans le cas d'ambiguïté, c'est-à-dire, quand le processus de reconnaissance obtient plus d'une hypothèse, on attend jusqu'à ce que la description courante soit suffisamment complète pour tomber sur une seule possibilité. Par exemple, supposons deux concepts connus, *bateau* et *voiture* qui présentent tous deux, deux cercles (les hublots pour le bateau et les roues pour la voiture) :



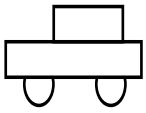
dessiner(triangle1, x1, y1)

dessiner(triangle2, x2, y2)

dessiner(cercle1, x3, y3)

dessiner(polygone1, x4, y4)

dessiner(cercle2, x5, y5)



dessiner(rectangle1, x1, y1)

dessiner(cercle1, x2, y2)

dessiner(cercle2, x3, y3)

dessiner(rectangle2, x4, y4)

Si l'utilisateur commence à dessiner les deux cercles, la description courante ne contient que ces deux cercles : il y a ambiguïté, le processus de reconnaissance obtient deux hypothèses. On attend un nouvel événement pour résoudre l'ambiguïté, par exemple un rectangle. Au moment où l'utilisateur dessine un rectangle la machine « comprend l'intention » de dessiner une voiture ; elle est dans les conditions de déclencher un sous dialogue pour continuer la tâche : elle peut par exemple anticiper sur le dessin d'une voiture si on est dans une stratégie d'assistance coopérative ou demander des informations à propos du concept.

Le processus de formation de la description courante est similaire à la phase

d'ordonnancement pendant l'apprentissage d'une tâche, mais il est plus complexe puisqu'il doit prendre en compte sa nature incrémentale. A chaque fois que la machine observe un événement, celui-ci est intégré à la description courante. On fait les réductions appropriées pour obtenir une description plus simple, dépouillée des actions intermédiaires redondantes. Cependant, l'intégration d'une nouvelle action observée peut entraîner l'élimination des conditions qui invalident des déductions préalables. Dans notre exemple, après que l'utilisateur a dessiné deux cercles, le processus de reconnaissance a établi deux hypothèses. Puis si l'utilisateur dessine un troisième cercle, la déduction doit être annulée puisqu'elle n'est plus valide. Au même temps, les nouvelles conditions créées renvoient à un autre ensemble d'hypothèses.

Enfin, une fois qu'il existe une hypothèse valide, le processus de coopération doit d'une part, établir les actions déjà réalisées et faire la planification du reste des actions pour finir la tâche ; et d'autre part, demander les paramètres inconnus, pour particulariser la tâche en cours, à partir d'un sous dialogue de spécialisation.

Conclusion

Ce chapitre a présenté les deux grands types de modèles de tâche, explicites et implicites. Les modèles de tâche implicites dérivent d'une planification hiérarchique tandis que les modèles de tâche implicites, autorisant la dynamicité, sont plus adaptés à une planification de nature opportuniste (*situated action*).

Ce chapitre a également abordé quelques questions relatives à l'apprentissage dans un modèle de dialogue pour affronter des situations de conception innovantes. L'idée générale est de favoriser l'interaction entre l'homme et la machine à travers le dialogue en langage naturel mais aussi de profiter du dialogue pour adapter la machine à l'homme. L'apprentissage est essentiel pour améliorer la communication ; si la machine apprend à faire une tâche, à acquérir de nouveaux concepts et à agir de manière opportune alors l'utilisateur pourra lui confier des travaux de production assistée. La machine sera capable de coopérer avec lui. En retour, l'apprentissage permet de guider le dialogue en utilisant une stratégie basée sur des buts et qui est la base de toute stratégie coopérative

L'apprentissage proposé est basé sur un formalisme qui incorpore des aspects fondamentaux pour représenter des actions. Il permet d'exprimer des notions de précédence temporelle pour fixer un ordre et, de cette façon, faire ressortir les parallélismes possibles et les séquencements obligatoires entre les actions d'un plan. L'apprentissage est réalisé en utilisant un processus de généralisation. La machine acquiert les connaissances pour faire une tâche, et le processus de généralisation fait évoluer ses connaissances à chaque fois que l'utilisateur montre une nouvelle instance de la tâche. Le résultat est la constitution, pas à pas, d'une connaissance partagée. La stratégie de généralisation exposée, la *généralisation plus spécifique,* a des limites importantes, bien qu'elle puisse être étendue pour prendre en compte des connaissances d'arrière-plan : il est toujours possible de tomber sur des généralisations incorrectes. Une autre solution possible est l'utilisation de la *programmation logique inductive* qui propose des opérateurs inductifs à l'inverser les règles de déduction comme l'unification, la résolution et l'implication [Berdagano, 96].

Il est important de relever qu'un inconvénient de cette approche d'apprentissage est sa dépendance vis-à-vis de l'utilisateur pour la formation de concepts. L'ordre de présentation des exemples, le nombre d'instances et la classification d'une tâche sont des aspects déterminants qui influent fortement sur les concepts appris et par conséquent, dans la reconnaissance des buts de l'utilisateur.

Finalement, il est aussi important de rappeler que la reconnaissance des buts est un « passage obligé » pour doter la machine de capacités qui lui permet de cordonner les processus actionnels pendant le dialogue homme machine. A cet égard l'apprentissage apporte aussi des solutions à ce problème.