Coordination d'agents rationnels par planification distribuée

S. Boussetta[*], A. El Fallah-Seghrouchni[#], S. Haddad[*], P. Moraïtis[*], M. Taghelit[*+]

[*] *LAMSADE (URA 825 CNRS) - Université Paris IX - Dauphine*
Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16
{boussetta, haddad, moraitis, taghelit}@lamsade.dauphine.fr

[#] *LIPN- (URA 1507 CNRS) - Institut Gallilée, Université Paris XIII*
Avenue Jean-Baptiste Clément, 93400 Villetaneuse
Adresse électronique : elfallah@ura1507.univ-paris13.fr

[+] *Université de Tours, Faculté des Sciences et Techniques,*
Parc Grandmont, 37200 Tours

RESUME. Ce papier propose un modèle récursif de représentation et de manipulation de plans, fondé sur les réseaux de Petri récursifs (RdPR), permettant la spécification des activités concurrentes, le raisonnement sur des actions simultanées et des processus continus. De plus, ce formalisme supporte une théorie de vérification et des mécanismes de transformation (i.e. abstraction, raffinement, fusion). Ses principales caractéristiques sont l'indépendance par rapport au domaine, la couverture de plusieurs situations d'interaction (positives et négatives) et une sémantique formelle pour la coordination. Notre approche fondée sur la sémantique des RdPR gère l'entrelacement de la planification et de l'exécution. Dans ce papier, nous montrons comment cette approche peut être utilisée pour coordonner les plans d'agents dans un environnement dynamique.

ABSTRACT. This paper proposes a recursive model for the representation and the handling of plans based on Recursive Petri Nets (RPN) which support the specification of concurrent activities, reasoning about simultaneous actions and continuous processes, a theory of verification and mechanisms of transformation (e.g. abstraction, refinement, merging). The main features of the RPN formalism are domain independence, broad coverage of interacting situations and operational coordination. We also suggest an approach to the interleaving of planning and execution, based on the RPN semantics. In this paper, we show how this approach can be used to coordinate agents' plans in a dynamic environment.

MOTS-CLES : Système Multi-Agent, Planification Distribuée, Coordination, Interaction, Réseaux de Petri Récursifs, Raffinement, Fusion de Plans.
KEY WORDS : Multi-Agent System, Distributed Planning, Coordination, Interaction, Recursive Petri Nets, Refinement, Plan Merging

1. Introduction

Les Systèmes Multi-Agents (S.M.A.) proposent une approche originale de conception de systèmes intelligents et coopératifs. Ils se caractérisent par la présence d'agents rationnels et autonomes évoluant dans un environnement partagé et dynamique. La conception de tels systèmes introduit de nouveaux problèmes inhérents à la vision partielle des agents, la poursuite d'objectifs locaux et l'entrelacement des activités des agents. La coopération entre agents, en vue de engendrer un comportement global cohérent du S.M.A., requiert des mécanismes élaborés de coordination afin d'éviter d'éventuels conflits et de favoriser la synergie des activités des agents [BON 88].

Ferber [FER 95] définit une large taxonomie des différentes formes de coopération dont la collaboration coordonnée qu'il qualifie comme étant " la situation de coopération la plus complexe" du fait qu'elle doit gérer à la fois des buts incompatibles, des ressources insuffisantes et des compétences insuffisantes. Il existe différentes techniques de coordination d'actions dans les S.M.A. On peut en distinguer quatre [FER 95].

1. *La coordination par synchronisation d'actions* qui s'apparente aux techniques utilisées dans les systèmes distribués. Il s'agit de spécifier l'enchaînement des actions et de gérer leur parallélisme potentiel. Cette technique n'intègre pas de mécanismes cognitifs et peut être vue comme un support de base à toute autre forme de coordination.

2. *La coordination réactive* repose sur le mécanisme de perception et suppose que le contexte est très évolutif ce qui rend l'anticipation sur la suite des actions difficilement réalisable. Cette technique s'applique aux systèmes réactifs.

3. *La coordination par réglementation* consiste à introduire (imposer) des règles de comportement afin d'éviter les conflits entre agents. Cette technique permet de réduire les coûts de communication entre agents [BRI 95]. Cependant, les lois pèsent sur l'autonomie des agents dont il faut prédire précisément le comportement sans pour autant se focaliser sur les mécanismes de coopération.

4. *La coordination par planification multi-agent* (centralisée ou distribuée) où les agents élaborent un plan multi-agent qui spécifie leurs actions futures et leurs interactions. La synthèse des plans exige un raisonnement sur la manière dont les actions des différents agents peuvent interférer et par conséquent des représentations d'actions parallèles. L'exécution non contrôlée de plans concurrents amène à des situations de blocage dues par exemple à l'utilisation incohérente de ressources. La synchronisation des plans peut être effectuée à différents points de la résolution d'un problème : durant la phase de décomposition du problème [COR 79], lors de la construction des plans en bâtissant des plans hiérarchisés qui interagissent faiblement [COR 79], après la construction des plans [GEO 83], en alignant des plans partiels d'une façon incrémentale [DUR 89][DEC 92], ou en introduisant dans le processus de planification un raisonnement sur les interactions et les dépendances possibles [ROS 82][EPH 95].

Les systèmes proposés en planification multi-agent centralisée [GEO 83] [GEO 88] présentent des options intéressantes pour la résolution des conflits et la convergence vers

une solution globale mais tous ont les mêmes désavantages liés à la centralisation du contrôle au niveau d'un seul agent, à savoir :

- si le coordinateur est défaillant, le système est incontrôlable ;

- le coordinateur central constitue un goulet d'étranglement au niveau de la communication et du traitement, i.e. les différents agents travaillant d'une façon concurrentielle dans le réseau doivent envoyer leurs actions au coordinateur afin qu'il résolve les conflits éventuels et gère leurs interactions ;

- la charge de communication est très importante puisque des quantités importantes de données non nécessairement utiles sont envoyées vers le coordinateur en pénalisant le débit utile du réseau.

La planification distribuée s'intéresse aux systèmes où plusieurs agents peuvent planifier et exécuter des plans de manière concurrente et autonome [GEO 90] [MAR 92] [EPH 95]. En outre, l'environnement des agents est évolutif et des événements peuvent survenir durant les phases de synthèse et/ou d'exécution de plans.

La planification distribuée requiert le développement de modèles d'actions et de plans multi-agents [GEO 90] permettant la gestion de la concurrence potentielle entre les activités des agents. Ceci conduit à l'élaboration de formalismes offrant une grande puissance d'expression du parallélisme, de concurrence d'actions, d'événements non atomiques, etc., ainsi que l'expression de nouveaux buts tels que les buts de maintenance, d'ordonnancement d'actions, etc.

Différentes approches ont été proposées en planification distribuée et plus particulièrement la planification incrémentale qui consiste à considérer un ensemble de plans déjà coordonnés dans lequel on insère un nouveau plan à coordonner avec les plans existants.

L'approche développée par von Martial [MAR 92] repose sur la définition d'une taxonomie des relations entre les plans de différents agents et le développement d'une structure de communication entre des agents autonomes qui échangent leurs plans et qui négocient pour résoudre les relations entre ces plans. Le modèle formel proposé offre plusieurs avantages tels que la résolution des interactions positives et négatives. Cependant il ne coordonne que deux agents à la fois ce qui ne permet pas de traiter les interactions entre n agents.

L'approche développée par Alami [ALA 94] est fondée sur le paradigme de fusion de plan. Le modèle proposé est robuste et coordonne n agents à la fois, mais la coordination est contrôlée par le nouvel agent qui insère son plan. De plus, le processus de coordination suspend les exécutions des plans en cours et le nouveau plan est généré à partir des plans existants ce qui est incompatible avec la notion d'agents autonomes qui produisent leur plan indépendamment de ceux des autres et les coordonnent ensuite.

L'approche que nous avons présentée dans [ELF 96a] propose un algorithme distribué qui coordonne n agents simultanément. Le modèle formel est basé sur la notion d'ordre partiel entre les actions. La coordination de plans consiste à trouver un ordre total qui étend l'ordre partiel en introduisant des arcs de synchronisation entre les actions. L'ordre total obtenu assure la faisabilité du plan global. D'une part, l'algorithme proposé résout les

interactions positives en utilisant les pré- et post-conditions des actions. D'autre part, il résout les interactions négatives par synchronisation d'actions. Cependant, le formalisme utilisé implique une représentation du monde très abstraite et le traitement des interactions positives ne couvre qu'un nombre restreint de situations.

Dans cet article, nous proposons un modèle formel de représentation et de manipulation de plans en vue de la coordination d'agents. Cette approche unifie l'algorithme de coordination à *n* agents [ELF 96a] et le formalisme des Réseaux de Petri Récursifs (RdPR) [ELF 95][ELF 96b] introduits pour pallier à certaines limitations des catégories usuelles des réseaux de Petri [JEN 91]. L'intérêt du modèle provient non seulement d'une représentation formelle des plans mais aussi de celle des conditions de validité de leur exécution.

Nous présentons maintenant les grandes lignes de notre démarche : les hypothèses sur les S.M.A. considérés, les objectifs d'un mécanisme de planification distribuée pour ces S.MA., les contraintes qui garantissent la viabilité d'un tel mécanisme dans un contexte multi-agents, et les principaux aspects de notre solution.

Hypothèses

- Les agents définissent leur comportement à l'aide d'un plan. Afin de prendre en compte les évolutions de l'environnement, chaque agent entrelace planification et exécution. Cet entrelacement peut se traduire par une planification partielle ou incrémentale. Cette planification incrémentale doit prendre en compte les interactions avec d'autres agents. Deux types d'interaction peuvent être distingués : les interactions positives correspondant à des délégations d'actions de la part d'autres agents et les interactions négatives portant sur des conflits de ressources.
- Notre solution n'impose pas un algorithme spécifique de synthèse de plan. Cependant, nous supposerons que l'agent est toujours capable de compléter son plan en cours d'exécution par au moins un sous-plan purement local (i.e. ne nécessitant pas d'interaction ultérieure) dépendant de son contexte. Ce sous-plan doit être interprété comme garantissant un comportement minimal vis à vis des buts de l'agent.

Objectifs

Notre principal objectif est d'assurer l'exécution globale du S.M.A. Cette exécution globale signifie que le plan de chaque agent sera toujours exécutable et ceci :

- quelque soit l'ordre d'exécution local de chaque agent,
- et quelque soit l'entrelacement des exécutions des agents du S.M.A.

Nous ne présenterons pas dans cet article de preuve formelle de ce résultat qui sera publié ultérieurement. Cette preuve est une extension d'un résultat similaire publié dans [ELF 96a].

Nous visons également à fournir au concepteur de S.M.A. un modèle formel de traitement unifié des interactions entre agents du point de vue de la planification. Les aspects organisationnels et décisionnels ne sont pas couverts par cette approche.

Contraintes

- On ne cherchera jamais à former un plan global (ni durablement ni même temporairement) pour éviter un coût de communication trop élevé, une lourde charge pour l'agent qui constitue le plan et une éventuelle suspension de l'exécution lors du calcul de ce plan.
- Le traitement des interactions entre agents doit être sélectif (i.e. n'implique que les agents concernés par cette interaction), il ne suspend pas l'exécution des actions engagées et une fois celui-ci terminé, les agents retrouvent leur autonomie d'exécution moyennant les synchronisations produites.

Principes de notre solution

Le modèle de plan que nous adoptons est celui des Réseaux de Petri Récursifs (RdPR) défini dans [ELF 96b]. Ses principaux avantages en termes de planification sont les suivants :

- il permet d'exprimer un ordre partiel entre les actions ainsi que certaines relations temporelles telles que la disjonction des intervalles d'exécution des actions,
- les actions abstraites sous-jacentes à la récursivité fournissent un support adéquat pour la planification hiérarchique, partielle et conditionnelle,
- les techniques de construction du graphe d'états réduit [GOD 96] basées sur les ordres partiels s'appliquent de manière particulièrement efficace au calcul de la consistance d'un plan modélisé par un RdPR (cf. Section 3),
- on peut définir formellement plusieurs fusions de RdPR chacune traduisant une nature d'interaction (cf. Section 3),
- le modèle d'exécution de plans permet de décrire l'environnement par des objets et des méthodes. Il est donc intégrable dans un environnement de programmation objet.

Le protocole de traitement des interactions que nous proposons distingue les interactions positives et négatives.

1. Le traitement d'une interaction positive satisfait les critères suivants :

- la détection d'une interaction se fait par nécessité (i.e. incapacité d'exécuter certaines actions d'un plan local),
- la stratégie et le mécanisme du choix d'un agent "fournisseur" sont indépendants de notre traitement et peuvent s'appuyer sur des techniques classiques telles que le réseau contractuel [SMI 80] ou les graphes de dépendances [SIC 93],

- à la différence d'une simple délégation de tâches, l'agent choisi n'accepte de réaliser les actions demandées que si elles s'insèrent de manière consistante dans son plan d'exécution (toujours partiel et incrémentable),
- cette consistance s'obtient par fusion de ce plan avec une abstraction du plan envoyée par l'agent demandeur,
- de plus, l'exécution des actions demandées est asynchrone i.e. les deux agents en interaction continuent leur exécution de manière autonome mais synchronisée.

2. Les interactions négatives traitées ici concernent des plans présentant des accès conflictuels aux ressources d'un même agent. Une extension de ce type de conflit aux ressources partagées du S.M.A. a été proposée dans [ELF 97].
- une interaction négative est automatiquement détectée lors d'une incrémentation de plan (grâce au concept de transition exclusive),
- ici aussi une fusion des sous-plans résout le problème de la consistance ; cette fusion recherche de plus à conserver le maximum de parallélisme dans l'exécution des deux sous-plans.

3. Ce protocole est capable de traiter simultanément et systématiquement les interactions positives et négatives contrairement à la plupart des approches proposées reposant sur des heuristiques multiples [DUR 91][MAR 92]. Ce protocole repose sur la caractérisation ainsi que la détection à la fois statique et dynamique des situations d'interactions dont les plus significatives sont présentées dans la suite (cf. Section 4).

Ce papier est organisé comme suit : la section 2 décrit le cadre d'expérimentation du modèle de planification distribuée proposé. Nous décrivons brièvement l'architecture d'un prototype de S.M.A. et le cas d'étude inspiré du scénario de transport de marchandises qui sera utilisé dans la suite. La section 3 présente notre modèle de plan d'un point de vue syntaxique pour la représentation des plans, donne la sémantique du modèle pour l'exécution, et fournit les algorithmes de fusion consistante de plans. La section 4 caractérise les différentes situations d'interaction et présente le protocole de coordination mis en place pour leur traitement. La section 5 développe un scénario d'exécution qui illustre les mécanismes décrits plus haut. Enfin, la section 6 présente une discussion relative à l'approche proposée.

2. Cadre d'expérimentation

Le modèle de planification distribuée que nous proposons, est indépendant du domaine et ne nécessite pas d'architecture particulière de S.M.A. cognitifs. Un tel modèle a été développé et intégré dans le prototype RAPID (Réseau d'Agents à Planification Interactive Distribuée) de S.M.A. Celui-ci a permis une expérimentation appliquée au diagnostic réparti du réseau téléphonique commuté dans le cadre d'une action thématique CNET. Cette expérimentation est en cours et ses résultats feront l'objet de publications ultérieures.

Le cas d'étude utilisé dans ce papier est inspiré du scénario de transport de marchandises largement repris dans le domaine des S.M.A. [MAR 92][FIS 93] pour l'hétérogénéité de ses agents en termes de rôles et pour la diversité de ses situations d'interactions.

2.1 Architecture du Prototype de S.M.A.

Le système multi-agents RAPID est composé d'un ensemble d'agents homogènes dans leur structure, mais se différenciant par leurs connaissances du domaine.

2.1.1 Organisation inter-agents

Nous avons choisi une implantation en couches afin de ne pas contraindre les choix conceptuels par des choix techniques (tels que la gestion des objets). Nous distinguons trois couches :

- la couche observation, qui permet à l'utilisateur de tracer l'exécution du S.M.A., chaque agent participant à cette trace. La technique que nous adoptons est celle des horloges vectorielles qui assure la reconstitution des ordres causaux [MAT 88] entre les messages échangés par les agents et les notifications au sein d'un même agent,
- la couche objet, qui fournit à la couche supérieure une transparence d'accès aux données locales ou distantes. Les objets peuvent migrer d'un agent vers un autre sans que ceci affecte le mécanisme d'accès. Ce dernier est basé sur les tables d'indices [RAY 92],
- la couche coopération, qui se focalise sur la gestion des interactions entre agents. Elle incorpore notre modèle de planification.

2.1.2 Organisation intra-agents

Un agent est composé de trois modules : le module de planification, le module de décision, le module de communication (cf. Figure 1). Ces trois modules fonctionnent en parallèle.

Au sein d'un même agent, les interactions entre les trois modules sont gérées par le contrôle intra-agents. Les modules accèdent à une mémoire partagée divisée en deux parties : la bibliothèque de plans et les objets gérés par l'agent. Les modules communiquent entre eux par envoi de notifications (messages intra-agents). Ces dernières sont émises dans les files d'attente associées aux modules concernés.
Dans ce qui suit, nous décrivons brièvement chacun de ces modules.

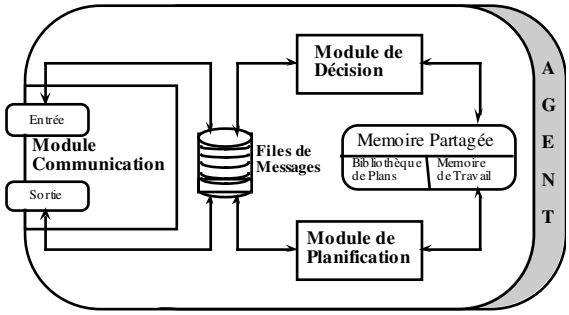

Figure 1 : *Structure d'un agent*

Le module de planification

Ce module assure l'exécution de plans. Un agent entrelace planification et exécution. Les connaissances de l'agent sont formées d'objets du domaine et de plans d'actions qui les manipulent. Les plans sont génériques (non instanciés) et regroupés dans une bibliothèque. Ils sont instanciés par le module de décision avec les objets du domaine. Le savoir-faire d'un agent réside dans ses squelettes de plans définis à la création et les méthodes associées aux actions. Une description détaillée du modèle de planification sera donnée à la section 3.

Le module de décision

La principale fonctionnalité du module de décision est le raffinement d'une action abstraite par un plan. Ce raffinement est notifié par le module de planification. L'arrivée d'une notification peut donner lieu : soit à un raffinement immédiat, soit à un raffinement en attente lorsque certaines ressources font défaut, soit encore à un achèvement de raffinement lorsque la notification avertit de la présence des ressources manquantes. Un raffinement nécessite l'instanciation des variables des squelettes de plans possibles qui forment des plans candidats, et le choix d'un plan candidat.

Le module de communication

Le module de communication assure la perception de l'environnement et les échanges entre agents au moyen de sa liste d'accointances. Ses fonctionnalités sont assurées par deux processus qui gèrent les communications entrantes et sortantes. Le processus de communication entrante est toujours à l'écoute des messages provenant de l'environnement ou des autres agents. Une fois un message arrivé, il le met en forme et prépare une notification qu'il dépose dans la file de messages du (des) module(s) concerné(s). Le flux de notifications est déterminé par le type des messages et est donc indépendant du contenu du message. Le processus de communication sortante récupère dans la file de messages des demandes d'interaction avec d'autres agents envoyées par les autres modules (i.e. demande

de fusion, envoi d'objet, envoi de plans, etc.). A partir du contenu des notifications, il prépare des messages qu'il envoie après consultation de la liste d'accointances à l'agent concerné. Ces messages véhiculent des informations appartenant aux trois couches conceptuelles décrites ci-dessus.

2.2 Exemple : scénario de transport de marchandises

Le transport est assuré par un système comportant deux types d'agents : des convoyeurs et des marchands. Un convoyeur évolue dans un réseau de transport maritime. Ce réseau est vu par le convoyeur comme un graphe dont les nœuds représentent les ports. Les arcs représentent des canaux entre ports que le convoyeur peut emprunter. Un convoyeur connaît tous les ports du réseau mais chaque convoyeur dispose d'une table de routage qui lui est spécifique. Cette table de routage indique à partir d'un port donné le(s) chemin(s) permettant d'atteindre un autre port.

Le but (ou activité) d'un convoyeur est de transporter des marchandises à travers les ports du réseau. Il assure ce transport à l'aide d'une péniche caractérisée par un volume. Un convoyeur dispose également de deux hangars sur chaque port : un hangar d'entrée et un hangar de sortie. Le premier sert à charger les marchandises devant être transportées, le second à décharger celles qui viennent d'être acheminées à leur destination. Un hangar ne peut contenir plus d'une marchandise à la fois, le volume n'ayant aucune incidence.
Un marchand est un client potentiel du convoyeur. Un marchand produit des marchandises sur un port de départ, puis les consomme sur un port d'arrivée. Il fait d'abord appel à un convoyeur pour assurer le transport de la marchandise entre les deux ports. Il place à cet effet sa marchandise dans le hangar d'entrée du convoyeur sur le port de départ. Une fois la marchandise livrée au port d'arrivée, le marchand la reprend dans le hangar de sortie de son convoyeur.

Notons que cet exemple est le support d'interactions qui nécessitent des mécanismes variés de coordination :

- il y a autonomie et synchronisation entre les activités des agents : le marchand dépose sa marchandise dans le hangar tandis que le convoyeur s'y rend ; le convoyeur doit cependant attendre le dépôt avant de reprendre sa route ;
- l'entrelacement de la planification et de l'exécution permet à un convoyeur de charger une marchandise alors qu'il en transporte déjà d'autres si la route empruntée par celui-ci rencontre des hangars pleins ;
- la gestion des ressources peut aussi conduire le convoyeur à décharger temporairement une marchandise pour en charger une autre afin d'améliorer l'utilité de son parcours.

Il va de soi que la difficulté dans le traitement de ces interactions est la contrainte de rester indépendant du domaine ; dans le cas contraire, des solutions ad hoc sont facilement concevables.

3. Modèle de plans

Généralement, un plan est un ensemble d'actions partiellement ordonnées et instanciées. Un tel plan peut être soit produit par un système de planification soit directement spécifié

par le concepteur de l'application. Dans le cadre des S.M.A. la gestion de plans doit d'une part intégrer un mécanisme de révision à l'exécution et d'autre part un mécanisme de fusion de plans pour la coordination des agents.

Aussi nous présentons d'abord le modèle théorique de plan adapté aux ordres partiels et basé sur une extension des réseaux de Petri. Puis nous enrichissons ce modèle par des fonctionnalités objet nécessaires à l'exécution d'un plan. Enfin, nous fournissons les opérateurs de composition de plans.

3.1. Modèle théorique : Réseaux de Petri Récursifs

3.1.1. Définitions syntaxiques

Notre formalisme est fondé sur des plans qui implantent des méthodes abstraites. Ces plans incluent à leur tour des "appels de méthodes abstraites ou élémentaires". Aussi les deux définitions suivantes se référencent mutuellement. Au niveau du modèle théorique, les méthodes sont de simples mécanismes d'exécution dont l'aspect données est absent.

Définition 1 : Méthode

Une méthode est définie par :

- un label,
- un type : méthode abstraite ou élémentaire,
- un ensemble de Réseaux de Petri Récursifs dans le cas d'une méthode abstraite.

Un réseau de Petri Récursif se présente comme un réseau de Petri ordinaire acyclique et dont les nœuds sources sont des places. Cette description est enrichie par un typage des transitions auxquelles sont associées des méthodes. Une transition élémentaire se comporte comme une transition de réseau de Petri. Le franchissement d'une transition abstraite donne lieu à l'activation d'un réseau de Petri récursif implantant la méthode associée à la transition. Une transition de fin achève un plan et par conséquent le franchissement de la transition correspondante dans le réseau de niveau supérieur.

Définition 2 : Réseau de Petri Récursif

Un réseau de Petri récursif $R = <P,T, W^-, W^+, M_0>$ est défini par :

- P, l'ensemble des places du réseau,
- T, l'ensemble des transitions du réseau. Il est constitué de l'union disjointe d'un ensemble de transitions élémentaires T_{elem}, un ensemble de transitions abstraites T_{abs} , et un ensemble de transitions de fin T_{fin} :

  $T = T_{elem} \oplus T_{abs} \oplus T_{fin}$ où $\oplus$ représente l'union disjointe.

A chaque transition t est associée une méthode dénotée *meth(t).* Si la transition est abstraite la méthode l'est aussi, sinon elle est élémentaire. L'ensemble des transitions

abstraites T_{abs} est constitué de l'union disjointe d'un ensemble de transitions parallèles T_{par} et d'un ensemble de transitions exclusives T_{exc} :

$$T_{abs} = T_{par} \oplus T_{exc}$$

- W^- est la matrice d'incidence arrière de P x T dans $\mathbb{N}$,
- W^+ est la matrice d'incidence avant de P x T dans $\mathbb{N}$,
- M_0 est le marquage initial défini comme un vecteur de P dans $\mathbb{N}$, dénoté par le multi-ensemble de P : Σ M(p).p.

Remarque : le modèle théorique ne distingue pas sémantiquement les transitions exclusives et les transitions parallèles.

La figure 2 donne un exemple de représentation de plan d'actions au moyen d'un Réseau de Petri Récursif qui modélise le traitement d'une marchandise par un agent convoyeur. La partie gauche de la figure spécifie le plan abstrait qui consiste à se déplacer au port de résidence de la marchandise puis à la transporter à destination. La partie droite montre un raffinement possible pour la méthode abstraite "Aller_à". Ce raffinement consiste à se déplacer en un port voisin puis replanifier le chemin vers la destination finale en appelant récursivement la transition abstraite "Aller_à".

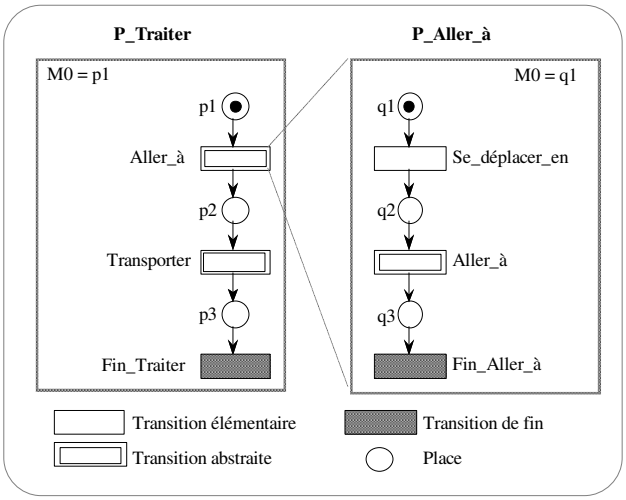

Figure 2 : *Représentation de plans par RdPR*

3.1.2. Définitions sémantiques

La définition précédente spécifie la syntaxe d'un RdPR. Nous en donnons maintenant une sémantique opérationnelle en termes d'état et de changement d'état. Un état d'un RdPR est un arbre dont les nœuds sont étiquetés par des réseaux marqués et les arcs par des transitions abstraites.

L'interprétation intuitive d'un état est la suivante :

- la racine de l'arbre correspond au plan initial,
- chaque arc représente un franchissement de transition abstraite où l'extrémité de l'arc dénote le réseau déplié,
- cette structure est similaire à une pile d'exécution à la différence près que le comportement parallèle d'un RdPR engendre un arbre "d'appels de transitions abstraites".

Définition 3 : Marquage étendu

Un marquage étendu est un arbre Tr = <S, A>, où :

- S : ensemble des sommets où chaque sommet s est étiqueté par un RdPR marqué, noté <R(s), M(s)>
- A = { <s, s'> / s' est fils de s },
- un arc a est étiqueté par une transition abstraite trans(a),
- l'état initial Tr_0 est un arbre réduit à un sommet s tel que R(s) est le RdPR initial et M(s) = M_0 .

La sémantique du franchissement d'une transition s'interprète ainsi dans l'arbre :

- un franchissement d'une transition élémentaire d'un nœud de l'arbre modifie le marquage de ce nœud,
- un franchissement d'une transition abstraite d'un nœud de l'arbre consomme les jetons en entrée de la transition, crée un fils à ce nœud dont l'arc est étiqueté par la transition abstraite et la nouvelle feuille est étiquetée par l'un des réseaux implantant la méthode abstraite associée,
- un franchissement d'une transition de fin d'un nœud de l'arbre élimine le sous-arbre issu de ce nœud, et produit les jetons en sortie de la transition abstraite qui a généré ce nœud.

La définition suivante formalise ces interprétations.

Notations :

- Soit Tr un arbre, on note Tr(s) le sous-arbre de racine s et PRUNE(Tr, s) l'arbre élagué Tr \ Tr(s).
- W^- (., t) (resp. W^+(., t)) est un vecteur de P à valeurs dans $\mathbb{N}$ correspondant à la colonne de t obtenue à partir de la matrice W^- (resp. W^+).
- Chaque fois qu'une structure dépendra de l'arbre considéré on l'indicera par celui-ci (i.e. A_{Tr} dénote l'ensemble des arcs A de l'arbre Tr).

Définition 4 : Règles de franchissement

** Condition de franchissabilité*

Une transition *t* est franchissable à partir du nœud si et seulement si :

$$M(s) \geq W^-(., t)$$

** Transition élémentaire*

Soit $t \in T_{elem}$ d'un RdPR associé à un nœud s de Tr, le franchissement de t à partir de s produit l'arbre Tr' tel que :

- ◊ $Tr' = (S_{Tr'}, A_{Tr'})$ avec :
  - $S_{Tr} = S_{Tr'}$ tel que $\forall\ s' \neq s$, $M_{Tr'}(s') = M_{Tr}(s')$ (inchangé pour tout $s' \neq s$),
  - $A_{Tr} = A_{Tr'}$,
- ◊ $M_{Tr'}(s) = M_{Tr}(s) + W^{+}(., t) - W^{-}(., t)$

** Transition abstraite*

Soit $t \in T_{abs}$ d'un RdPR associé à un nœud s de Tr, le franchissement de t à partir de s produit l'arbre Tr' tel que :

- ◊ $Tr' = (S_{Tr'}, A_{Tr'})$ avec :
  - $S_{Tr'} = S_{Tr} \cup \{s_n\}$ où le RdPR associé au nouveau sommet s_n est l'un des réseaux implantant meth(t) et $M_{Tr'}(s_n) = M_0$ (marquage initial d'appel associé à ce RdPR),
  - $A_{Tr'} = A_{Tr} \cup \{a\}$ où $a = \langle s, s_n \rangle$ et $trans_{Tr}(a) = t$,
- ◊ $M_{Tr'}(s) = M_{Tr}(s) - W^{-}(., t)$.

** Transition de fin*

Soit $t \in T_{fin}$ d'un RdPR associé à un nœud s de Tr, le franchissement de t à partir de s produit l'arbre Tr' tel que :

- ◊ $Tr' = PRUNE(Tr, s)$
  Soit s' le père de s dans l'arbre Tr et $a = \langle s', s \rangle$ l'arc étiqueté par t, alors :
  - $\forall\ a \in A_{Tr'}$, $trans_{Tr'}(a) = trans_{Tr}(a)$,
  - $\forall\ s'' \in (S_{Tr'} \setminus \{s'\})$, $M_{Tr'}(s'') = M_{Tr}(s)$,
- ◊ $M_{Tr'}(s') = M_{Tr}(s) + W^{+}(., trans_{Tr}(a))$.

La figure 3 illustre les règles de franchissements d'une transition abstraite suivie d'une transition élémentaire à partir du RdPR donné en figure 2. A gauche du nœud se trouve le nom du sommet et à droite son marquage courant. Par souci de simplification, nous n'avons pas indiqué le nom des réseaux. On peut noter que si la place P1 était marquée deux fois, alors le nœud initial admettrait une seconde instance de franchissement pour la transition "Aller_à" et donc un deuxième fils.

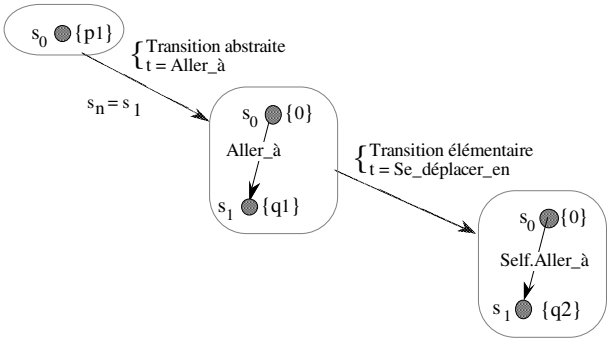

Figure 3 : *Evolutions successives de l'arbre d'exécution.*

3.1.3 Discussion du modèle

Le formalisme des Réseaux de Petri Récursifs (RdPR) offre plusieurs avantages pour la représentation des plans et leur exécution :

- il fournit une expression naturelle et graphique qui permet d'exprimer l'ordre partiel des tâches à exécuter ainsi que des relations temporelles complémentaires telles que la disjonction d'intervalles d'exécution.
- il prend en compte l'incertitude de l'environnement durant l'exécution soit à l'aide de transitions alternatives au sein d'un même plan, soit lors d'un raffinement par le choix d'un sous-plan implantant la méthode,
- la notion de transition abstraite permet de considérer un plan à différents niveaux d'abstraction et donc de planifier de façon hiérarchique,
- la phase de vérification de la consistance d'un plan repose sur la production d'un graphe des états d'exécutions d'un plan. Or les techniques de réduction de graphes d'états issus des réseaux de Petri assurent un gain important en complexité [CHI 91][VAL 91][GOD 94].
- la coordination de plans repose sur les mécanismes de fusion de réseaux de Petri et donc indépendant du domaine d'application (cf. § 3.3).

3.2. Modèle d'exécution de plans

Le modèle d'exécution de plans enrichit le modèle théorique des RdPR par des fonctionnalités objet. Les objets décrivent le(s) monde(s) réel(s) ou virtuel(s). Les méthodes spécifient les moyens d'agir sur ce(s) monde(s).

Les objets

Les objets représentent les données du domaine. Ils permettent d'instancier les variables des plans et les paramètres des actions. Un objet est composé d'attributs. La valeur de

chaque attribut est soit un symbole, soit une valeur terminale. A chaque objet est associé un qualificateur qui indique son type. Dans le modèle actuel, le contrôle de type se limite à la comparaison des qualificateurs pour l'instanciation de variables.

Les variables d'un plan

Un RdPR représente un plan générique (ou squelette de plan) qui sera instancié au moment de l'exécution pour s'appliquer à des objets différents. La généricité est assurée à travers la définition de variables qualifiées. Ces variables apparaissent dans les expressions des paramètres d'appel des méthodes associées aux transitions du plan.

Les méthodes

Les méthodes traduisent le "savoir-faire" d'un agent. Elles comportent des paramètres qui seront instanciés par des objets lors de l'appel. Le franchissement d'une transition appelle la méthode associée. Aussi, à chaque paramètre formel est associée une expression composée par les variables du plan. De plus, nous définissons pour chaque méthode des pré- et des post-conditions qui expriment des conditions de succès sur les attributs des objets liés aux paramètres formels. Le couple (pré-, post-conditions) exprime le fait que si les pré-conditions d'une méthode sont vérifiées, alors son exécution rendra vraies les post-conditions (comme dans le formalisme STRIPS [FIK 71]). Ainsi les post-conditions des transitions de fin d'un plan peuvent être vues comme les buts de celui-ci. De même, le plan peut être considéré comme valide si, étant données les conditions initiales, tous les ordonnancements de franchissements possibles induisent des appels de méthodes dont les pré-conditions sont vérifiées (voir l'algorithme de consistance donné ci-dessous).

Les opérations de raffinement

Dans le modèle d'exécution, le choix du plan de raffinement n'est pas totalement indéterministe contrairement au modèle théorique. En effet, un plan comporte une condition de raffinement qui est une expression portant sur les paramètres de la méthode abstraite que le plan doit raffiner. Cette condition doit être vérifiée pour que le plan soit un raffinement valide.

3.3 Consistance, composition et transformation de plans

Dans notre modèle, la coordination entre les agents induit des compositions de plans. Une coordination n'est possible que si le plan composé reste consistant. Nous présentons donc successivement le test de consistance d'un plan et les fusions par transition ou par place de deux plans.

3.3.1 Consistance d'un plan

Un RdPR modélise un plan dont il est important de vérifier la possibilité d'exécution. Cette vérification se déroule en deux étapes :

1. on construit un graphe réduit (cf. discussion ci-dessous) à partir de l'arbre des marquages accessibles du RdPR (modélisant le plan) considéré alors comme un réseaux de Petri ordinaire. L'arbre (et donc le graphe réduit) est toujours fini car le RdPR est acyclique et ses nœuds sources sont des places.

2. on invoque un algorithme de calcul de consistance de plan (appelé C.P.P.C.) basé sur les Pré- et Post-Conditions des transitions du RdPR. et qui prend en entrée le graphe réduit. Ce parcours évalue (fonction Evaluer) à chaque étape la satisfaction des pré-conditions associées aux arcs issus du nœud courant en démarrant à partir du nœud initial, puis s'appelle récursivement sur les nœuds fils en ayant au préalable appliqué (fonction Appliquer qui renvoie un nouveau contexte) sur le contexte courant les post-conditions. Un contexte est ici défini par les attributs des objets manipulés par le plan.

Notations

On notera s,s' les sommets du graphe réduit, et s[t>s' un arc dans ce graphe reliant s à s' et étiqueté par la transition t. Cette notation est une notation standard des réseaux de Petri.
Chaque sommet s est doté d'un attribut "s.visité" qui indique si le nœud a été visité par l'algorithme.

Algorithme de consistance(C.P.P.C.)

```
Fonction Est_Faisable (in s : nœud; in c : contexte) : Booléen;
Si s.visité Alors
      Renvoyer(VRAI)
Sinon
      s.visité := VRAI
      Pour_Chaque arc  s[t>s'  Faire
            Si     Evaluer (Pré (Meth (t)), c)    Alors
                  Si     Non (Est_Faisable(s', Appliquer(Post(Meth(t)), c))) Alors
                        Renvoyer (FAUX)
                  Fin_Si
            Sinon
                  Renvoyer (FAUX)
            Fin_Si
      Fin_Pour
      Renvoyer (VRAI)
Fin_Si
```

Discussion

Nous ne présentons pas la construction du graphe qui met en œuvre de nombreuses techniques de réduction. Nous nous contentons ici d'indiquer deux méthodes de réduction parmi les plus simples :

Soient t1 et t2 deux transitions du RdPR,

- t1 et t2 sont indépendantes s'il n'y a pas d'interférences entre les pré- et post-conditions de t1 d'un côté, et les pré- et post-conditions de t2 de l'autre. Les deux transitions peuvent être franchies simultanément. L'ordre de leur franchissement n'a

aucune incidence sur la faisabilité de l'exécution du plan global. Dans ce cas un ordre de franchissement arbitraire est décidé permettant ainsi d'élaguer de nombreux sous-arbres.

- t1 et t2 sont semi-indépendantes s'il n'y a pas d'interférences entre les post-conditions de t1 et t2. Les deux transitions affectent des attributs différents. Deux séquences de franchissement obtenues l'une de l'autre par permutation de t1 et t2 conduisent au même contexte. Dans ce cas, on peut identifier le nœud atteint par une séquence à celui de l'autre séquence.

Il est également possible à l'aide de techniques de vérification à la volée de diminuer la complexité en espace au détriment de la complexité en temps [JAR 91].

3.3.2 Fusion par transition

La fusion par transition de deux plans se fait en deux phases :

1. les transitions de chaque plan associées à des méthodes identiques sont fusionnées si l'unification des paramètres d'appel est possible. Les effets de l'unification sont propagés dans les plans. Dans le cas contraire, il y a échec de la fusion.
2. le calcul de consistance est appliqué au plan fusionné. En cas d'inconsistance, il y a échec de la fusion.

Il est à noter que l'unification est vue ici comme un outil de synchronisation pour la programmation parallèle et généralise le concept d'appel de procédure à distance. En effet, les paramètres d'exécution de la méthodes sont fournis indifféremment par les deux agents.

3.3.3 Fusion par place (séquentialisation/parallélisation)

La fusion par place se fait en deux phases. La première phase consiste à contraindre l'exécution des actions du second plan, à se dérouler après celles du premier. Les contraintes induites par cette séquentialisation sont ensuite progressivement relâchées tant que la consistance reste vérifiée. Cet algorithme dit de parallélisation est une généralisation de l'algorithme C.O.A. [ELF 96a].

1. Séquentialisation

La séquentialisation consiste à connecter deux plans P1 et P2. L'algorithme crée un ensemble de places et d'arcs de synchronisation. Une place est créée pour chaque paire composée d'une transition de fin dans P1, et d'une transition initiale (i.e. sans transition prédécesseur) dans P2. L'algorithme renvoie le plan résultant de la fusion et estampille les places de synchronisation. Cet estampillage servira pour la phase de parallélisation.

L'algorithme de consistance est ensuite appliqué au plan fusionné. En cas d'inconsistance, il y a échec de la fusion.

Fonction Séquentialisation (in P1 : Plan, in P2 : Plan) : Plan
{Renvoie un plan P3 où les places de synchronisation sont estampillées}
Soient
$E_1 = \{t_i \in P1 / t_i$ est une transition de fin$\}$
$E_2 = \{t_j \in P2 / t_j$ est sans transition prédécesseur dans P2$\}$

```
P3 = P1 ∪ P2
Pour_chaque (t_i, t_j) ∈ E1 x E2 faire
Début
        Créer dans P3 une place p_i,j estampillée
        Créer dans P3 un arc AE_i,j : t_i → p_i,j        {Créer un arc en entrée}
        Créer dans P3 un arc AS_i,j : p_i,j → t_j        {Créer un arc en sortie}
FinPour
Renvoyer (P3)
```

2. Parallélisation

Le but de cet algorithme est d'introduire un maximum de parallélisme dans le plan résultant de la séquentialisation. Le principe consiste à essayer de remonter (ou éliminer) les places de synchronisation. Plus précisément, chaque pas de l'algorithme s'applique à deux transitions connectées par une place de synchronisation et dont la transition amont n'a pas débuté son franchissement et dont les transitions prédécesseur n'ont pas terminé le leur. On produit une place de synchronisation par transition prédécesseur de la transition amont et on supprime la place initiale. Puis on procède symétriquement pour la transition avale. Si la consistance de ce nouveau plan est assurée, alors il se substitue à l'ancien. On procède ainsi itérativement.

```
Procédure Parallélisation (inout PL : Plan)
    Tantque (∃ p place de PL / p estampillée) Faire
        Soient
            t-in la transition en entrée de p
            t-out la transitions en sortie de p
            T-Prec l'ensemble des transitions prédecesseurs de t-in
            T-Succ l'ensemble des transitions successeurs de t-out
            P-Prec l'ensemble des places prédecesseurs de t-in
        Si
            ∀ p_i ∈ P-Prec, p_i n'est pas marquée et
            ∀ t_j ∈ T-Prec, t_j n'est pas en cours d'exécution
        Alors
            P2 = copie (PL)
            Pour Chaque t ∈ T-Prec Faire
                Créer dans P2 une place estampillée de t à t-out
            Fin_Pour
            Pour Chaque t ∈ T-Succ Faire
                Créer dans P2 une place estampillée de t-in à t
            Fin_Pour
            Si  Est_Faisable (P2, Contexte_Courant)  Alors
                PL = P2
            Sinon
                oter _ estampille (p)
            Fin_Si
        Sinon
            oter _ estampille (p)
        Fin_Si
```

FinTantQue

Discussion

Contrairement aux opérations déjà vues sur le plan, la parallélisation tient compte de l'état d'exécution du plan, ceci afin que les synchronisations générées soient effectives (i.e. toute place de synchronisation sera marquée).

Il est à noter que cet algorithme peut être optimisé au niveau du contrôle de consistance. En effet, ce test pourrait être réalisé de manière incrémentale à partir des tests sur chacun des plans P2 précédents. Cette optimisation n'est toutefois pas traitée dans le présent papier.

4. Protocole de traitement des interactions

4.1 Caractérisation des situations d'interaction

Les situations d'interaction prises en compte par notre approche couvrent aussi bien les interactions positives que négatives. Dans ce qui suit, nous présentons les situations les plus significatives.

4.1.1 Interactions Positives

Les interactions positives traduisent des situations où les actions d'un agent facilitent l'exécution du plan d'un autre agent. Nous distinguons deux types d'interactions positives.

Actions Redondantes

Deux actions sont dites redondantes si elles apparaissent au moins dans deux plans appartenant à deux agents différents, ont les mêmes pré- et post-conditions, et leurs méthodes sont instanciées par les mêmes paramètres sauf le paramètre agent (i.e. identité de l'agent qui exécute l'action). Dans ce cas, le processus de la coordination affecte l'exécution de l'action à un des agents. Ce dernier exécute l'action et en fournit les résultats aux autres qui modifient leur plan par l'ajout d'une transition de synchronisation.

Actions Favorables

Deux actions sont dites favorables si l'exécution de l'une rend valides toutes ou certaines pré-conditions de l'autre. L'exécution de cette dernière sera respectivement possible ou favorisée.

4.1.2 Interactions négatives

Les interactions négatives traduisent des situations où les actions d'un agent contrecarrent l'exécution du plan d'un autre agent. Nous distinguons trois types d'interactions négatives.

Actions Défavorables

Deux actions sont dites défavorables si l'exécution de l'une rend insatisfaites toutes ou certaines pré-conditions de l'autre. Ainsi, l'exécution de cette dernière sera respectivement impossible ou désavantagée.

Une telle situation doit être détectée avant l'exécution d'un nouveau plan afin de prédire d'éventuels échecs ou blocages. Notre mécanisme de coordination introduit un ordre entre les actions défavorables comme dans [ELF 96] qui offre un algorithme de coordination (COA) pour le traitement de telles interactions impliquant plusieurs agents à la fois.

Actions Exclusives

Deux actions sont dites exclusives si l'exécution de l'une empêche momentanément l'exécution de l'autre (i.e. leur exécution requiert une même ressource non consommable). Détectée dynamiquement, cette situation apparaît suite au franchissement d'une transition exclusive qui inhibe les autres exécutions. Dans ce cas, l'exécution reste possible mais doit être différée en conformité avec la règle de franchissement d'une transition exclusive.

Actions Incompatibles

Deux actions sont dites incompatibles si l'exécution de l'une empêche définitivement l'exécution de l'autre (i.e. leur exécution requiert une même ressource consommable). Dans notre modèle, une telle situation modélise une alternative (i.e. deux transitions partagent une même place en entrée contenant un seul jeton). Dans ce cas, l'exécution n'est possible que si la source de conflit (i.e. ressource critique) peut être remplacée.

4.2 Détection des situations d'interactions

Les situations d'interaction sont dans la plupart des autres approches détectées statiquement [MAR 92][DUR 91][DEC 92]. Cependant, l'entrelacement entre planification et exécution de plans requiert une détection à la fois statique et dynamique, ce que nous nous proposons de réaliser grâce à la sémantique des RdPR.

Dans ce modèle, une situation d'interaction positive peut se présenter lors de la phase de raffinement d'une transition abstraite (e.g. une des actions du plan requiert une compétence non locale). Ceci se traduit syntaxiquement lorsque la variable qui désigne l'agent susceptible d'effectuer l'action est d'un type différent de celui de l'agent local.

Les interactions négatives sont détectées lorsque des actions appartenant à des plans différents manipulent des objets de manière concurrente. Dans notre modèle, cette situation correspond au raffinement d'une transition abstraite exclusive. Si une instance de franchissement de cette transition est déjà en cours, alors le nouveau franchissement doit être fusionné avec le précédent afin de coordonner les accès aux ressources de l'agent désignées par les objets communs aux deux plans.

4.3 Le protocole d'interaction

Ce protocole repose exclusivement sur l'échange de plans. Lorsqu'une situation d'interaction positive est détectée, l'agent sélectionne à l'aide de sa liste d'accointances un agent possédant la compétence requise et lui envoie le sous-plan contenant l'action à déléguer. A la réception, l'agent détecte dans le plan reçu les actions qu'il est susceptible d'exécuter (détection syntaxique). Pour chacune de ces actions, il parcourt son arbre de plans courants à la recherche d'actions correspondantes. Si nécessaire, il raffine une transition abstraite franchissable. Cette phase se termine avec succès si le RdPR associé à un nœud de l'arbre contient toutes les actions à exécuter. Il procède alors à une fusion par transition du plan reçu avec le plan choisi. Si la fusion est possible (i.e. préserve la consistance) alors il renvoie le sous-plan de l'agent demandeur instancié et enrichi des synchronisations (places et transitions) avec son propre plan.

Chaque agent reprend son exécution ; la synchronisation sera assurée par des messages correspondants aux fins de franchissement des transitions de synchronisation.

Une situation d'interaction négative peut être détectée aussi bien localement qu' éventuellement lors du traitement d'une interaction positive. Dans ce cas, le traitement de l'interaction négative reste transparent à l'agent demandeur.

L'interaction négative est résolue par la séquentialisation/parallélisation (i.e. fusion par place) des deux plans correspondant à la même transition exclusive. Ainsi dans l'arbre d'exécution d'un agent une transition exclusive ne peut étiqueter qu'au plus un fils de chaque nœud.

5. Scénario d'exécution

Cette section présente le fonctionnement du S.M.A. à travers l'exemple décrit dans la section 2. Le scénario développé ici, implique trois agents : deux marchands et un convoyeur. Il illustre une situation d'interaction positive entre un marchand et un convoyeur suivie d'une deuxième interaction positive dont la résolution fait apparaître une situation d'interaction négative.

Le plan du marchand (cf. partie gauche de la figure 4) comporte quatre actions séquentielles : *déposer* la marchandise (dans le hangar d'entrée de Y correspondant à la localisation source de la marchandise), la transporter, la retirer (du hangar de sortie de Y correspondant à la localisation destination de la marchandise), et la transition de fin "distribuer ". Le transport ne peut être réalisé par le marchand comme l'indique le type de la variable Y (La variable Self désigne dans tout plan l'agent qui l'a généré). Le plan correspondant du convoyeur (cf. partie droite de la figure 4) est celui décrit en section 3.1.1. enrichi par les données.

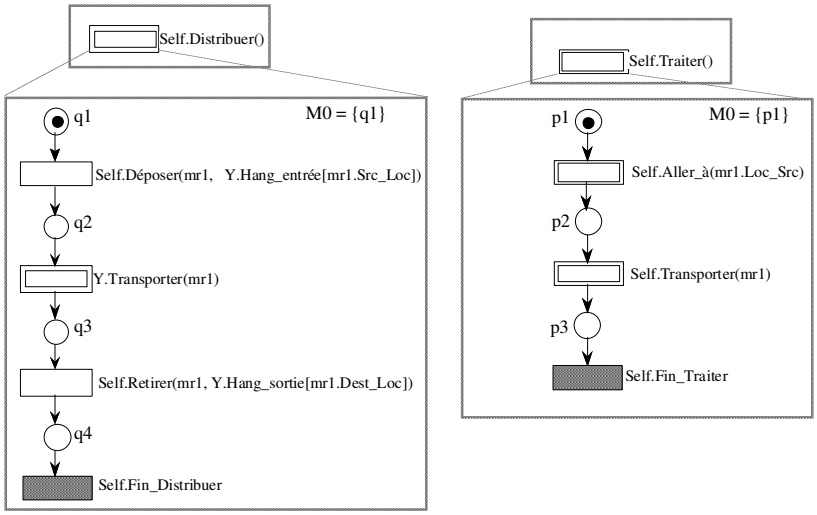

Figure 4 : *Plans du marchand et du convoyeur*

La fusion par transition est représentée en figure 5. On notera que l'unification a induit un renommage des variables Self (M1 dans le plan du marchand et C1 dans le plan du convoyeur).

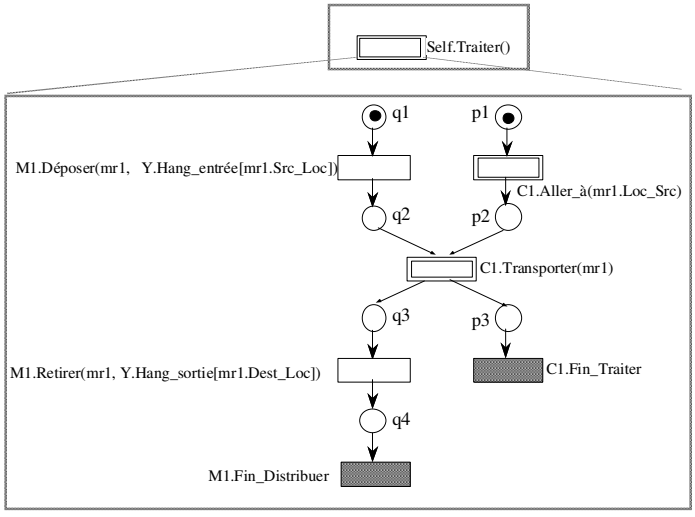

Figure 5 : *Fusion par transition*

Une fois la fusion achevée, chaque agent reprend l'exécution de son plan enrichi (cf. Figure 6). Dans le plan du marchand M1, la variable Y a été instanciée par l'identité du convoyeur “C1”. Dans le plan du convoyeur, la variable X a été instanciée par la marchandise “ mr1” et la place Q2 a été ajoutée pour s'assurer que la marchandise sera chargée avant son transport!

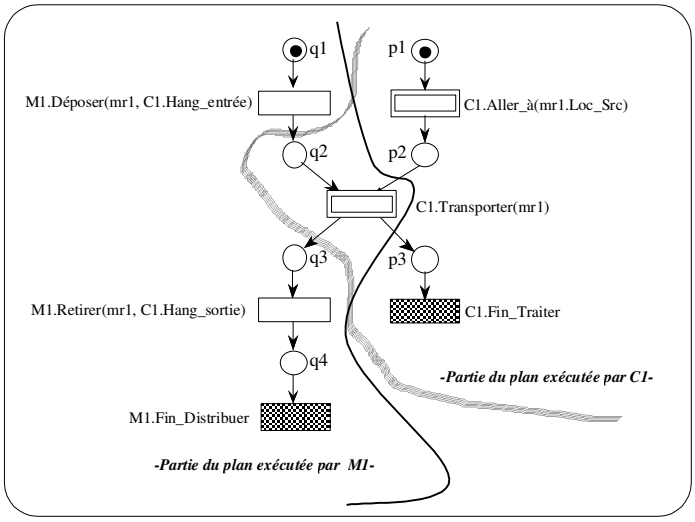

Figure 6 :*Exécution du plan fusionné*

Lors de la réception du plan du second marchand, le convoyeur procède de manière similaire. Cependant, la transition abstraite "Traiter" ayant généré ce second raffinement est une transition exclusive. Le convoyeur fusionne par place les deux raffinements. Le résultat de cette fusion est donné en figure 7. Par souci de simplification, la parallélisation qui est dépendante du contexte courant d'exécution a été omise.

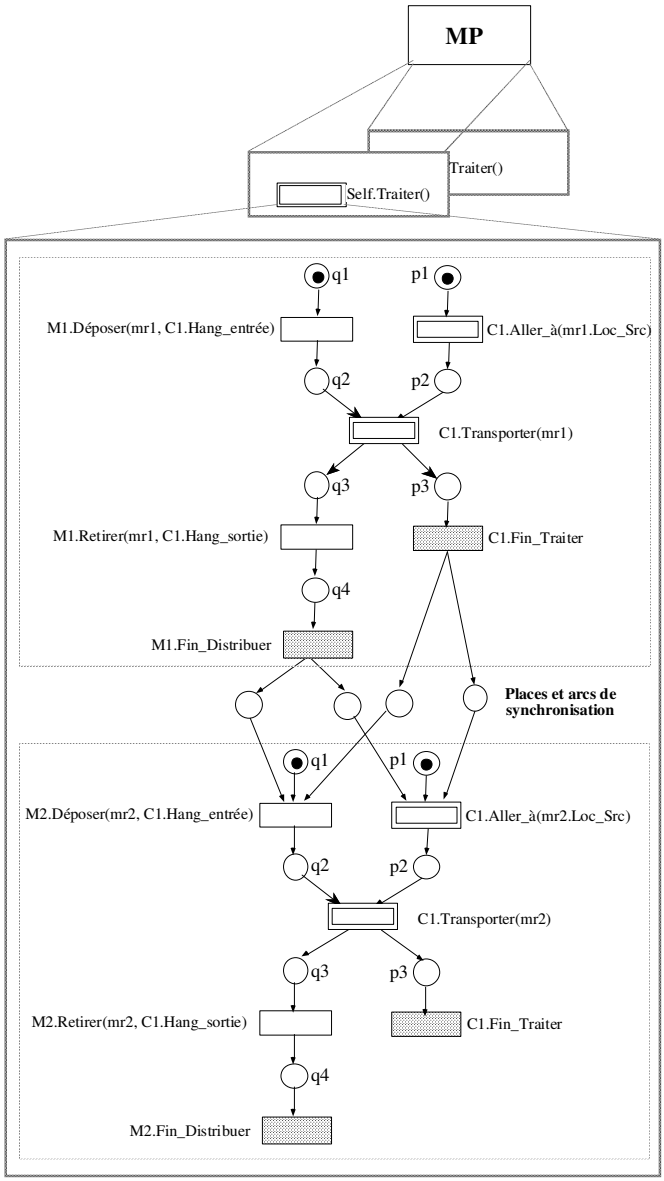

Figure 7 : *Fusion par place*

6. Discussion

Les situations d'interaction entre agents impliquent une coordination des actions des agents. Dans les approches proposées jusqu'alors les mécanismes de coordination présentaient l'un ou l'autre des inconvénients suivants :

- ils sont dépendants du domaine,
- ils traitent la coordination uniquement lors de phase de planification,
- ils ne gèrent pas à la fois les interactions positives et les interactions négatives,
- ils reposent sur des heuristiques,
- ils impliquent une centralisation du traitement et une suspension des activités des agents.

La solution que nous proposons ici, repose sur le modèle des RdPR qui recouvre les principales caractéristiques de la planification distribuée (telles que la récursivité, la dynamicité et l'entrelacement de la planification et de l'exécution) tout en étant indépendant du domaine. Il permet de représenter et de raisonner sur des actions simultanées et des processus continus (i.e. actions concurrentes, choix parmi plusieurs alternatives, synchronisation, etc.). Ce modèle gère l'abstraction des actions ce qui permet de choisir de manière dynamique la façon dont on exécute une action. De plus, il permet de vérifier la consistance des plans produits à l'aide d'algorithmes efficaces. Enfin, la fusion de RdPR est le support de la résolution des interactions positives et négatives avec la garantie de la consistance des traitements.

Afin d'élargir le champ d'application de notre modèle, nous poursuivons ces travaux dans plusieurs axes complémentaires. Tout d'abord, nous voulons étendre la taxonomie et le traitement des situations d'interaction notamment pour les interactions positives. La reconnaissance des situations d'interactions négatives devrait intégrer la gestion de ressources partagées distantes. Enfin, il est nécessaire de généraliser le modèle de planification. D'une part, nous cherchons à élaborer un mécanisme de synthèse de plans qui interagisse avec le mécanisme de coordination. D'autre part, notre approche doit prendre en compte les éventuels échecs d'actions élémentaires afin de replanifier en conséquence.

Remerciements

Ce travail de recherche a été en partie soutenu par FranceTelecom, CNET, sous le contrat *CTI 904-15-4371-123*. Nous tenons à remercier Jean-Philippe Coumes (CNET Paris) et Gilles Deflandre (CNET Lannion) pour leurs intéressantes discussions.

Bibliographie

[ALA 94] Alami, R., Robert, F., Ingrand, F. and Suzuku, S. *"A Paradigm for Plan-Merging and its use for Multi-Robot Cooperation"*. In Proceedings of IEEE International Conference on Systems, Man and Cybernetics, San Antonio, Texas (USA) ; 1994.

[BON 88] Bond, A.H., and Gasser, L., (Eds), *"Readings in Distributed Artificial Intelligence"*. Morgan Kaufmann Publishers, Inc ; 1988.

[BRI 95] Briggs, W., and Cook, D. *"A Flexible social laws "*. In Proceedings of IJCAI'95, pp. 688-693 ; 1995.

[CHI 91] Chiola, G., Duteillet, C., Franceschinis, G., and Haddad, S. *"On Well-Formed Coloured Petri Nets and their Symbolic Reachability Graph"*. In. High Level Petri Nets, theory and applications. Springer-Verlag, Jensen, K. and Rozenberg, G. (Eds.) ; 1991.

[COR 79] Corkill, D.D. *"Hierarchical Planning in Distributed Environment"*. In Proceedings of IJCAI'79, pp. 168-175 ; 1979.

[DAV 83] Davis, R., and Smith, R.G., *"Negotiation as a Metaphor for Distributed Problem Solving"*. Artificial Intelligence, 20(1), pp. 63-109 ; 1983.

[DEC 92] Decker, K.S., and Lesser, V.R., *"Generalizing the Partial Global Planning Algorithm"*. In International Journal on Intelligent Cooperative Information Systems. pp. 319-346 ; 1992.

[DUR 91] Durfee, E.H., and Lesser, V.R., *"Partial Global Planning : A Coordination Framework for Distributed Hypothesis Formation"*. IEEE Transactions on Systems, Man and Cybernetics, vol. 21, no. 5, pp. 1167-1183 ; 1991.

[ELF 95] El Fallah-Seghrouchni A., and Haddad S. *"A Formal Model for Coordinating Plans in Multi-Agent Systems"*. In Proceedings of Intelligent Agents Workshop. Augusta Technology Ltd, Brooks University, Oxford, UK ; 1995.

[ELF 96a] El Fallah-Seghrouchni A., and Haddad S. *"A Coordination Algorithm for Multi-Agent Planning"*. In Proceedings of MAAMAW'96. LNAI : 1038. Ed. Springer Verlag. Eindhoven, Netherlands ; 1996.

[ELF 96b] El Fallah-Seghrouchni A., and Haddad S. *"A Recursive Model for Distributed Planning"*. In Proceedings of ICMAS'96 (International Conference of Multi-Agent Systems). AAAI Press. Kyoto, Japon ; 1996.

[ELF 97] El Fallah-Seghrouchni A., and Haddad S. *"A Framework for Agent Coordination Oriented Distributed Planning and Resource Sharing Management"*. Soumis à publication ; 1997.

[EPH 95] Ephrati, E., and Rosenschein, J.S. *"A Framework for the Interleaving of Execution and Planning for Dynamic Tasks by Multiple Agents"*. In Proceedings of ATAL'95 ; 1995.

[FER 95] Ferber, J. *"Les Systèmes Multi-Agents : vers une intelligence collective"*. Inter-Editions ; 1995.

[FIK 71] Fikes, R.E., and Nilson, N.J. *"STRIPS: a New Approach to the Application of Theorem Proving to Problem Solving"*. Artificial Intelligence 2, pp. 189-208 ; 1971.

[FIS 93] Fisher, K., and Kuhn, N. *"A DAI approach to modelling a transportation domain"*. Technical report, RR-93-25. DFKI ; 1993.

[GAS 84] Gasser, L. *"The Social Dynamics of Routine Computer Use in Complex Organizations"*. Ph.D. thesis, Department of Computer Science, University of California, Irvine ; 1984.

[GEO 83] Georgeff, M.P. *"Communication and Interaction in Multi-Agent Planning"*. In Proceedings of AAAI'93, pp. 125-129 ; 1983.

[GEO 88] Georgeff, M.P. *"A Theory of Action for Multi-Agent Planning"*. In Readings in Distributed Artificial Intelligence, A.H. Bond and L. Gasser (Eds), Morgan Kaufmann Publishers, Inc , pp. 205-209 ; 1988.

[GEO 90] Georgeff, M.P. *"Planning"*. In Readings in Planning. Morgan Kaufmann Publishers, Inc. San Mateo, California ; 1990.

[GOD 96] Godefroid, P. *"Partial-Order Methods for the Verification of Concurrent Systems."* Springer-Verlag . LNCS 1032 ; 1996.

[JAR 91] Jard, C. and Jeron, T. *"Bounded-memory algorithms for verification on-the-fly"*. 192-202. Springer-Verlag. LNCS 575 ; 1991.

[JEN 91] Jensen, K., and Rozenberg, G. (Eds.). *"High Level Petri Nets, Theory and Applications"*. Springer-Verlag ; 1991.

[MAR 92] von Martial, F. *"Coordination plans of autonomous agents"*. Springer-Verlag ; 1992.

[MAT 88] Mattern, F. *"Virtual time and global states of distributed systems"*. In Proceedings of Parallel and Distributed Algorithms Conference, (Cosnard, Quinton, Raynal, Robert Eds), North-Holland, pp. 215-226 ; 1988.

[RAY 91] Raynal, M. *"La communication et le temps dans les réseaux et les systèmes répartis"*. CEA-EDF-INRIA, Ecole d'été d'informatique, Eyrolles CDEREF n°. 75.

[RAY 92] Raynal M. *"Gestion des données réparties : problèmes et protocoles"*. CEA-EDF-INRIA, Ecole d'été d'informatique, Eyrolles CDEREF n°. 82.

[ROS 82] Rosenschein, J.S. *"Synchronization of Multi-Agent Plans"*. In Proceedings of AAAI'82, pp. 115-119 ; 1982.

[SIC 93] Sichman, J., Conte, R., Demazeau, Y., and Castelfranchi, C. *"Reasoning about Others Using Dependence Networks"*. In Proceedings of the 3rd Italian Workshop on Distributed Artificial Intelligence, AI*IA, Rome, Italy ; 1993.

[SMI 80] Smith, R.G. *"The Contract Net Protocol: High-Level Communication and Control in Distributed Prolem Solver"*. IEEE Trans. on computers, 29 (12) pp. 1104-1113 ; 1980.

[VAL 91] Valmari, A. *"Stubborn Sets for Reduced State Space Generation"*. In LNCS 483, Springer-Varlag ; 1991.