Algèbre relationnel comme outil d'analyse et de synthèse pour les systèmes multiagents

B. Chaib-draa

{chaib}@ift.ulaval.ca
Département d'informatique, Université Laval
Ste-Foy, Québec, Canada, G1K 7P4

Résumé Dans le cadre des systèmes multiagents (SMAs), nous utilisons présentement différents outils formels d'analyse et de synthèse. Ces outils formels sont basés pour la plupart, sur la logique, la prise de décision et les mécanismes de marché. À notre avis, ces outils doivent être complétés par le calcul relationnel si on veut représenter et raisonner sur les "structures sociales" pour lesquelles, les *relations* entre agents sont incontournables. La principale contribution de ce papier réside dans l'introduction d'une approche formelle basée sur le calcul relationnel en vue de raisonner sur les relations aussi bien "entières" que "floues". Les éléments importants que couvre cet article sont pour l'essentiel : (1) les notions de base du calcul relationnel ; (2) l'utilisations des relations entières dans les SMAs et, (4) l'utilisation des relations floues dans les SMAs.

1 Introduction

Un état de l'art portant sur les concepts, présentement utilisés pour les SMAs, montrent clairement que les états mentaux des agents comme les croyances, les désires, les intentions, etc. sont, sans aucun doute, des éléments clés pour toute forme de délibération quelle soit au niveau de chacun des agents, ou au niveau du groupe d'agents considérés comme un tout [8,23,30]. Un tel raisonnement délibératif permet à chaque agent de raisonner sur les autres et de prendre le moment venu, la bonne décision qui pourrait contribuer à maintenir une coordination globale du groupe d'agents. Jusqu'ici, on a plutôt privilégié les logiques classiques et non-classiques en vue de représenter et raisonner sur les SMAs. L'utilisation de ces logiques offrent plusieurs avantages : (1) un langage bien défini qui permet d'identifier les classes d'objets acceptables et qui "dicte" *ce qui* peut être exprimé d'une manière rigoureuse ; (2) une sémantique rigoureuse qui assigne à chaque objet une signification formelle et ; (3) un système de preuves "bien défini" qui permet de nouvelles inférences. L'utilisation de telles logiques permet également aux concepteurs de SMAs de spécifier et de vérifier leurs systèmes à base d'agents. Les directions que prennent les recherches dans ce domaine incluent : le raisonnement sur (a) les attitudes informationnelles (comme les connaissances et les croyances) [15] ; (b) les attitudes motivationnelles (comme les buts, les plans, les intentions, etc.) [7] et, (c) les aspects sociaux [16,29,34].

En plus des outils logiques, la communauté SMA utilise également la théorie de décision, les mécanismes de marchés et plus généralement des théories issues des sciences économiques [25]. De tels outils sont généralement utilisés pour concevoir des protocoles de négociation afin de résoudre, entre autres, des problèmes d'allocation de tâches et de ressources. Ils permettent aux agents d'atteindre des solutions de "bonne qualité" (au sens de *satisficing solution* de Simon) et ce, tout en évitant des interactions coûteuses qui pourraient mener à la désorganisation des agents, voire au chaos.

Si on veut cependant, représenter et raisonner sur les "structures sociales", dans lesquelles, les *relations* entre agents constituent la pierre angulaire, il nous faut compléter la panoplie des outils formels existants, par le calcul relationnel, qui soit dis en passant, est l'outil "naturel" pour rendre compte des relations entre agents. Pour cela, il nous faudrait développer de nouveaux développements mathématiques autour de l'algèbre relationnel afin de rendre compte des *relations binaires* susceptibles d'exister : (i) entre agents (e.g., les engagements entre agents) ; (ii) entre agents et tâches (e.g., les distributions de tâches) ; entre agents et ressources (e.g. distribution des ressources) ; etc.. Dans ce contexte, on considère tout d'abord, une relation binaire R, entre 2 ensembles X et Y, comme étant un prédicat avec 2 slots vides. Ensuite, lorsqu'un élément x de X est mis dans le premier slot, et un élément y de Y dans le second, une assertion (pouvant être vraie ou fausse) va suivre. Par exemple, "possède la ressource" est une relation entre l,ensemble X des agents et l'ensemble Y des ressources individuelles. Cette relation peut alors être vraie ou fausse suivant que l'agent visé "a" ou "n'a pas" la ressource.

Dans le calcul relationnel, quand l'assertion relative à la paire (x, y) est vraie, on écrira :

$$xRy \quad \text{or} \quad (x, y) \in R$$

Par contre, quand l'assertion est jugée fausse, on écrira :

$$x\overline{R}y \quad \text{or} \quad (x, y) \notin R$$

On peut les lire respectivement comme : "x est en relation R avec y" et "x n'est pas en relation R avec y". Dans ces conditions, "l'agent x est engagé envers l'agent y en vue de réaliser le but g" pourrait être représenté par :

$$xR_gy$$

Telle qu'elle est introduite, la notation R_g signifie "est engagé pour réaliser le but g envers ...". À noter que de telles engagements se produisent souvent dans les SMAs et servent généralement à "cimenter" le groupe ou l'organisation d'agents, contribuant ainsi à la coordination et à la cohérence entre de tels agents [31].

Dans la section ci-après, nous introduisons le lecteur au calcul relationnel, pour que dans la section 3, nous expliquons comment appliquer un tel calcul aux SMAs.

2 Une vue rapide sur le calcul des relations binaires

L'origine du traitement mathématique des relations est supposé remontée à Aristote. Ses débuts modernes toutefois, remontent aux travaux initiés par de Morgan [13] et Peirce [21]. Les travaux de ces deux pionniers ont été poursuivis de manière systématique par Schröder [28] qui publia autour des années 1890, ses trois tomes sur *Algebra der Logik*. Une cinquantaine d'années plus tard, Tarski [32] et McKinsey [20], entre autres, ont lancé les fondations du calcul moderne des relations.

Intuitivement, une relation est utilisée pour indiquer un aspect qui *connecte* deux ou plusieurs choses. La relation "est père de ..." est définie sur un ensemble de personnes ; la relation "moins que" pourrait être définie sur l'ensemble N des entiers naturels, etc. Généralement, une relation est vraie ou fausse, dans le sens où elle "connecte" les deux éléments concernés ou pas. Ainsi par exemple, 5 est plus petit que 7, mais 5 n'est pas plus petit que 3. Dès lors, toute relation est sensée partitionner l'ensemble des éléments de l'ensemble concerné, en un sous-ensemble où cette relation est vraie et, un sous-ensemble où cette relation n'est pas vraie. Par exemple, la relation "est suivie par" (telle tâche est suivie par une autre) sur l'ensemble des tâches T d'un SMA, partage cet ensemble en $S \subset T \times T$, l'ensemble des tâches qui répondent à cette relation, et en $S' \subset T \times T$, l'ensemble des tâches restantes. Évidemment, $S \cup S' \subseteq T \times T$.

Les relations sur des ensembles finis peuvent être aussi représentées par des tables ou par des matrices, comme le montre la table ci-dessus ainsi que la matrice Booléenne sur l'exemple relatif à la relation "congruo modulo 3" sur l'ensemble $\{1, 2, ..., 7\} \subset N$.

$$\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 4 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 5 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 6 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 7 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} \qquad \begin{array}{c|l} 1 & \{1,4,7\} \\ 2 & \{2,5\} \\ 3 & \{3,6\} \\ 4 & \{1,4,7\} \\ 5 & \{2,5\} \\ 6 & \{3,6\} \\ 7 & \{1,4,7\} \end{array}$$

La relation ci-dessus est juste donnée à titre d'exemple et on aurait pu donner comme exemple la relation "doit s'exécuter au moins 5mn avant" pour un ensemble de tâches T d'un SMA, ou tout autre exemple relatif à ces systèmes. Bien entendu, relations et graphes sont intimement liés. Usuellement, pour représenter un graphe on représente une relation B sur un ensemble F comme suit : les éléments de F sont connectés par une ligne orientée ayant x comme départ et y comme arrivée, quand précisément $(x, y) \in B$.

Pour introduire le formalisme relationnel, nous commençons par utiliser les symboles $(\vee, \wedge)$, $(\cup, \cap)$, et $(\sqcup, \sqcap)$ pour les opérations Booléennes binaires sur respectivement les valeurs de vérité, les sous-ensembles et les relations. Les symboles

$\wedge$ et $\vee$ reflètent, quant à eux, la *conjonction* et la *disjonction* entre valeurs de vérité. $\cup$ and $\cap$ reflètent l'*union* et l'*intersection* de 2 sous-ensembles. Finalement, $\sqcup$ et $\sqcap$ reflètent l'*union* et l'*intersection* de relations.

D'autres symboles utilisés dans ce texte sont : $\Longrightarrow$ pour la conséquence ; $\Longleftrightarrow$ pour l'équivalence au niveau métalangage ; $:\Longleftrightarrow$ pour la définition au niveau du métalangage ; $:=$ égalité (définition) ; $\rightarrow$ "si–alors" habituelle de la logique propositionnelle.

D'une façon informelle, une *relation binaire* est simplement une collection de paires ordonnées. Précisément, une relation binaire R d'un ensemble X vers un ensemble Y est un sous-ensemble de l'ensemble de toutes les paires (x, y) où $x \in X$ et $y \in Y$. Symboliquement, on peut écrire :

$$R \subseteq X \times Y = \{(x, y) : x \in X \ \wedge \ y \in Y\}.$$

Si jamais $X = Y$, nous dirons alors que R est une *relation homogène* sur X.

Il convient de noter que les notations $(x, y) \in R$, xRy, and Rxy sont équivalents. Comme les relations sont des ensembles et par conséquent, elles sont ordonnées sous l'inclusion. Précisons également que la relation la plus petite est la relation vide $\emptyset$ et la relation la plus grande est la *relation universelle*. La relation vide $\emptyset \subset F \times F$ est notée par O (relation zéro), et la "relation universelle" est le produit cartésien entier $F \times F$, notée L. Finalement, une dernière relation spéciale est la *relation identité* et elle est définie sur chaque X par

$$I = \{(x, x) : x \in X\}.$$

2.1 Opérations sur les relations

Comme les relations sont des ensembles, toutes les opérations sur les ensembles s'appliquent aux relations. Cependant, les structures particulières des relations permettent d'obtenir d'autres opérations, dont les plus usuelles sont :

(1) *post-ensemble (afterset) :* $xR := \{y : (x, y) \in R\}$,
(2) *pré-ensemble (foreset) :* $Ry := \{x : (x, y) \in R\}$,
(3) *union :* $R \sqcup S := \{(x, y) : (x, y) \in R \vee (x, y) \in S\}$,
(4) *intersection :* $R \sqcap S := \{(x, y) : (x, y) \in R \wedge (x, y) \in S\}$,
(5) *transposition :* $R^{\top} := \{(x, y) : (y, x) \in R\}$,
(6) *inclusion :* $R \subset S :\Longleftrightarrow \forall x, y : [(x, y) \in R \rightarrow (x, y) \in S]$,
(7) *complément :* $\overline{R} := \{(x, y) : (x, y) \notin R\}$,
(8) *composition (ou produit $\circ$) :* $R \circ S := \{(x, y) : \exists z (x, z) \in R \wedge (z, y) \in S\}$,
(9) *le sous produit :* $xR \lhd Sz := xR \subseteq Sz$,
(10) *le super produit :* $xR \rhd Sz := xR \supseteq Sz$,
(11) *le produit $\star$:* $xR \star Sz := xR = Sz$,
(12) *produit de Peirce :* $R : X := \{y : \exists x[(y, x) \in R \wedge x \in X]\}$,
(13) *exposant :* $R^0 := I$, et $R^{n+1} := R \circ R^n$.

2.2 Propriétés des relations

Une relation est dite :

(14) *réflexive* si $I \subseteq R$, c'est à dire (c-à-d), $\forall x \in X \ (x,x) \in R$,
(15) *transitive* si $R \circ R \subseteq R$, c-à-d, $(x,y) \in R \wedge (y,z) \in R \rightarrow (x,z) \in R$,
(16) *symétrique* si $R = R^{\top}$, c-à-d, $(x,y) \in R \rightarrow (y,x) \in R$,
(17) *antisymétrique* if $R \sqcap R^{\top} = I$, c-à-d, $(x,y) \in R \wedge (y,x) \in R \rightarrow x = y$,
(18) *connexe* si $\overline{R} \subset R^{\top}$, c-à-d, $\forall x, y \ (x,y) \in R \vee (y,x) \in R$,
(19) *semiconnexe* si $\overline{R} \subset R^{\top} \sqcup I$, c-à-d, $\forall x, y : [x \neq y \rightarrow \{(x,y) \in R \vee (y,x) \in R\}]$,
(20) *pré-ordre ou quasi-ordre* si elle est réflexive et transitive,
(21) *d'équivalence* si elle forme un pré-ordre symétrique,
(22) *ordre partiel* si elle forme un pré-ordre antisymétrique.

Comme nous l'avons déjà mentionné une relation binaire R sur un ensemble X est dite d'*ordre partiel* si elle réflexive, transitive et antisymétrique. Dans ce cas, la paire (X,Y) forme un ensemble ordonné partiellement (i.e., *poset*). Usuellement, R est remplacée avec un symbole plus suggestive, soit '$\leq$' et dans ce cas $R^{\top}$ s'écrit $\geq$. Dans le cas où X a seulement quelques éléments, il est alors plus commode de représenter $(X, \leq)$ par un *diagramme de Hasse* où x est connecté par une ligne droite à y sur le haut du diagramme ssi $x \leq y$ et il n'y a pas de $z \in X$ distinct de x et de y avec $x \leq z \leq y$. Dans ce contexte, retrouver une relation d'ordre à partir de ce diagramme peut être vue comme une opération de fermeture. Dans ce contexte, deux opérations de fermetures sont définies : (1) la fermeture transitive et, (2) la fermeture réflexive transitive.

La *fermeture transitive* est définie directement par :

$$R^{+} := \bigsqcup_{i=1}^{\infty} R^{i} := R \sqcup R^{2} \sqcup R^{3} \sqcup \cdots \tag{1}$$

La *fermeture réflexive et transitive* de R est, quant à elle, définie comme : $R^{*} := \bigsqcup_{i=0}^{\infty} R^{i} := R^{0} \sqcup R^{+}$.

Si différentes opérations apparaissent dans la même expression, des parenthèses sont alors nécessaires pour indiquer l'ordre des opérations qu'il convient d'exécuter. Toutefois, pour éviter la prolifération de parenthèses, les conventions suivantes sont généralement adoptées.

Priorité des opérations : Les opérations unaires du genre $({}^{\top}, {}^{+}, {}^{*}, {}^{n})$ sont exécutées en premier lieu, elles sont suivies par $^{-}$. Viennent ensuite, les relations binaires $(\circ, \lhd, \rhd)$ et finalement, les opérations ensemblistes binaires $(\sqcup, \sqcap)$.

2.3 Propriétés algébriques des opérations relationnelles

Les propriétés les plus usuelles sont :

(23) *composition est associative :* $(R \circ S) \circ T = R \circ (S \circ T)$,

(24) *I est une identité pour les relations sur* X : $I \circ R = R = R \circ I$,
(25) *composition distribue sur l'union :* $R \circ (S \sqcup T) = R \circ S \sqcup R \circ T$,
(26) *transposition distribue sur l'union :* $(R \sqcup S)^\top = R^\top \sqcup S^\top$,
(27) *transposition est une involution :* $(R^\top)^\top = R$ et $(R \circ S)^\top = S^\top \circ R^\top$,
(28) *Product de Peirce est associatif :* $(R : S) : X = R : (S : X)$,
(29) *Produit de Peirce distribue sur* $\cup$: $R : (X \cup Y) = R : X \cup R : Y$,
(30) $\cup$ *distribue sur le produit de Peirce :* $(R \sqcup S) : X = R : X \cup S : X$,
(31) *I est une identité pour le produit de Peirce :* $I : X = X$.

En fait, il y a plusieurs autres propriétés (see [27]), mais celles qui sont fournies ici, forment la base pour un traitement formel des relations.

2.4 Algèbre Booléenne et algèbre relationnel

C'est à Tarski [32] que revient le mérite d'introduire la notion d'*algèbre relationnel*, dans une tentative de faire, pour le calcul des relations binaires, ce que l'algèbre Booléen fait pour le calcul des ensembles. Autrement dit, Tarski espérait capturer le raisonnement du "premier ordre" concernant les relations binaires comme conséquence d'un ensemble d'axiomes équationnels. Pour se fixer les idées, il convient de voir les relations binaires $R, S, T, \ldots$ comme des ensembles et par conséquent, ces ensembles forment une algèbre Booléenne sous l'union, l'intersection et le complément. De plus, les relations sont dotées d'opérations relationnel comme la composition, la transposition, l'identité, etc. Toutes ces relations obéissent aux lois équationnelles, précédemment introduites (23)–(31) et à bien d'autres. C'est pourquoi, Tarski a introduit une *algèbre relationnelle* comme étant une algèbre Booléenne dotée des opérations '$\circ$', '$^\top$', identité et qui obéissent aux axiomes comme (23)–(31) et bien d'autres.

3 Algèbre relationnel comme outil formel pour les systèmes multiagents

3.1 Spécifications utilisant l'algèbre relationnel

Différent auteurs ont caractérisé un système multi-agent comme une forme de tâche distribuée [17,18]. Cette tâche distribuée est généralement représentée comme un classique espace de recherche et/ou. Un tel espace peut être rehaussé par des ressources nécessaires à la résolution des tâches primitives. Dans ce contexte, des relations peuvent exister entre agents, agents et tâches, tâches et ressources, etc.. Ainsi, si on dénote par X, l'ensemble des agents, Y l'ensemble des tâches et Z l'ensemble des ressources, on pourra alors spécifier :

- des relations agents-agents lorsque les relations sont du type $R \subseteq X \times X$;
- des relations agents-tâches lorsque les relations sont du type $R \subseteq X \times Y$;
- des relations tâches-tâches lorsque les relations sont du type $R \subseteq Y \times Y$;
- des relations tâches-ressources lorsque les relations sont du type $R \subseteq Y \times Z$;
- des relations ressources-ressources lorsque les relations sont du type $R \subseteq Z \times Z$.

Relations agents-agents Les relations ici sont de la forme $R \subseteq X \times X$ et elles peuvent représenter *Est-coéquipier-avec, Coopérer-avec, Est-supérieur-hiérarchique-de, Ontologiquement-voisin-de, Est-engagé-envers, etc.* ou toute autre relation entre agents.

Les 4 premières relations sont des relations qui sont réflexives, symétriques, et transitives. Elles forment donc des relations d'équivalences. Rappelons qu'une relation est dite relation de pré-ordre si elle est réflexive et transitive et, une relation d'équivalence est une relation de pré-ordre et symétrique (voir (20) et (21) ci-dessus). Une relation d'équivalence R sur l'ensemble des agents X va donc partitionner cet ensemble en classes d'équivalences. Dès lors la "class d'équivalence" de l'agent x relativement à R est l'ensemble $|x|_R := \{y \in A | (x, y) \in R\}$ consistant en tous les éléments de X qui sont reliés à x par R.

Considérons par exemple l'ensemble X des agents $\{A_1, A_2, \ldots, A_7\}$ reliés par la relation R_g, (une relation pouvant signifier par exemple "coopérer avec quelqu'un, en vue de réaliser le but g), suivante :

$$R_g = \begin{array}{c} \\ A_1 \\ A_4 \\ A_7 \\ A_2 \\ A_5 \\ A_3 \\ A_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} A_1 & A_4 & A_7 & A_2 & A_5 & A_3 & A_6 \end{array} \\ \left(\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array} \right) \end{array} \qquad \begin{array}{rl} C_1 & \{A_1, A_4, A_7\} \\ C_2 & \{A_2, A_5\} \\ C_3 & \{A_3, A_6\} \\ C_4 & \{A_1, A_4, A_7\} \\ C_5 & \{A_2, A_5\} \\ C_6 & \{A_3, A_6\} \\ C_7 & \{A_1, A_4, A_7\} \end{array}$$

Comme on le voit ci-haut à droite, nous avons les classes d'équivalence des agents coopérants entre eux. Bien entendu, les 7 classes se réduisent à 3 classes d'équivalence distinctes, soient les classes C_1, C_2 et C_3. Dans un SMA, ces 3 classes distinctes, permettent de savoir qui coopère avec qui, en vue de réaliser le but g ? et quels sont les sous-goupes ou équipes qu'on peut former pour réaliser ce but ? De telles connaissances peuvent être cruciales quand viendra le moment de se coordonner, de communiquer, de partager les ressource, de résoudre les conflits, etc..

La relation *Est-supérieur-hiérarchique-de* est une relation de préordre et par conséquent elle pourrait permettre aux agents d'inférer de nouveaux liens hiérarchiques en utilisant la transitivité de ce préordre. Le fait de connaître les liens hiérarchiques permet dès lors aux agents de savoir à qui s'adresser dans une organisation hiérarchique, à qui il faudra communiquer des informations ou des résultats, etc.. Il convient de garder à l'esprit, que les liens de communication suivent généralement les liens hiérarchiques.

Finalement, la relation *Est-engagé-envers* est une relation un peu spéciale, dans la mesure où elle peut être symétrique ou antisymétrique dépendemment du problème adressé. En tout cas, elle est généralement non transitive. Cependant, on pourrait se poser la question de savoir comment à partir de $(A_1, A_2) \in E_g$ et $(A_2, A_3) \in E_g$ on pourrait arriver à $(A_1, A_3) \in E_g$, sachant que la relation E_g représente ici "être engagé envers quelqu'un pour le but g". Autrement dit,

comment un agent pourrait se retirer d'un lien d'engagement au profit d'un autre. Intuitivement, on voudrait reprendre ici l'idée suivante : "André doit x à Jean et ce dernier doit également x à Paul. Dans ce cas, les trois peuvent s'entendre pour qu'André soit redevable de x à Paul". Ce qui rend Paul libre de tout engagement.

Relations agents-tâches Les relations sont cette fois-ci du type $R \subseteq X \times Y$ et elles expriment *Peut-coopérer-sur, Doit-faire, Peut-faire, Peut-planifier*, ou n'importe qu'elle autre relation entre agents et tâches. Notons ces relations respectivement par R_{pc}, R_{pf}, R_{df} et R_{pp}. On peut aussi supposer que si les agents peuvent faire une tâche donnée (sous entendu, en entier), ils acceptent aussi de coopérer dessus. Soit donc, $R_{pf} \subseteq R_{pc}$.

Pour illustrer les relations agents-tâches, considérons par exemple l'ensemble des agents $\{A_1, A_2,$
$A_3, A_4, A_5\}$ et l'ensemble des tâches $\{T_1, T_2, T_3\}$. Ces agents communiquent entre eux et suite à cela, ils arrivent à la relation R_{pc}, présentée ci-dessous à gauche. À partir de cette relation, on peut alors remonter à une relation du type agent-agent du genre qui peut coopérer avec qui. Nous avons noté cette relation R_{caa} pour "coopération agent-agent" et nous l'avons représentée ci dessous à droite :

$$R_{pc} = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \end{array} \begin{array}{c} \begin{array}{ccc} T_1 & T_2 & T_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \end{array} \qquad R_{caa} = \begin{array}{c} \\ A_2 \\ A_4 \\ A_5 \\ A_1 \\ A_3 \end{array} \begin{array}{c} \begin{array}{ccccc} A_2 & A_4 & A_5 & A_1 & A_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Avec la relation R_{caa}, on pourrait en déduire des classes d'équivalences comme nous l'avons indiqué dans la section précédente. Trois classes peuvent être dégagées ici, une première classe regroupant (A_1, A_4, A_5) sur la tâche T_1, une deuxième classe regroupant cette fois-ci, les agents (A_2, A_4, A_5) sur la tâche T_3 et enfin, la classe d'agents (A_3, A_5) sur la tâche T_2. Maintenant que nos agents ont pu déterminer qui peut faire quoi, et avec qui il peut le faire, ils peuvent être tenté de négocier un partage de tâche conformément à la charge de travail exigée par chacune des dites tâches. Cependant, pour que cette négociation soit efficace, il faudra se pencher sur les relations tâches-ressources et les relations tâches-tâches.

Relations tâches-tâches Les relations sont cette fois-ci du type $T \subseteq Y \times Y$ et elles dénotent des relations tâches-tâches du genre *Est-similaire-à, Est-concourante-à, Permet, Empêche, Facilite, etc.* ou n'importe qu'elle autre relation entre tâches. Reprenons l'exemple précédent à propos des agents
$\{A_1, A_2, A_3, A_4, A_5\}$ et des tâches $\{T_1, T_2, T_3\}$ et supposons que les relations entre tâches doivent être régies par les relations suivantes "T_3 concourante à T_2 et T_2 facilite T_1". Contraintes qu'on peut écrire formellement de la manière suivante : $(T_3, T_2) \in R_{||}$ et $(T_2, T_1) \in R_f$ avec $R_{||}$ "est concurrent à" et R_f " facilite".

Rappelons que les classes d'équivalence nous imposaient également l'allocation : (A_1, A_4, A_5) pour T_1, (A_2, A_4, A_5) pour T_3 et enfin, (A_3, A_5) pour T_2. Avec les contraintes tâches-tâches qu'on vient d'établir et celles imposées par les relations agents-tâches, nous voyons qu'il y a un problème dans la mesure où A_5 est pris dans les tâches qui doivent se dérouler de manière concourante et de ce fait, il devrait opter pour l'une ou l'autre de ces tâches. Pour cela, il devrait entamer des pourparlers avec les autres agents, pour savoir (1) si (X_5, X_4) peuvent faire sans lui la tâche T_3 et, (2) si X_3 peut faire tout seul T_2. Intuitivement, les agents pourraient ici raisonner sur des relations du genre *Est-similaire-à, Empêche, etc.* pour déterminer ce genre de capacités. Ainsi par exemple, si (X_5, X_4) trouvent qu'ils ont la capacité d'exécuter une tâche T_x similaire à T_3, ils peuvent inférer (toutes chose égales par ailleurs) qu'ils peuvent se débrouiller tous seul pour T_3 et dans ce cas, A_5 pourrait être tenté, par exemple, d'aider X_3 pour la tâche T_3.

Relations tâches-ressources, agents-ressources et ressources-ressources

Tâches-ressources : Les relations sont cette fois-ci, du type $S \subseteq Y \times Z$ et elles expriment des relations liant les tâches aux ressources comme par exemple : *Requiert, Produit, Libère, etc.*, ou n'importe qu'elle autre relation que le concepteur jugera importante. Dans le cas précédent par exemple, nos agents (X_5, X_4) peuvent être intéressés par préalablement circonscrire les ressources que "requiert" T_3 et partant de là, de construire des relations agents-ressources qui spécifient quel agent détient quelle ressource. Partant de là, ils pourraient être tenté de négocier avec les agents détenteurs des ressources, la meilleure manière de se partager ces ressources.

Agents-ressources : Avec de telles relations, on spécifie des liens entre les agents et les ressources comme par exemple *Détenir, Produire* (en exécutant une tâche), *Posséder, etc.*. Comme on l'a précédemment signalé, ces relations permettent aux agents de savoir "qui à quoi" comme ressource. De telles connaissances permettent aux agents de se coordonner sur aussi bien l'allocation des tâches comme on l'a vu jusqu'ici, que sur les tâches en cours, et qui pourraient nécessiter des ressources supplémentaires ou des ressources de remplacement.

Ressources-ressources : Finalement, il est important aussi de caractériser les relations du type "resource-resource" comme par exemple *Nécessite, Peut-remplacée, Va-avec, etc.*. Ces relations permettraient aux agents d'utiliser de manière efficace les ressources en les substituant, accompagnant, etc. par d'autres ressources. Bien entendu, les exemples qu'on donne ici ne sont que des exemples illustratifs et le lecteur intéressé n'aura aucune difficulté à trouver d'autres exemples.

Il convient de préciser ici que la triade agent-tâche-resource constitue la charpente de tout "bon" comportement d'un système multiagent, dans la mesure où la coordination est généralement vue comme l'acte de gérer les interdépendances entre agents, activités et ressources [19]. Dès lors, un raisonnement d'agents basé sur les relations qu'on vient de voir pourrait permettre à ces agents de se coordonner de manière autonome.

3.2 Opérations sur les relations dans les SMAs

Si on considère une relation R telle que $R \subseteq X \times Y$, on pourrait alors considérer xR et Ry pour tous les xRy. En fait, xR et Ry sont des sous-ensemble de Y et de X respectivement. xR consiste en ces éléments de Y reliés à x via la relation R. De manière similaire, Ry consiste en ces éléments de X reliés à y par la relation R. Ainsi par exemple, si R représente la relation *Possède-capacité-de-faire* liant un ensemble d'agents X à un ensemble de tâches Y, alors xR représente l'ensemble des tâches que l'agent x a la capacité de faire, et Ry représente l'ensemble des agents pouvant exécuter la tâche y. De manière similaire, on pourrait obtenir l'ensemble des agents engagés à réaliser avec l'agent x, un but donné g ; l'ensemble des agents qui peuvent coopérer pour une tâche donnée y ; etc.

Bien entendu, l'union $R \sqcup R'$ et l'intersection $R \sqcap R'$ sont très utiles pour représenter l'union et l'intersection, c'est à dire, le "ou" et le "et" de la logique propositionnelle. Par exemple, si R est la relation *Est-capable-de-faire* et R' est la relation *Est-capable de-planifier* alors $R \sqcup R'$ dénote une nouvelle relation stipulant "est-capable de faire ou de planifier", alors que $R \sqcap R'$ exprime "est-capable de faire et de planifier". Dès lors un agent qui sait que l'agent i est lié à la tâche T par $(i, T) \in R \sqcap R'$ pourrait alors inférer $(i, T) \in R'$ et $(i, T) \in R'$, chose qu'il ne peut faire avec $(i, T) \in R \sqcup R'$, bien entendu.

La transposition $R^{\top}$ précises l'inverse de la relation R. Ainsi, par exemple si R exprime *Est-capable-de-planifier* entre X, ensemble d'agents, et Y, ensemble de tâches, alors $R^{\top}$ exprime une nouvelle relation entre Y et X et elle précise cette fois-ci, la relation *Peut-être-planifiée-par*. Il convient de noter ici que si jamais R est symétrique alors $R = R^{\top}$. C'est le cas par exemple, de la relation *Est-similaire* qui pourrait liée 2 tâches ou 2 ressources. Dans ce cas, dire que la tâche T_1 est similaire à T_2 revient aussi à dire que T_2 est similaire à T_1.

L'inclusion $R \subset R'$ peut être utilisée pour spécifier $\forall x, y : [(x, y) \in R \rightarrow (x, y) \in R']$. En fait, toutes les expressions indiquant le "si ... alors" en logique propositionnelle (ou les propositions sont des relations) peuvent être représentées par l'inclusion.

Le complément d'une relation, noté habituellement $\overline{R}$ peut représenté dans le cadre des SMAs des informations du genre *Ne-requiert-pas Non-similaire*, etc.

Nous allons maintenant revoir quelques applications du produit dans le cadre des systèmes multiagents. Comme nous l'avons vu plus haut, il existe différents types de produits, chacun d'eux pouvant être utilisé de manière distincte dans le cadre des SMAs. Dans ce papier, nous avons adopté les produits relationnels à la Bandler [2] dans les mesure où ces produits s'y prêtent bien aux aspects multiagents. Ces produits ont été précédemment introduits (voir (9), (10), (11)).

Avant d'aller plus loin au niveau de ces produits, il convient de rappeler que toute relation entre 2 ensembles X et Y peut s'écrire sous la forme d'une matrice M_R. Les lignes de cette matrice sont les éléments x_i de X, et les colonnes sont les éléments y_i de Y. Dans ce cas, l'entrée d'une cellule (i, j) où i représente la ligne et j la colonne, est l'assertion $x_i R y_i$. Cette entrée est appelée R_{ij} et elle est booléenne. En fait, quand on multiplie des relations (par exemple R et

S) il est plus commode d'utiliser leur représentation matricielle parce que le résultat peut être obtenu en ré-écrivant les formules pour la multiplication des matrices de manière booléenne. Le produit $\circ$ par exemple, se construit élément par élément de la manière suivante :

$$(R \circ S)_{ik} = \vee_j \{R_{ij} \wedge S_{jk}\}.$$

Pour les autres produits, il est important de considérer les relations entre ensembles. Si, on note respectivement par $(\rightarrow)$ et $(\leftarrow)$ les opérations Booléennes d'implication et "est impliqué par", alors on pourra écrire :

(32) $(R \circ S)_{ik} = \vee_j \{R_{ij} \wedge S_{jk}\}$,
(33) $(R \lhd S)_{ik} = \vee_j \{R_{ij} \rightarrow S_{jk}\}$,
(34) $(R \rhd S)_{ik} = \vee_j \{R_{ij} \leftarrow S_{jk}\}$,
(35) $(R \star S)_{ik} = \vee_j \{(R_{ij} \rightarrow S_{jk}) \wedge (R_{ij} \leftarrow S_{jk})\}$.

Il y a plusieurs significations aux relations "produit" dans les SMAs. Ainsi, si on considère par exemple X, l'ensemble des agents, Y l'ensemble des ressources, Z l'ensemble des tâches, R et S les relations : *Possède-ressource* et *Est-ressouce-de*. Dans ce cas, nous pourrons avoir :

- $xR \circ Sz$: l'agent x a au moins une ressource de la tâche z.
- $xR \lhd Sz$: les ressources de x sont parmi ceux qui sont nécessaires à la tâche z.
- $xR \rhd Sz$: les ressources de x inclut toutes celles qui sont nécessaires pour la tâche z.
- $xR \star Sz$: les ressources de x sont exactement celles qui sont nécessaires pour la tâche z.

Un autre important exemple est le suivant : soit X et Y des ensembles d'agents et de buts ; et R la relation *A-pour-but* (dans ce cas, la relation $R^\top$ est la relation *Est-but-de*. Dans ce cas, on peut considérer, deux types de produits : $RR^\top$ et $R^\top R$.

1. Pour le produit $RR^\top$
   a) Si, pour x_1, $x_2 \in X$, on a : $x_1 R \lhd R^\top x_2$; on pourra alors interpréter cette expression comme "x_1 partage au moins un but avec l'agent x_2". Dans ce cas, x_1 et x_2 peuvent essayer de négocier comment coopérer pour réaliser les buts qu'ils ont en commun.
   b) Si $x_1 R \rhd R^\top x_2$, ceci pourrait alors exprimer par exemple : "Les buts de x_1 incluent ceux de x_2". Dans cette situation, x_1 pourrait (par exemple) aider x_2 en réalisant seul l'ensemble des buts.
   c) Si $x_1 R \star R^\top x_2$, ceci pourrait alors exprimer le fait que les buts de x_1 sont les buts de x_2. Dans ce cas,
      - x_1 et x_2 pourraient négocier pour éviter la redondance ;
      - x_1 et x_2 pourraient négocier pour partager les ressources.
2. Pour le produit $R^\top R$: le raisonnement est similaire

Le dernier produit que nous considérons ici est le produit de Peirce [1] Ce produit est noté $R : X$, où R est une relation et X un ensemble. Ce produit obéit aux lois (28)–(31). De plus, les règles de dérivation de ce produit peuvent être dérivées des règles usuelles du calcul relationnel. Soit donc, P, Q deux relations et X un ensemble. Alors,

(36) $Q \subset R \Longrightarrow Q : X \subseteq R : X$,
(37) $Q \subset R \Longrightarrow PQ \subset PR \Longrightarrow PQ : X \subseteq PR : X$,
(38) $P \subset Q \Longrightarrow PR \subseteq QR \Longrightarrow PR : X \subseteq QR : X$.

Toutes ces dérivations peuvent être utilisées par les agents pour faire de nouvelles inférences.

Avec le produit de Peirce, nous pouvons représenter des ensembles spécifiques dans les systèmes multiagents. Par exemple, soit X l'ensemble de tous les agents et ces agents sont classées selon trois catégories $X1$, $X2$, et $X3$. De plus, soient R_c, R_T, R_s, les relations qui précisent *Coopère-avec*, *Partage-tâche-T-avec*, *Est-subordonné-de*. Nous avons alors dans ce contexte, les produits de Peirce suivants :

1. $R_c : X1$ représente l'ensemble des agents coopérants avec les agents dont le type est $X1$;
2. $R_T : Y1$ représente l'ensemble des agents partageant T avec les agents dont le type est $Y1$;
3. $R_s : Z1$ représente l'ensemble des agents qui sont les subordonnés des agents dont le type est $Z1$, etc.

3.3 Comment utiliser les propriétés des relations dans un SMA

Les propriétés des relations peuvent être très utiles quand viendra le moment d'analyser ou de synthétiser des notions de SMA. La "similarité" entre tâches et entre ressources par exemple, peut être décrite par de telles propriétés. Formellement, si on dénote par "a est similaire à b" de la manière suivante $R_s(a, b)$, on pourra alors avoir les propriétés suivantes :

- chaque tâche (ou ressource) est similaire à elle-même ;
- si a est similaire à b, alors b est similaire à a ;
- si a est similaire à b, et b à c, alors a est similaire à c.

De telles propriétés peuvent être exprimées en utilisant des formules du genre $R_s(a, a)$ (i.e., R_s est réflexive), $R_s(a, b) \leftrightarrow R_s(a, b)$ (R_s est symétrique), et $(R_s(a, b) \wedge R_s(b, c) \rightarrow R_s(a, c)$ (R_s est transitive). Une relation qui satisfait toutes ces propriétés est appelée une relation d'équivalence. Dans cette optique, la *Classe d'équivalence* de a (avec $a \in A \ : (a, b) \in R$) consiste en tous les éléments reliés à a par la relation R_s ("est similaire à").

Il convient de noter ici que la transitivité amène également, deux autres implications, soient :

[1] Ce produit est crédité à Peirce parce qu'il fut le premier à l'utiliser, dans son important papier de 1870 [22].

(39) $R(a,b) \wedge \overline{R}(a,c) \rightarrow \overline{R}(b,c)$,
(40) $R(b,c) \wedge \overline{R}(a,c) \rightarrow \overline{R}(a,b)$.

Ces deux nouvelles implications se déduisent de l'implication classique "si-alors" de la logique propositionnelle. De telles implications, peuvent être utilisées comme règles d'inférence par les agents, pour raisonner sur les relations dans des environnements multiagents.

3.4 Relations floues et leurs applications aux SMAs

Comme nous l'avons précédemment introduit, xRy se lit comme "x est en relation R avec y". Quand la décision, sur le fait de savoir si un tel énoncé est vrai ou non, peut se faire sur une base "entière", la relation R est dite "entière". Quand en revanche, la décision peut être jugée selon des degrés, la relation R est dite "floue". Nous adoptons ici, la convention suivante : le nombre 0 est assigné à l'assertion xRy "faux" et 1, à l'assertion xRy "vraie". les nombres entre 0 et 1 seront utilisées pour évaluer les degrés d'incertitude intermédiaires. De cette façon, les relations entières requièrent seulement l'ensemble $\{0,1\}$ pour leur valeurs, tandis que les relations floues utilisent, quant à elles, l'intervalle fermé entier suivant $[0,1]$.

Comme nous l'avons précédemment signalé, il est préférable de représenter toute relation R entre 2 ensembles X et Y, sous la forme de matrices M_R où toute entrée dans une cellule, ligne i et colonne j représente l'assertion $x_i R y_i$. Cette entrée est appelée R_{ij} et dans le cas des relations "entières" elle vaut 0 ou 1. Dès lors, la matrice M_R est une matrice qui consiste en des 0 et des 1 pour les relations "entières", et en des valeurs comprises entre $\{0,1\}$ pour les relations floues. Bien entendu, dans l'un ou l'autre de ces cas, le fait de connaître cette matrice permet de connaître la relation R.

Dans le contexte du multiagent, pour n'importe quel élément (agent par exemple) x_i de l'ensemble X, alors $x_i R$ est le sous-ensemble de Y (représentant des tâches par exemple) qui lie les éléments de celui-ci à x_i via R (R pourrait refléter par exemple *Est-impliqué-dans*). Si R est floue, alors $x_i R$ est un sous-ensemble flou de Y. Cet ensemble est complètement décrit par la $i$ème ligne de la matrice M_R.

Des formules précédentes (32)–(35), il convient de préciser que la (32) a déjà été introduite dans la logique floue par Zadeh [35] in 1971. Pour faire la même chose pour les autres, il est alors nécessaire de choisir un *Opérateur d'implication floue*, en remplacement des flèches $\rightarrow$ et $\leftarrow$. Il y a pour cela, différents opérateurs flous que le lecteur intéressé pourrait trouvé dans la littérature [2]). Il suffit peut être de juste signaler ici que les opérateurs suivants ont déjà fait leur preuves au niveau de applications réelles.

(41) Lukasiewiech $a \rightarrow_5 b = min(1, 1 - a + b)$;
(42) Kleene-Dienes Lukasiewicz $a \rightarrow_{5.5} b = min(1, 1 - a + ab)$;
(43) KD Kleene-Dienes $a \rightarrow_6 b = (1 - a) \vee b$.

Dans le cas du produit de Peirce, si R reflète une relation floue, nous avons alors un produit de Peirce floue $R : X$ qui représente des ensembles spécifiques flous. Par exemple, l'ensemble des agents qui sont "fortement engagés" ou l'ensemble des agents qui sont "faiblement engagés" (à réaliser un but), etc.

4 Conclusion

Dans ce papier, nous avons essayé de franchir une première étape vers un modèle formel utilisant les relations pour caractériser une "structure sociale". Précisément, nous avons proposé un formalisme basé sur l'algèbre relationnel qui rend compte des inter-relations entre agents, tâches, ressources, et autres composants d'un système multiagent. Pour réaliser cela, nous avons présenté en détails les notions de base de ce formalisme, et expliqué comment on pourrait les utiliser dans un contexte multiagent.

La conséquence la plus évidente qui milite en faveur de l'utilisation d'un tel formalisme et de permettre à chaque agent de raisonner sur les inter-relations de manière "naturelle". Avec un tel raisonnement, les agents pourraient (i) gérer les inter–dépendances entre leurs activités pour obtenir et maintenir une certaine coordination entre eux, (ii) gérer des information de haut niveau qui leur permettraient de diminuer la charge de communication.

Il est prévu un exemple plus complexe et plus complet d'interactions entre agents pour valider un tel formalisme.

Références

1. R. Axelrod, Ed., *Structure of Decision : The Cognitive Maps of Political Elites*, Princeton University Press, 1976.
2. W. Bandler and L. J. Kohout. A Survey of Fuzzy Relational Products in their Applicability to Medicine and Clinical Psychology. Bandler and Kohout (eds) *Knowledge Representation in Medicine and Clinical Behavioral Science*, 1986.
3. R. Berghammer and H. Zierer. Relational algebraic semantics of deterministic and non-deterministic programs. *Theorem. Comput. Sci. 43*, 1986, 123–147.
4. R. Berghammer, A. Haeberer, G. Schmidt and P. Veloso. Comparing two different approaches to products in abstract relation algebras. Proc. Conf. Algebraic Methodology and Software Technology (AMAST'93), Enschede, The Netherlands, June 1993, Springer-Verlag, 1994, 167–176.
5. C. brink, Boolean modules, *Journal of Algebra* **71**(2), 291–313, 1981.
6. C. Catelfranchi, M. Micel and A. Cesta, Dependences relations among automous agents, in E. Werner, and Y. Demazeau (eds.) *decentralized A. I. 3*, Elsevier Science Pub., 1992.
7. C. Catelfranchi, Guarantees for autonomy in cognitive agent architecture, in *Intelligent Agents : ECAI–94 Work. on Agent Theories, Architectures, and Languages* (ed.) M. J. Wooldridge, N. R. Jennings, Springer Verlag (LNAI–890), 1994.
8. B. Chaib-draa, Formal tools for multi-agent systems. Rapport de recherche, DIUR RR–9507, Département d'Informatique de l'Université Laval, Ste-Foy, Can., 1995.
9. B. Chaib-draa, A Relation graph formalism for relationships between agents. *IEEE Trans. on Knowledge and Data Eng.* (to appear).

10. L.H. Chin and A. Tarski. Distributive and modular laws in the arithmetic of relation algebras. *University of California Publications 1*, 1951, 341–384.
11. K. S. Decker, "Environment centered analysis and design of coordination mechanisms," Ph.D. dissertation also as UMass CMPSCI Tech. Rep. 95-XXX, 1995.
12. J. Desharnais, Abstract relational semantics, Thèse de Ph.D., School of Computer Science, McGill University, Montréal, July, 1989.
13. A. De Morgan, *Formal Logic : The Calculus of Inferences, Necessary and Probable*, Taylor and Walton, London, 1847.
14. E. H. Durfee and V. Lesser, "Partial global planning : A coordination framework for distributed hypothesis formation," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, 1991, pp. 1167-1183.
15. R. Fagin, J. Y. Halpern, Y. Moses, M. Y. Vardi. *Reasoning About Knowledge*, MIT Press, 1995.
16. N. R. Jennings, *Cooperation in Industrial MultiAgent Systems.* World Scientific in Computer Science, vol. 43, 1994.
17. N. R. Jennings, "Coordination techniques for distributed AI," in G. O'hare and N. Jennings, Eds, *Contribution to Foundation of DAI*, Wiley Inter-Science, 1996 (to appear).
18. V. R. Lesser, "A retrospective view of FC/C distributed problem solving," *IEEE Trans. Syst., Man, Cybern.* vol. 21, 1991, pp. 1347-1362.
19. Malone, T.W. and Crowston, K. What is coordination theory and how can it help design cooperative work systems? In D. Tatar (ed), *Proc. of the 3rd Conf on Computer-Supported Cooperative Work,* LA, CA, ACM Press, 1990, 357-370.
20. J. C. C. McKinsey, Postulates for the calculus of binary relations, *Journ. Symbolic Logic*, 5, pp. 85-97, 1940.
21. C. S. Peirce *The Collected Papers of Charles Sanders Peirce*, vol. III, Harshorne and Weiss, Ed., Harvard Univ. Press, Camb. Mass., (1931-1935).
22. C. S. Peirce, Description of a notation for the logic of relatives, resulting from an amplification of the conceptions of Boole's calculus of logic, *Mem. Amer. Acad.*, **9**, 317–378, 1870.
23. A. S. Rao and M. P. georgeff, "Modeling rational agents within a BDI-architecture," *Proc. of Princ. of Knoweledge Representation and Reasoning*, 1991, pp. 437-484.
24. J. S. Rosenschein and G. Zlotkin, *Designing Conventions for Automated Negotiation Among Computers*, MIT Press, 1994.
25. T. W. Sandholm, Distributed Rational Decision Making. in G. Weiss (ed.) *Multiagent Systems*, MIT Press, pp : 201–258, 1999.
26. G. Schmidt and T. Ströhlein. Relation algebras : Concept of points and representability. *Discrete Math. 54*, 1, 1985, 83–92.
27. G. Schmidt and T. Ströhlein. *Relations and Graphs.* Springer-Verlag, Berlin, 1993.
28. E. Schröder, *Algebra der Logik.* vol. 3, Teubner, Leipzig, 1895.
29. J. S. Sichman, Y. Demazeau, R. Conte, C. Castelfranchi, A social reasoning mechanism based on dependence networks, *Proc. ECAI–94, 11th European Conf. on AI*, 1994.
30. M. P. Singh, Formal methods in DAI : logic-Based representation and reasoning. in Weiss G. (eds), *Multiagent Systems*, MIT Press, pp 331–376, 1999.
31. M. P. Singh. Social and Psychological Commitments in Multiagent Systems. *AAAI Fall Symposium on Knowledge and Action at Social and Organizational Levels*, 1991.
32. A. Tarski. On the calculus of relations. *J. Symb. Log. 6*, 3, 1941, 73–89.
33. F. von Martial, *Coordinating Plans of Autonomous Agents,* Lectures Notes in AI no 610, Springer Verlag, 1992.

34. M. wooldridge, This is MyWorld : the logic of an agent–oriented DAI testbed, in *Intelligent Agents : ECAI–94 Work. on Agent Theories, Architectures, and Languages* (ed.) M. J. Wooldridge, N. R. Jennings, Springer Verlag (LNAI–890), 1995.
35. L. A. Zadeh, Similarity relations, *Inform. Sci.*, **3**, 177–200, 1971.