Communication et langages de communication dans les SMA

Bernard ESPINASSE
Aix-Marseille Université (AMU)
LSIS UMR CNRS 7296

2012

- **Problématique de la communication entre agents**
- **Eléments théoriques pour la communication par envoi de messages**
- **Actes du langage**
- **Le langage KQML**
- **Le langage ACL – FIPA**
- **Comparaison KQML et ACL-FIPA**
- **Ontologie commune**
- **Langages de conversation**

Plan

Références bibliographiques

Cours :

- **Gleize M.P.**, Cours "Intelligence collective", Université de Toulouse, IRIT.
- **Quinqueton J.**, "Systèmes multi-agents", Université de Montpellier, LIRMM.
- **Esfandiari B.**, "Sofware Agents" Course, University of Carleton, Canada.
- **Finin T., Labrou Y.**, "Tutorial on Agent Communication Languages", University of Maryland Baltimore Country.
- **Florea A. M.**, "Agents et Systèmes Multi-agents", Université de Bucarest, Roumanie.
- …

Articles :

- **Cohen, P. R. and Levesque, H. J.**, 1990b. "Rational interaction as the basis for communications". Intentions in Communication, pp. 221-56, MIT Press.
- **Rao, A. S. and Georgeff, M. P.**, 1991. "Modelling rational agents within a BDI architecture". In Proc. of Knowledge Representation and Reasoning (KR&R-91), pp. 473-484, Morgan Kaufman.
- **Rao, A. S. and Georgeff, M. P.**, 1995. "BDI Agents: From Theory to Practice". In Proc. of ICMAS-95, San Fransisco, June 1995.
- **Woolridge, M. and Jenning, N. R.**, 1995. "Intelligent agents: theory and practice". The Knowledge Engineering Review, 10(2), pp. 115-52.
- …

Références bibliographiques

Livres :

- **Ferber J. (95), Les systèmes multi-agents**, InterEditions.
- **Weiss G. - editor (00), Multiagent Systems**, MIT Press.
- **Singh M. (94), Multiagent Systems,** Springer Verlag.
- **Conte R., Castelfranchi C. (1995), Cognitive and Social Action**, UCL Press.
- **Haddadi A. (95), Communication and Coopération in Agent Systems**, Springer Verlag.
- **Dennett, D. C.**, 1987. "The intentional stance", MIT Press.
- **O'Hare G.M.P. & Jennings N.R. - editors (96), Foundations of Distributed Artificial Intelligence**, Wiley-Interscience.
- **Bradsham M. - editor (97), Software Agents**, AAAI Press - The MIT Press.
- **Huhns M.N. & Singh M.P. - editors (97), Readings in Agents**, Morgan-Kaufmann.
- ...

1. Problématique de la communication entre agents

- **Communication Versus Computation**
- **Communication entre Agents Versus communication entre Objets**
- **L'interopération d'agents**
- **Échanges d'information entre agents : par partage d'information, par envoi de messages**

Problématique de la communication

Pourquoi des agents communiquent ?

- pour **coordonner** leurs activités
- des agents peuvent se **coordonner sans communiquer** que s'ils possèdent des modèles de comportement des autres agents
- pour **coopérer**, **collaborer**, **négocier**, ...

Que communiquer ?

- pour se coordonner, coopérer, négocier les agents ont en général besoin de communiquer leurs **intentions, buts, résultats** et **états**

Modalité de la communication ?

- **expéditeur** et **destinataire**(s) du **message**
- **comment** communiquer - **ressources** et **protocole** utilisés :
  - par **liens de communication** (communication directe): **envoi de messages**
  - par **mémoire partagée** (communication indirecte): **tableau noir** (blackboard), ...
  - par l'intermédiaire de **conventions partagées**, ...

Communication Versus Computation

La communication est généralement plus coûteuse et moins fiable :

- la **re-computation** est souvent **plus rapide** que faire des requêtes d'information sur un canal de communication
- la communication peut **conduire à des négociations prolongées**
- la **mise à jour de chaînes de croyances** et de **buts** résultant de communication peut ne jamais terminer

La communication est qualitativement supérieure :

- l'information **ne peut pas toujours être reconstruite localement**
- la **communication peut être évité** sauf quand les agents sont conçus dés le départ pour partager toute la connaissance partagée (une hypothèse limitative)

Communication entre Agents Versus communication entre Objets

Sémantique de la communication entre objets :

- dépendante du récipient du message (concept de polymorphisme)

Sémantique d'un langage de communication d'agents (ACL) :

- ne doit pas dépendre des agents participants

Communication entre agents plus puissante :

- permet des données complexes, de l'information logique (ex: buts), des commandes, scripts ou même des programmes complets (cas des agents mobiles)

L'interopération d'agents

l'interopérabilité = trait particulier des agents

- possibilité **d'échanger** de **l'information**, de la **connaissance** et des **services** entre plusieurs agents (avec des capacités, des architectures différentes) dans une **compréhension mutuelle**
- **rend possible la communication effective** entre des agents séparés comme le développement de MAS
- est la **base de la coopération**, de la **résolution de problèmes** au travers de d'interaction mutuellement compréhensibles entre un certain nombre d'agents

l'interopérabilité nécessite une communication à différents niveaux :

- de **simples requêtes** avec un simple et concis ensemble de réponses possibles
- à une **communication complexe dans un protocole approprié** par exemple pour une conversation dans une négociation

=> besoin d'un protocole de communication standardisé commun et indépendant de l'application (KQML)

Organisation et interopérabilité d'agents

Différentes formes d'organisation d'agents ayant à inter-opérer sont possibles :

- de **couplage fort maître - esclave**
- à des **organisations spécifiques** où les agents sont complètement indépendants et sans but mutuels (voire buts opposés en négociation ou aux enchères)
- en ayant recours à des agents **facilitateurs** (facilitators), une classe d'agents assumant la charge de l'interopérabilité notamment :
  - localiser les agents par leur nom (**pages blanches** - white pages) ou par leur capacité (**pages jaunes** - yellow pages)
  - **faire suivre les requêtes** aux agents concernés
  - **décomposer la requête en sous-requêtes**, **faire suivre** les différentes sous-requêtes aux différents agents concernés, **collecter** toutes les réponses et les **composer** dans une réponse unique à la requête initiale
  - **traduire l'information entre différents vocabulaires** afin de s'adapter à différents interfaces d'agent
  - **stoker temporairement l'information** (buffer), …

Communication dans une interopération intelligente

La communication nécessite :

- un **protocole de transport** : un mécanisme de transport de données (e.g. TCP/IP, SMTP, HTTP, …) qui doit être opaque pour les agents
- un **langage de communication** : un médium au travers duquel les attitudes relatives au contenu de l'échange sont communiquées
- un **protocole d'interaction** : une stratégie de haut niveau (Ex: une stratégie de négociation) qui gouverne des conventions qui doivent être respectées lors de l'interopérabilité (ou la coopération)

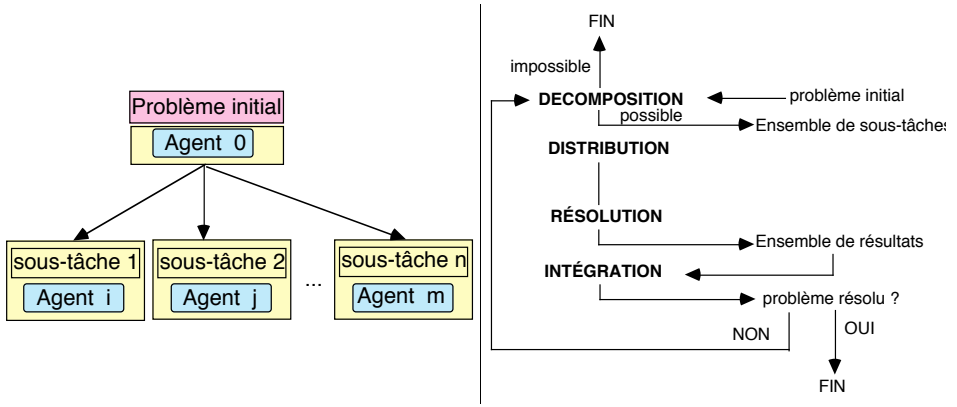
Limitations générales :

- **pas de résolution d'inconsistances** dans l'utilisation de **syntaxe** et de **vocabulaire**
- **pas de communication de connaissance**

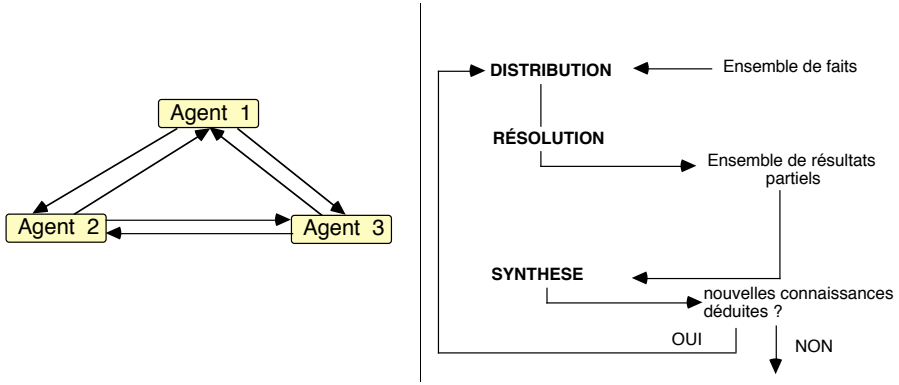
Partage et compréhension mutuelle de connaissance entre agents:

- **traduction d'un langage de représentation des connaissances en un autre langage** (ou d'une famille de langage de représent. des connaissances en une autre)
- **partage du contenu sémantique de la connaissance représentée** : traduction pas suffisante : chaque base de connaissance contient des hypothèses implicites sur la signification de faits représentés et ces abstraction doivent être partagées

Exemple de communication : partage des tâches



Exemple de communication : partage des résultats



correspondance image synthétisée - objet réel

Échanges d'information entre agents

Comment un agent peut informer un autre agent ?

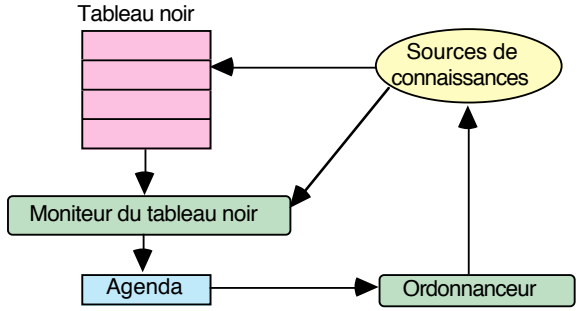
- envoyer l'information dans un message (**envoi de message**)
- écrite l'information à un endroit où l'autre agent pourra la lire (**mémoire partagée : communication par partage d'information - tableau noir/blackboard**)
- montrer ou démontrer à l'autre agent (teaching)
- insérer ou programmer l'information directement dans l'autre agent (maître -> esclave; contrôleur -> contrôlé - "chirurgie du cerveau")

Comment un agent peut obtenir de l'information d'un autre agent ?

- poser une question à l'autre agent (**envoi de message**)
- lire un endroit où les autres agents ont écrit quelque chose (**mémoire partagée : communication par partage d'information - tableau noir/blackboard**)
- observer les autres agents (learning)
- accéder à l'information directement au sein de l'autre agent ("chirurgie du cerveau")

Communication par partage d'information : tableaux noirs [Barbara Hayes-Roth 85]

- **structure de données** : partagée et éventuellement partitionnée
- **modèle du tableau noir (blackboard)** : sources de connaissances + tableau central + composante de contrôle



KS = knowledge source = spécialiste (connaissances homogènes, contrôle connu bien maîtrisé, indépendance)

Communication par tableaux noirs [Barbara Hayes-Roth 85]

Mécanisme de contrôle :

- coordonner de manière efficace les KS
- procédural, hiérarchique, opportuniste ou hybride
- architectures diffèrent par la façon de traiter et d'implémenter le contrôle

KSAR = (module, liste d'hypothèses) et Agenda = liste des KSAR

Cycle :

1. les modules créent KSAR dans l'agenda
2. contrôle choisit un KSAR
3. le KSAR choisi est déclenché et une combinaison d'actions est effectuée sur le tableau

Avantages des tableaux noirs :

- peu de pertes informations
- correction des erreurs très rapide

Inconvénients des tableaux noirs :

- risque d'accumulation de données inutiles
- contraintes de vocabulaire
- dépendant du domaine d'application

Communication par envoi de message

Transmission par diffusion : réseau de contrats (Davis)

Transmission directe :

- modèle d'acteur (Hewitt)
- modèle des accointances

Avantages :

- liberté d'expression
- souplesse de la communication
- parallélisation

Inconvénients :

- pertes d'informations
- correction d'erreurs difficiles
- saturation des communications

Connaissances des agents différentes

=> système indépendant du domaine d'application

2. Eléments théoriques pour la communication par envoi de messages

- **Besoin de langages et de protocoles de communication**
- **Apport de la Théorie des actes du langage**
- **Actes illocutoires**

Besoins liés à un échange de messages

Les agents doivent savoir :

- avec **quels autres agents communiquer**
- **comment établir un canal de communication** avec eux
- **quel protocole utiliser** pour le dialogue
- **quel langage utiliser** pour échanger des **connaissances**
- **quels termes du langage utiliser** pour garantir que l'autre agent va **interpréter les expressions dans le même sens**
- **comment gérer l'information inutile** et les **possibles données erronées** qui apparaissent des différentes vues du monde

=> Besoin de langages et de protocoles pour communiquer

Langage de communication

L'échange de messages nécessite un langage avec :

- **une syntaxe :**

  nécessite un **langage commun** pour représenter information et requêtes

- **une sémantique :**

  nécessite un **vocabulaire structuré** et un **cadre partagé de connaissance** : une **ontologie partagée**

- **une pragmatique** précisant :
  - **avec qui communiquer** et comment le ou les **trouver**
  - **comment initialiser** et **maintenir** un **échange**
  - **l'effet** de la communication sur le **destinataire**

Protocoles de communication

Protocole de communication : Spécifie :

- **l'expéditaire** (sender)
- le ou les **destinataires** (receiver(s))
- le **langage** utilisé dans le protocole
- les **actions** à réaliser par les **participants** au protocole à différentes étapes
- est **défini** : **au dessus d'une couche de transport et au niveau connaissance** (knowledge level) : implique des concepts de haut niveaux comme : croyances, intentions, engagements, permissions, demandes

Apport de la théorie des actes du langage [Austin, Searle, Vandervecken]

- "**Communiquer c'est agir**" : la communication envisagée sous la forme d'une action, qui doit être gérée comme les autres actions
- **catégoriser des types de communication**: informer, demande de faire, demande d'info, réponses, promesse, proposition et offre...
- **type de communication décrit par ses conséquences** : permet un traitement logique de la communication, protocoles de com. associés à chaque acte de langage

Actes du langage [Austin, Searle, Vandervecken]

Composantes d'un acte de langage (actes élémentaires):

- **locutoire**: génération d'**énoncés** (production d'une phrase dans une langue donnée)
- **illocutoire**: **acte** réalisé par le **locuteur** sur le **destinataire** : informer, demande de faire, demande d'info, réponses, promesse, proposition, offre...
  Ex: affirmer("il pleut !") ou questionner("il pleut ?")
- **perlocutoire**: **effets** que les actes illocutoires peuvent avoir sur l'**état du destinataire**
  Ex: persuader = affirmer avec désir que le destinataire partage les croyances du locuteur

Classification des actes du langage:

- **informatifs**: affirmer un fait - Ex : "Achille a 4 ans"
- **directifs :**
  - **exercitifs**: demander de faire quelque chose
  - **interrogatifs**: poser une question
- **promissifs**: engagement à accomplir un acte - Ex : je ferai cours demain
- **expressifs**: exprimer un état - Ex : je suis heureux
- **déclaratifs**: accomplir un acte par l'énonciation - Ex : je t'aime

Acte illocutoire

Acte illocutoire :

- produire un certain effet sur le **destinataire** lors de la formulation d'un **énoncé**
- largement étudiés en **pragmatique du langage**

Acte illocutoire = contenu propositionnel (P) + force illocutoire (F)

F(P) ou <performative> (<contenu>)

- **performative** = type d'acte illocutoire - verbe (Informer, Demander de faire, Questionner, Répondre, Promettre, Affirmer,...)

Exemples :

- **Affirmer (il pleut)**
- **Questionner (il pleut)**

Succès d'un acte illocutoire : acte reconnu par l'auditeur

=> Langage opératoire KQML

3. KQML (Knowledge Query and Manipulation Language)

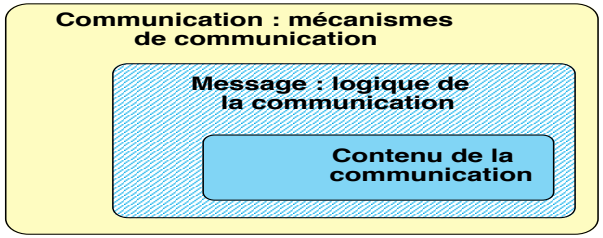
- **Couches, Hypothèses et Sémantique de KQML**
- **Mots clés et performatives de KQML**

KQML (Knowledge Query and Manipulation Language)

- KQML = un **langage** et **protocole** de communication de haut niveau, **orienté message** pour l'**échange d'information** entre agents
- développé par le **DARPA** (Knowledge Sharing Effort) pour **supporter l'interopérabilité** entre les agents intelligents dans une application distribuée

- KQML opérationnalise la **théorie des actes de langage**
- KQML est **indépendant** :
  - du **mécanisme de transport** (e.g., tcp/ip, email, corba objects, IIOP, etc.)
  - du **langage du contenu** (e.g., KIF, SQL, STEP, Prolog, etc.)
  - de **l'ontologie** concernée par le contenu
- KQML offre des **primitives de message types d'intérêt particulier** pouvant être utilisés dans un **dialogue** entre agents
- dans KQML tout agent est vu comme **gérant une base de donnée (BD)**

KQML: un langage à 3 couches



- **couche contenu** : les agents doivent s'accorder sur le **langage de contenu** (KRSL, KIF, LOOM, prolog, Clips, ...) et **l'ontologie** à utiliser
- **couche message**: codage du message qu'un agent souhaite transmettre à un autre:
  - **données**: contient une description d'une partie de la connaissance
  - **déclarations** : type d'information échangée, type d'acte du langage (assertion, requête, réponse, message d'erreur, etc...)
- **couche communication**: échange de "**packages**" - enveloppe autour du message - spécifiant les agents expéditeur et destinataire(s)

Hypothèses et Sémantique de KQML

Hypothèses de communication (Communication Assumptions) :

- les agents sont connectés par des **liens unidirectionnels** transportant des messages discrets
- les liens ont des **délais de transport non nulle**
- un **agent connaît le lien d'un message reçu**
- un **agent contrôle le lien pour un envoi d'un message**
- les **messages** envoyés à un simple destinataire **arrivent** dans l'**ordre** de leurs **envois**
- les **messages délivrés** sont **fiables**

Sémantique KQML (KQML Semantics) [Labrou et Finin] :

- chaque agent gère une **base de connaissance virtuelle** (virtual knowledge base - **VKB**)
- les éléments de la VKS sont soit des **croyances** (beliefs) soit des **buts** (goals)
- les **croyances** = **information qu'a un agent sur lui même et son environnement**
- les **buts** = **états de l'environnement** que **l'agent** souhaite **atteindre** par ses actions
- les **agents** utilisent **KQML** pour **communiquer** sur le **contenu** de leur propre et des autres **VKBs**

Un message KQML

Syntaxe d'un message KQML	Exemple
(<**performative**>	(**tell**
:sender <expéditeur>	:**sender** AgBernard
:receiver <destinataire>	:**receiver** AgPaul
:in-reply-to <xxxxx>	:**in-reply-to** id7.24.97.45391
:ontology <nom-de-l'ontologie utilisée>	:**ontology** Onto1
:language <langage du contenu>	:**language** Prolog
:content <contenu>**)**	:**content** "prix(ISBN3429459,24.95)"**)**

(tell :sender AgBernard :content (PRICE IBM 14) :receiver AgPaul :in-reply-to id7.24.97.45391 :language Prolog :ontology Onto1)

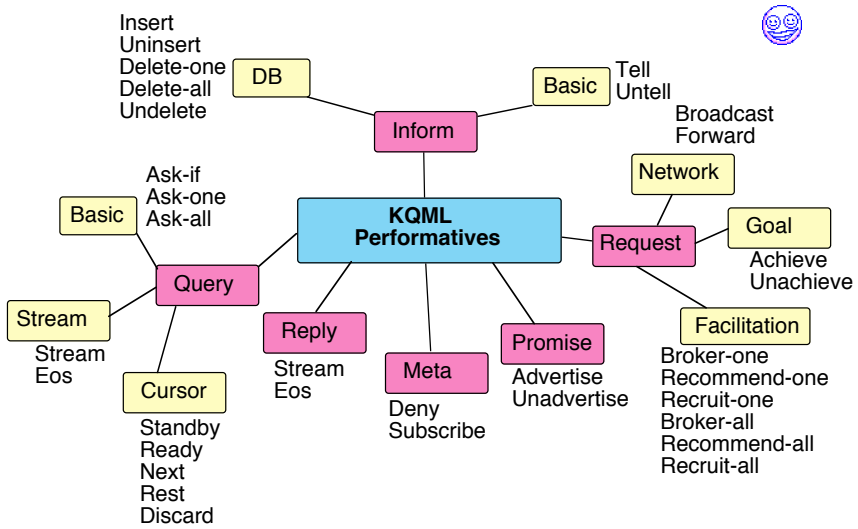
- représente un **acte du langage** ou une **performative** : ask, tell, reply, subscribe,…
- avec une **sémantique et un protocole** associés **:**

  tell(i,j, $B_i\phi$) = fp[B_i $B_i\phi \wedge \neg B_i$ (Bif_j $B_i\phi \vee Uif_j$ $B_i\phi$)] ∧ re[B_j $B_i\phi$] ...
- avec une liste de paires d'**attribut/valeur :** :content, :language, :from, :in-reply-to, ...

Mots-clé paramètres réservés de KQML (1997)

:sender	**l'expéditeur** actuel de la performative
:receiver	le **destinataire** actuel de la performative
:from	**l'origine** de la performative dans :content quand un forward est utilisé
:to	la **destination finale** de la performative dans :content quand un forward est utilisé
:in-reply-to	le label attendu dans une **réponse à un précédent message** (de même que la valeur :reply-with du message précédent)
:reply-with	le label attendu dans une **réponse à le message courant**
:language	le **nom du langage de représentation de** :content
:ontology	le **nom de l'ontologie** (e.g., ense. de definitions de termes) assumé dans le paramètre :content
:content	**l'information** sur laquelle la performative exprime une attitude.

Performatives de KQML (1997)



Performatives de KQML en détail (1)

Dans KQML, tout agent est vu comme **gérant une base de donnée (BD)**

QUERY

Performative		Signification / Effet attendu
Basic	**ask-about**	A veut toutes les expressions sur P de la BC de B
	ask-all	A veut toutes les réponses de B à une question
	ask-one	A veut une réponse de B à une question
Stream	**stream-about**	Version à réponses multiples de 'ask-about'
	stream-all	Version à réponses multiples de 'ask-all'
	eos	Fin d'une suite de réponse à une requête préalable
Cursor	**standby**	A veut que B soit prêt à répondre à un performatif
	ready	A est prêt à répondre à un performatif préalable de B
	next	A veut la réponse suivante à une requête préalable
	discard	A ne veut pas les réponses suivantes de B

A : agent Expéditeur du message, B: agent Destinataire, BC : Base de Connaissances d'un agent

Performatives de KQML en détail (2)

INFORM

Performative		Signification / Effet attendu
Basic	**tell**	A indique qu'une expression P fait partie de la BC de A
	un-tell	indique qu'une expression ne fait pas partie de la BC de A
DB	**insert**	A veut que B insère une expression P dans sa BC
	delete	A veut que B supprime une expression de sa BC
	delete-all	A veut que B supprime toutes les P correspondantes de sa BC
	delete-one	A veut que B supprime une expression P de sa BC
	undelete	A veut que B supprime une expression de sa BC

A : agent Expéditeur du message, B: agent Destinataire, BC : Base de Connaissances d'un agent

Performatives de KQML en détail (3)

REQUEST

Performative		Signification / Effet attendu
Network	**brodcast**	A veut que B envoie un performatif à toutes ses accointances
	forward	A veut que B délègue un performatif
Goal	**achieve**	A veut que B accomplisse une action
	unachieve	A veut que B accomplisse une action
Facilitation	**broker-all**	A veut que B récupère toutes les réponses à une performative
	broker-one	A veut de l'aide pour récupérer la réponse à une performative
	recruit-all	A veut que tous les agents capables répondent à une performative
	recruit-one	A veut qu'un agent capable réponde à une performative
	recommend-all	A veut tous les agents qui répondent à une performative donnée
	recommend-one	A veut un agent qui réponde à une performative donnée

A : agent Expéditeur du message, B: agent Destinataire, BC : Base de Connaissances d'un agent

Performatives de KQML en détail (4)

REPLY

Performative		Signification / Effet attendu
	stream-about	Version à réponses multiples de 'ask-about'
	stream-all	Version à réponses multiples de 'ask-all'
	eos	Fin d'une suite de réponse à une requête préalable

META

Performative		Signification / Effet attendu
	deny	A indique à B que la performative ne s'applique plus à B
	subscribe	A veut que B mette à jour ses réponses à une performative

PROMISE

Performative		Signification / Effet attendu
	advertise	A indique qu'il peut faire P
	unadvertise	A indique qu'il peut faire P

A : agent Expéditeur du message, B: agent Destinataire, BC : Base de Connaissances d'un agent

Exemple de dialogue KQML (1)

1. L'agent A1 demande à l'agent A2 le prix de l'imprimante HP-Jet et A2 lui répond :

```
(ask-one
        :sender A1
        :receiver A2
        :content (val (prix HP-Jet))
        :language KIF : ontology imprimantes
        )
(ask-one
        :sender A2
        :receiver A1
        :content (=(prix HP-Jet) (scalar 190 USD))
        :language KIF
        : ontology imprimantes
        )
```

Exemple de dialogue KQML (2)

2. L'agent A1 demande à l'agent A2 toutes les informations concernant l'imprimante HP-Jet, et A2 lui répond par plusieurs messages qui se terminent avec un 'eos':

```
(stream-about
        :sender A1
        :receiver A2
        :reply-with hpj
        :language KIF : ontology imprimantes
        :content HP-Jet
        )
(tell
        :sender A2 :receiver A1
        :in-reply-to hpj
        :content (=(resolution HP-Jet) (scalar 300 dpi))
        :language KIF : ontology imprimantes
        )
(tell
        :sender A2 :receiver A1
        :in-reply-to hpj
        :content (=(prix HP-Jet) (scalar 190 USD))
        :language KIF : ontology imprimantes
        )
 (eos
        :sender A2 :receiver A1
        :in-reply-to hpj
        )
```

Exemple de syntaxes de performatives KQML

Informatives	Database Informatives	Query
tell :content <expression> :language <word> :ontology <word> :in-reply-to <expression> :force <word> :sender <word> :receiver <word>	**insert** :content <expression> :language <word> :ontology <word> :reply-with <expression> :in-reply-to <expression> :force <word> :sender <word> :receiver <word>	**evaluate** :content <expression> :language <word> :ontology <word> :reply-with <expression> :sender <word> :receiver <word>
deny :content <performative> :language KQML :ontology <word> :in-reply-to <expression> :sender <word> :receiver <word>	**delete** :content <performative> :language KQML :ontology <word> :reply-with <expression> :in-reply-to <expression> :sender <word> :receiver <word>	**reply** :content <expression> :language KQML :ontology <word> :in-reply-to <expression> :force <word> :sender <word> :receiver <word>
untell :content <expression> :language <word> :ontology <word> :in-reply-to <expression> :force <word> :sender <word> :receiver <word>		**ask-one** :content <expression> :aspect <expression> :language <word> :ontology <word> :reply-with <expression> :sender <word> :receiver <word>

Facilitateurs KQML

Les facilitateurs = classe spécifique d'agents distribuant des méta-informations sur les autres agents et offrant des services de communication tels que :

- **retransmission** et **distribution des messages** (message forwarding and broadcasting)
- **découverte des ressources**
- **routage** basé sur le contenu du message
- **appariement**

Les **performatives** principalement utilisées pour les facilitateurs sont :

- `advertise,`
- `broker,`
- `recruit,`
- `recommend,`
- `forward,`
- `broadcast`

dans leurs variantes "P-one" et "P-all".

Les facilitateurs peuvent être des **agents intelligents** ou simplement des "**pages**

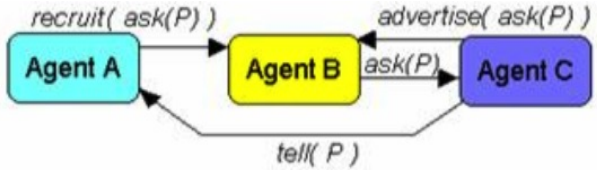
jaunes"

Facilitateurs KQML : exemple de dialogues

1. Schéma des messages échangés entre les agents A et C en utilisant les services du facilitateur B dans le cas de la performative `broker` :

On peut supposer que la performative advertise(ask(P)) a été exécutée par l'Agent C avant que l'Agent A n'ait lancé broker(ask(P)).

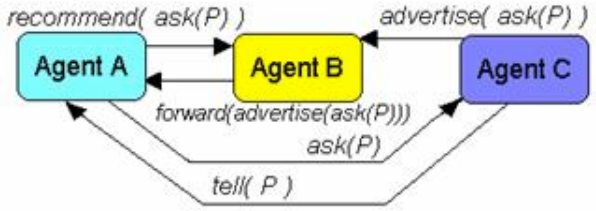
2. Dialogue de 2 agents pour la réalisation de la performative `recruit` à l'aide du facilitateur B :



Facilitateurs KQML : exemple de dialogues

3. Dialogue de 2 agents et exécution d'une performative `recommend` à l'aide du facilitateur B :

le facilitateur B retransmet vers l'agent A une éventuelle performative **`advertise(ask(P))`** reçue antérieurement de la part de l'Agent C.



après la réception du message **`forward(advertise(ask(P)))`** émis par le facilitateur B, l'Agent A ouvre un dialogue direct avec l'Agent C.

4. Le langage ACL - FIPA

- **Actes de communication de ACL-FIPA**
- **Performatives ACL-FIPA**
- **Similitudes et différences entre KQML et FIPA-ACL**
- **Mise en œuvre de KQML et ACL-FIPA**

Introduction au langage ACL - FIPA

La **FIPA** (Foundation for Intelligent Physical Agents) créée en 1996 pour promouvoir l'usage des agents logiciels dans l'industrie, a spécifié un langage de communication entre agents nommé FIPA-ACL.

FIPA-ACL a une syntaxe similaire à KQML et s'appuie sur la définition de 2 ensembles :

- un **ensemble de 21 actes de communication primitifs**, auquel s'ajoutent les autres actes de communication pouvant être obtenus par la composition des ces actes de base
- un ensemble de **messages prédéfinis** que tous les agents peuvent comprendre

Ces 21 actes de communication sont exprimés par des **performatives** regroupées en 5 groupes :

- **passage d'information : `inform*, inform-if (macro act), inform-ref (macro act), confirm*, disconfirm*`**
- **réquisition d'information : `query-if, query-ref, subscribe`**
- **négociation : `accept-proposal, cfp, propose, reject-proposal`**
- **distribution de tâches (ou exécution d'une action) : `request*, request-when, request-whenever, agree, cancel, refuse`**
- **manipulation des erreurs : `failure, not-understood`**

ACL- FIPA : actes primitifs et composés

- En FIPA-ACL il n'y a pas de primitives de gestion ni de facilitation
- **actes communicatifs primitifs** : sont définis de façon atomique, c'est-à-dire qu'ils ne sont pas définis à partir d'autres actes (suivis d'une étoile "*").
- **actes communicatifs composés** : sont définis à partir d'autres actes par **l'une des opérations suivantes :**
  - **un acte fait partie d'un autre acte**, à travers l'opérateur de composition " **;** " indiquant une séquence d'actions
  - à travers l'opérateur de composition " **I** " indiquant un choix non déterministe de l'action

Exemple de message en ACL-FIPA :

```
(inform
        :sender A
        :receiver B
        :reply-with laptop
        :language KIF
        :ontology ordinateurs
        :content (=(prix HP-Jet) (scalar 1500 USD))
        :reply-by 10
        :conversation-id conv01
        )
```

Performatives ACL- FIPA

Performatives de ACL-FIPA :

Performative	Description
`sender`	L'émetteur du message
`receiver`	Le destinataire du message
`reply-to`	Participant à l'acte de communication
`content`	le contenu du message (l'information transportée par la performative)
`language`	le langage dans lequel le contenu est représenté
`encoding`	décrit le mode d'encodage du contenu du message
`ontology`	le nom de l'ontologie utilisé pour donner un sens aux termes utilisés dans le content
`protocol`	contrôle la conversation
`conversation-id`	identificateur de la conversation
`reply-with`	identificateur unique du message, en vue d'une référence ultérieure
`in-reply-to`	référence à un message auquel l'agent est entrain de répondre (précisé par l'attribut reply-with de l'émetteur)
`reply-by`	impose un délai pour la réponse

Similitudes et différences entre KQML et FIPA-ACL

- **ils sont très similaires sur leurs concepts et principes de base**
- **ils ont la même syntaxe** : un message de KQML et un message de FIPA-ACL sont syntaxiquement identique (excepté différences sur les noms des performatives)
- **ils n'impliquent tous les deux aucun engagement** pour un langage pour le contenu

ils diffèrent :

- **au niveau de la description de la sémantique :**
  - pré-conditions, post-conditions, et conditions d'accomplissement pour KQML
  - pré-conditions de faisabilité et effets rationnels pour FIPA-ACL
- **au niveau du choix et des définitions des modalités** qu'ils utilisent (langage employé pour décrire les états des agents).
- **dans la gestion des agents** (tâches d'enregistrement, localisation, appartenance à un groupe,...) :
  - KQML : ces tâches sont associées aux performatives que le langage traite en tant que propositions de premier ordre
  - FIPA-ACL : ces tâches individuellement ne sont pas associées aux performatives mais comme des actions rattachées aux performatives

=> traduction systématique impossible entre performatives KQML et FIPA

Mise en œuvre de KQML et FIPA-ACL

Tout SMA utilisant KQML ou FIPA-ACL doit fournir les éléments suivants :

- **Niveau 1 :** une **suite d'APIs** qui facilitent la composition, l'envoi et la réception des messages de ACL
- **Niveau 1 :** une **infrastructure des services** aidant la gestion des agents comme l'enregistrement, l'identification, la recherche d'agents, déclaration des services offerts, etc.
- **Niveau 1 :** un **code pour chaque type de message réservé** (acte communicatif ou performative) qui prend en charge la sémantique des actions relatives à un domaine d'application

Mise en œuvre :

- le **programmeur** intervient normalement seulement à ce **niveau 3**
- au **niveau 1 et 2 existence de composants réutilisables** pouvant être intégrer dans le code de l'application.
- **aucune condition** sur le **langage de programmation** ni sur **la plate-forme** utilisée pour l'implantation
- **plusieurs implantations** de KQML et FIPA-ACL (JADE, JATLite, …)

5. Ontologies communes

- une **représentation partagée** est nécessaire pour réussir une communication
- pour les êtres humains : réalisé par des mondes physique, biologique et social communs
- pour les agents logiciels: usage d'une **ontologie commune** définissant clairement :
  - les **termes** utilisés dans la communication
  - les **politiques d'interaction** mises en oeuvres
- la spécification d'une ontologie nécessite l'emploie d'un **langage de représentation des connaissances** (Ontolinga, ...)
- **travaux majeurs** sur les ontologies :
  - Cyc
  - DARPA ontology sharing project
  - Ontology Base (ISI)
  - WordNet (Princeton)
  - ...

Ontologies communes

Une ontologie définie des catégories et des relations entre ces catégories ou d'autres informations

Un exemple d'ontologie utilisant le langage SHOES :

```
<ONTOLOGY ID="cs-dept-ontology" VERSION="1.0">
Categories for this example:

<DEF-CATEGORY NAME="Person" ISA="base.SHOEEntity">
<DEF-CATEGORY NAME="Worker" ISA="Person
<DEF-CATEGORY NAME="Faculty" ISA="Worker">
<DEF-CATEGORY NAME="Student" ISA="Person">
<DEF-CATEGORY NAME="Professor" ISA="Faculty">

Relationships for this example:
<DEF-RELATION NAME="advisor">
<DEF-ARG POS="1" TYPE="Student">
<DEF-ARG POS="2" TYPE="Professor">
</DEF-RELATION> <DEF-RELATION NAME="tenured">
<DEF-ARG POS="1" TYPE="Professor">
<DEF-ARG POS="2" TYPE=".TRUTH">
</DEF-RELATION>
```


6. Langage de conversation

- L'interopérabilité est plus que le seul échange de messages
- comment **organiser des séquences de messages structurés en conversation entre agents**
- comment **développer des protocoles d'interaction** : ensemble de conversation finalisé

=> langage de conversation ou d'interaction

Approches par graphes d'états :

- **COOL (COOrdination Language** - Barbuceaunu / Toronto)
- **RCA (Tranvouez - Espinasse - Marseille)**
- **Agentis** (Westminster), ...

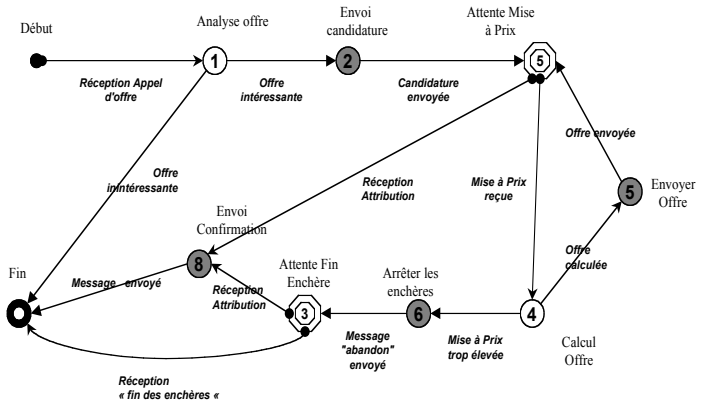
Approches par Réseaux de Pétri Colorés :

- **BRIC** (Ferber - Montpellier)
- ...

Approches orientées-objet :

- **AUML** (USA) – **GAIA** (Londres)
- ...

Graphes RCA (Tranvouez & Espinasse 99)



état initial	état final	état d'action élémentaire	État d'action composite	État d'attente illimitée	État d'attente limitée	État de communication	Transition interne	Transition externe
●	◉	○	□	⬡	▽	●	→	⟶

Graphes RCA (Tranvouez & Espinasse 99 - DIAM)

état initial	état final	état d'action élémentaire	État d'action composite	État d'attente illimitée	État d'attente limitée	État de communication	Transition interne	Transition externe
●	◉	○	□	⬡	▽	●	→	⟶

Syntaxe BNF :

"Plan comportemental" := <Plan Local> I <Protocole de Rôle>

<Plan Local> := "État Initial" "Transition Interne" (<État d'Action> "Transition Interne")* "État Final"

<Protocole de Rôle>::= "État Initial" "Transition Interne" (<État PR> <Transition>)* "État Final" I "État Initial" "Transition Externe" (<État PR> <Transition>)* "État Final"

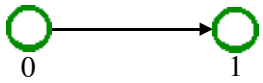
<État PR> ::= <État d'Action> I <État d'Attente> I "État de Communication"
<État d'Action> ::= "État d'Action Elémentaire" I "État d'Action Composite"
<État d'Attente> ::= "État d'Attente Limitée" I "État d'Attente Illimitée"
<Transition> ::= "Transition Interne" I "Transition Externe"

Graphes RCA (Tranvouez & Espinasse 99 - DIAM)

Implémentation en JESS de graphes RCA :

- traduction état par état, transition par transition
- règles de ré-écriture

Ex de traduction RCA -> Jess :



État 0 d'action élémentaire

```
(defrule ACTION-PR-SQ-0
?e <- (etat
                (plan PR-SQ)
                (plan-id ?plan-id)
                (etat-id 0)
)
=>
(retract ?e)
(printout t "Je suis dans l'etat 0." crlf)
(assert
                (resultat
                (plan PR-SQ)
                (plan-id ?plan-id)
                (etat-id 0) )
                (valeur  [ RESULTAT
DE L'ACTION ] ) )
)
) ; FIN defrule ACTION-PR-SQ-0
```

État 1 d'action élémentaire

```
(defrule ACTION-PR-SQ-1
?e <- (etat
                (plan PR-SQ)
                (plan-id ?plan-id)
                (etat-id 1)
)
=>
(retract ?e)
(printout t "Je suis dans l'etat 1." crlf)
(assert
                (resultat
                (plan PR-SQ)
                (plan-id ?plan-id)
                (etat-id 1) )
                (valeur  [ RESULTAT
DE L'ACTION ] ) )
)
) ; FIN defrule ACTION-PR-SQ-1
```

Transition 0 -> 1

```
(defrule TI-PR-SQ-0-1
?res <- (resultat
                (plan PR-SQ)
                (plan-id ?plan-id)
                (etat-id 1) )
                (valeur   [ TESTER LA
VALEUR] )
)
=>
(retract ?res [PLUS AUTRES FAITS]
)
(printout t "Je transite de l'etat init 0
vers l'etat 1 ")
(assert (fin SQ ?plan-id ) )
); FIN  defrule TI-PR-SQ-0-1
```