Agent Communication Languages

FIPA-ACL

Foundation for Intelligent Physical Agents, 1996

FIPA (Foundation for Intelligent Physical Agents) : une organisation qui propose des standards pour l'interopérabilité des agents logiciels.

L'interopérabilité: Capacité de systèmes, unités, matériels à opérer ensembles

FIPA-ACL

- *KQML*: premier essai de standardisation d'un *ACL* mais
  - Pas de sémantique formelle, pas d'infrastructure pour la gestion des agents ,….
- Création en Avril 1996 de la *FIPA* (50 membres)
- Identification et sélection d'applications

  Personal Travel Assistance, Personal Assistant, Audio Visual Entertainment and Broadcasting, Telecommunication Management

- Sélection d'axes technologiques

  Gestion des agents, communication entre agents, Intégration Agent/SW

- De nombreuses spécifications (العديد من المواصفات)

KQML versus FIPA-ACL

KQML [Finin, Labrou]

- Plusieurs performatives (36)
- Un message:
  - Contenu
  - Langage (Java, XML, etc.)
  - Performative
  - Ontologie

FIPA-ACL

- Approche similaire
- Sémantique formelle
- Protocoles

ACL: *ACL-FIPA*[FIPA 97, 99]

- *KQML* et *FIPA-ACL* ont la même syntaxe des messages

- le langage *FIPA-ACL* s'appuie sur la définition de deux ensembles :

  a) **ensemble d'actes de communication primitifs**, auquel s'ajoutent les autres actes de communication peuvent être définis par composition de ces actes de base.

  b) **ensemble de messages prédéfinis ,** que tout agent doit être capable de traiter:

  - *Not-understood*: si l'agent reçoit un message qu'il ne peut pas comprendre. Tout agent doit être capable de traiter un tel message

ACL: ACL-FIPA[FIPA 97, 99]

FIPA-ACL possède **21** actes communicatifs, exprimés par des **performatives**, qui peuvent être groupés selon leur fonctionnalités en **5 catégories d'actes**

1) **Passage d'information** inform*, inform-if (macro act), inform-ref (macro act), confirm*, disconfirm*
2) **Demande d'information** query-if, query-ref, subscribe
3) **Négociation** accept-proposal, cfp, propose, reject-proposal
4) **Réalisation d'action** request*, request-when, request-whenever, agree, cancel, refuse
5) **Gestion d'erreur** failure, not-understood

ACL: ACL-FIPA[FIPA 97, 99]

- Les actes communicatifs peuvent être primitifs ou composés.

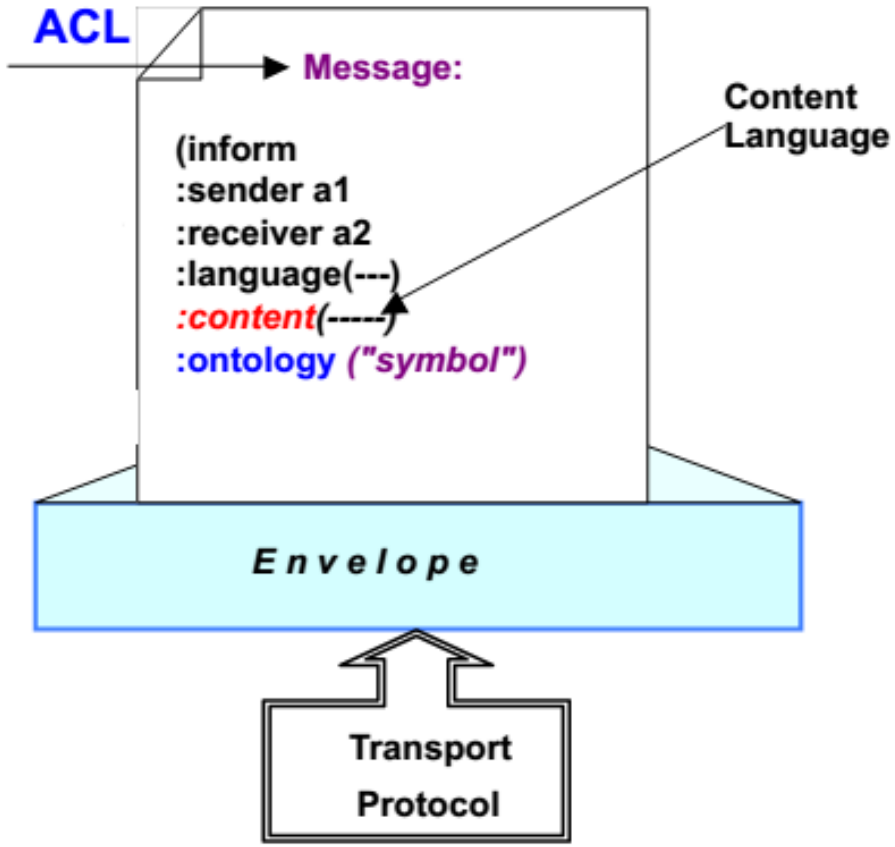
- Les **actes communicatifs primitifs** sont définis de façon atomique, c'est-à-dire qu'ils ne sont pas définis à partir d'autres actes (dans la classification ci-dessus ils sont suivis d'une étoile - "*").

- En revanche, les **actes communicatifs composés** sont définis à partir d'autres actes par l'une des opérations suivantes :
  - un acte fait partie du contenu d'un autre acte, à travers l'opérateur de composition " ; " pour indiquer une séquence d'actions
  - à travers l'opérateur de composition " | " pour indiquer un choix non déterministe de l'action.

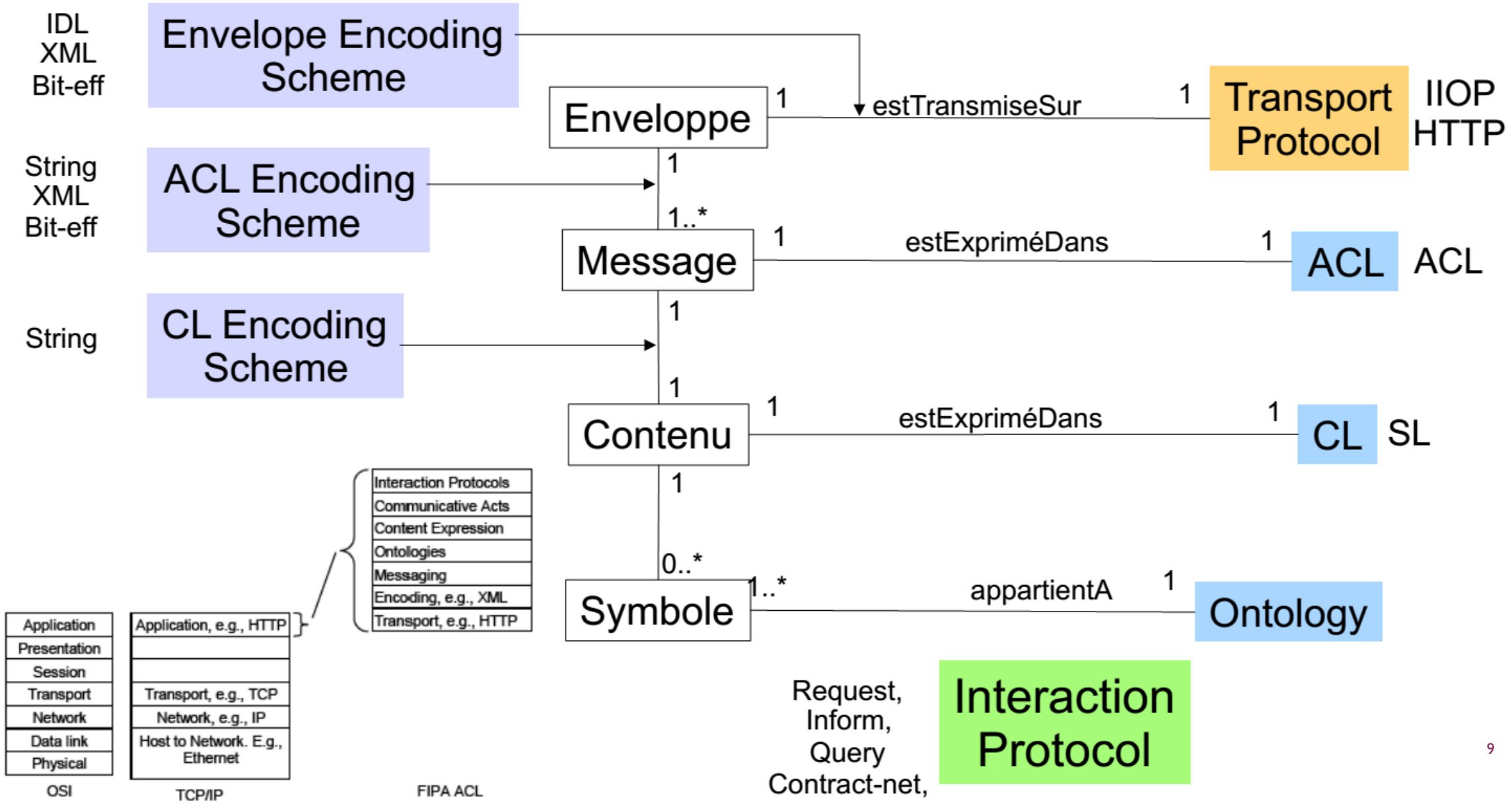
FIPA-ACL : syntaxe

Structure d'un message:

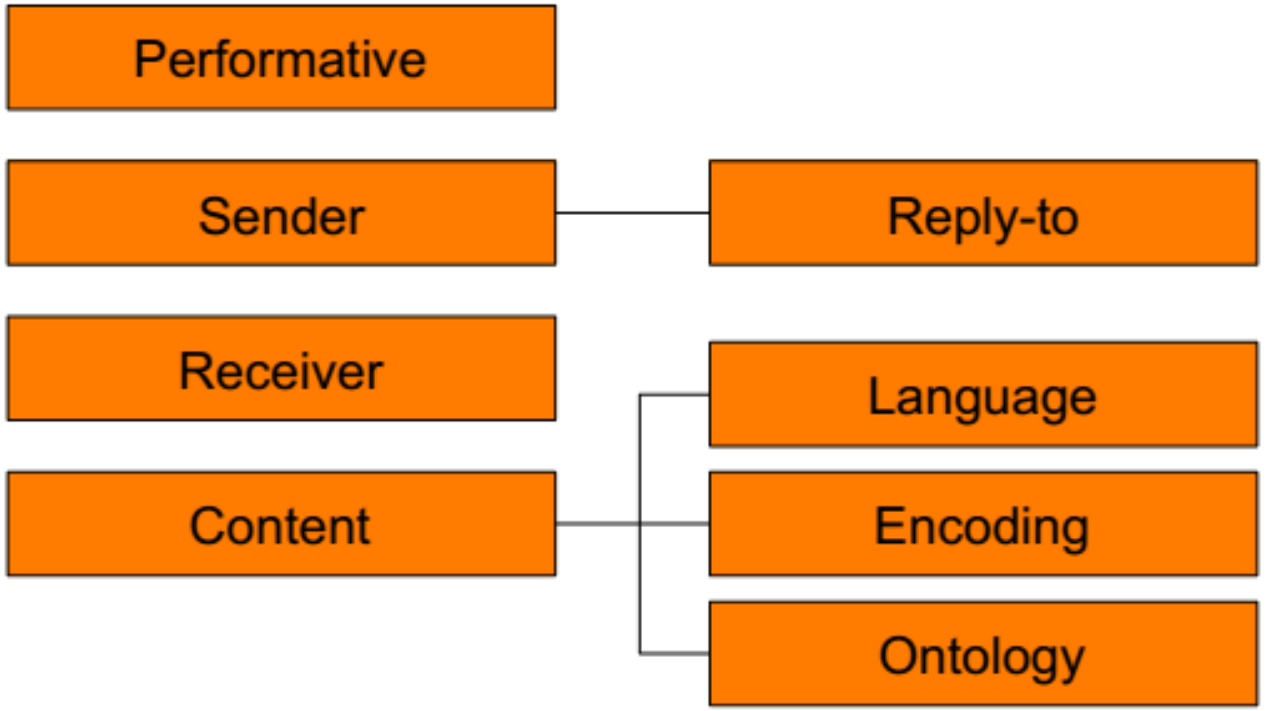
(Acte_de_communication
:content <expression>
:language <mot>
:ontology <mot>
:in-reply-to <expression>
:sender <mot>
:receiver <mot>
)



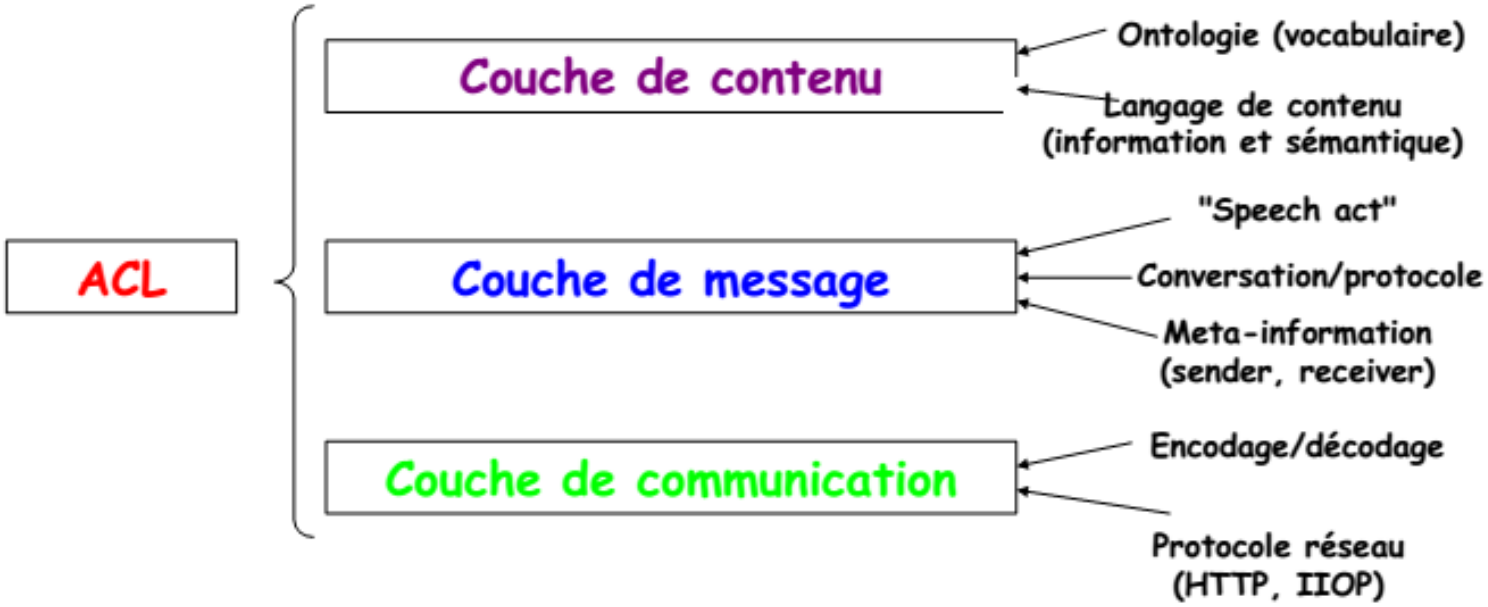
FIPA-ACL : syntaxe



Type de Structure *ACL*



Type de Structure *ACL*



Structure des messages

Mot clé	Description	Objet
performative	Dénote l'acte de communication du message	Type de performative
sender	Dénote le nom de l'agent qui envoie le message	Participant de la conversation
receiver	Dénote le nom de l'agent qui reçoit le message	Participant de la conversation
reply-to	Elément qui le thread de conversation, au lieu du nom de l'agent	Participant de la conversation
content	Dénote le contenu du message : de l'objet ou de l'action	Contenu du message
language	Dénote le langage dans lequel le contenu est exprimé	Description du contenu
encoding	Dénote l'encodage de l'expression du contenu	Description du contenu
ontology	Dénote l'ontologie qui donne du sens à l'expression du contenu	Description du contenu
protocol	Dénote le protocole utilisé pour envoyer les messages ACL	Contrôle de la conversation
conversation-id	Introduit l'identificateur de conversation (expression) qui est utilisé pour la suite d'actes de communication qui constitue la conversation	Contrôle de la conversation
reply-with	Introduit l'expression qui est utilisé par l'agent répondant pour identifier ce message	Contrôle de la conversation
in-reply-to	Dénote l'expression qui fait référencer à l'action précédente au quel ce message répond	Contrôle de la conversation
reply-by	Dénote une expression du temps et/ou de la date qui indique le temps limite auquel l'agent envoyant désire recevoir une réponse	Contrôle de la conversation

Exemple de message :

```
(inform          //type d'acte de communication
    :sender A    //infos. utiles pour le routage
    :receiver B  //infos. utiles pour le routage
    :content (price (bid goood02) 150)  //proposition ou action
    :in-reply-to round-4
    :reply-with bid04
    :encoding 1000
    :language fipa-sl1  //langage utilisé dans content
        :ontology hpl-auction

    :reply-by 10  //deadline pour la réponse
    :protocol offer  //protocole d'interaction utilisé
    :conversation-id conv02) //conversation en cours
```

ACL-FIPA

Conversation

Agent *A* à Agent *B*:

(**evaluate**

:language *KIF*

:ontology moteurs

:reply-with q1

:content (val (km auto1))

)

Agent *B* à Agent *A* en réponse:

(**reply**

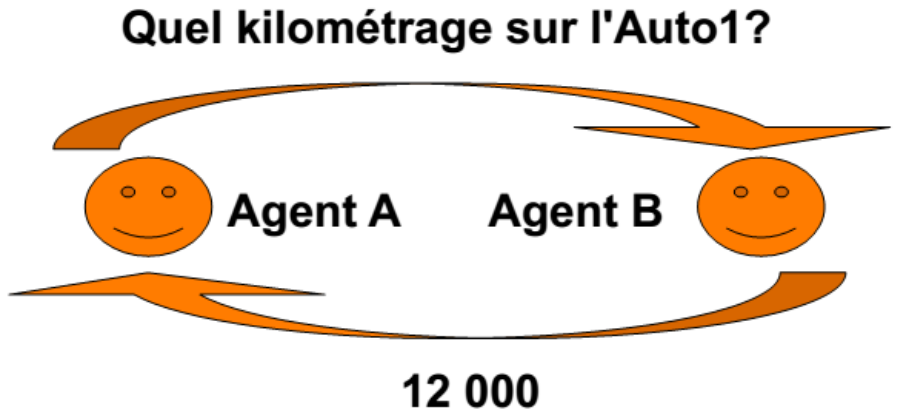
:language *KIF*

:ontology moteurs

:in-reply-to q1

:content (scalar 12000)

)



SL (Semantic Language)

est le langage formel employé pour définir la sémantique des performatives de *FIPA-ACL*.

SL utilise une logique modale des croyances :

- **croyances incertaines**:
  - *B (belief),*
  - *U (uncertain belief)*

  avec les questions associées ***Bif*** **(croire?)** et ***Uif*** **(ne pas croire?)**

- **désirs**:
  - *D (desire)*

- **intentions**:
  - *I (intention ou buts persistants)*

SL peut représenter des propositions, des objets, et des actions.

Modélisation **B.D.I**

Modalités **B,U, I**

$B_i(p)$: L'**agent *i*** croit que ***p*** est **vrai**, **(les connaissances de l'agent locales)**

$U_i(p)$: L'**agent *i*** croit ***p*** plus probable que ¬ ***p*** " , **(les propositions probables)**

$I_i(p)$: L'**agent *i*** veut que ***p*** soit **vrai** , **(les propositions que l'agent souhaite vraies)**

$Bif_i(p)$: signifie que l'**agent *i*** a une **opinion certaine** sur la **vérité** ou la **fausseté** de sa **proposition.**

$Uif_i(p)$: signifie que l'**agent *i* n'est pas certain** la **vérité** ou la **fausseté** de sa **proposition**

$a_1;a_2$: séquence,

$a_1|a_2$: choix

Feasable(a), ***Done(a)***, ***Agent(i,a)***

➢Les **composantes** des **actes** de **communication** planifiés au sein d'un agent caractérisent à la fois;

- le **but** pour lequel l'**acte** est **choisi** et
- les **conditions** qui doivent **être satisfaites** pour que l'acte soit exécuté.

➢ Pour un **acte donné**,

- le premier est désigné sous le nom de **l'effet rationnel** (rational effect), et
- le dernier comme **pré-conditions** de **faisabilité** (feasibility preconditions) qui sont les **qualifications** de **l'acte**.

Ainsi, chaque **acte** de **communication** est défini de cette manière, ce qui permet aux **agents** de **raisonner** et d'avoir un **comportement rationnel cohérent.**

La **sémantique** d'un **acte** de **communication** (***CA***) est spécifiée comme un **ensemble** des **formules** en ***SL*** qui décrivent :

- **Précondition de faisabilité** (Feasibility Preconditions) (*FPs*)
  les conditions nécessaires pour l'émetteur; l'émetteur n'est pas obligé à faire le *CA*

- **Effet rationnel** (Rational Effect) (*REs*)
  l'effet que l'agent peut s'attendre à suite de la *CA*; indique les conditions qui doivent être vraies pour le récepteur

- **Récepteur**
  il n'est pas obligé à assurer l'effet

- **Emetteur**
  il ne peut pas être sûr que l'effet sera accompli

Performatif ***Inform***

Message : ***Inform (p)*** : Agent ***i*** → Agent ***j*** **(**Agent ***i*** dit à l'agent ***j*** que ***p*** est **vrai)**

Pré-condition

- Agent ***i*** croit que ***p*** est **vrai**
- Agent ***i*** ne croit pas que agent ***j*** connait déjà ***p***

Effet

- Agent ***j*** croit que ***p*** est **vrai**

NB: ***p*** peut être **négative**

Performatif *Inform*

Formellement : $< i, inform\ (j, p) >$

Pré-condition : $B_i(p) \wedge \neg B_i(Bif_j\ (p) \vee Uif_j\ (p))$

Effet : $B_j(p)$

Avec $Bif_x\ (\phi) \equiv B_x(\phi) \vee B_x(\neg \phi)$

Et $Uif_x\ (\phi) \equiv U_x(\phi) \vee U_x(\neg \phi)$

Performatif ***Inform-if***

Message : ***Inform-if (p)*** : Agent ***i*** → Agent ***j*** **(**Agent ***i*** dit à l'agent ***j*** que ***p*** est **vrai** ou **faux)**

Pré-condition

- Agent ***i*** connait la valeur de vérité de ***p*** (ou de ¬p)
- Agent ***i*** ne croit pas que agent ***j*** connait déjà ***p***

Effet

- Agent ***j*** croit que ***p*** est **vrai** ou **faux**

Performatif *Inform-if*

Formellement : $< i, inform\ –if\ (j, p) > \equiv\ < i, inform\ (j, p) > \vee < i, inform\ (j, \neg p) >$

Pré-condition : $B_i(p) \wedge \neg B_i(Bif_j(p) \vee Uif_j(p))$

Effet : $B_j(p)$

Avec $Bif_x(\phi) \equiv B_x(\phi) \vee B_x(\neg \phi)$

Et $Uif_x(\phi) \equiv U_x(\phi) \vee U_x(\neg \phi)$

Performatif ***Query-if***

Message : ***Query-if (p)*** : Agent ***i*** → Agent ***j*** **(**Agent ***i*** demande à l'agent ***j*** si ***p*** est **vrai)**

Pré-condition

- Agent ***i*** ne connait pas la valeur de vérité de ***p***
- Agent ***i*** ne croit pas ***p*** et ***i*** ne croit pas ¬ ***p***
- Agent ***i*** croit que ***j*** peut lui donner cette valeur

Effet

- Agent ***j*** dit à l'agent ***i*** que ***p*** est **vrai** ou **faux**

Performatif ***Query-if***

Formellement : $< i, Query\text{–}if\ (j, p) >$

Pré-condition : $\neg\, Bif_i(P) \wedge \neg\, Uif_i(P) \wedge B_i(Bif_j(P)) \wedge \neg\, B_i(I_j(Done(< j, Inform - if(i,P) >)))$

Effet : $Done(< j, Inform - if(i,P) >)$

Example : Agent ***i*** asks agent ***j*** if ***j*** is registered with server *d1*

```
(query-if_      :sender i
                :receiver j
                :content registered(j,d1)
                :language Prolog)
```

Performatif *REQUEST*

Message : **request *(p)*** : Agent ***i*** → Agent ***j*** **(**Agent ***i*** demande à l'agent ***j*** de faire ***p*** **)**

Pré-condition

- Agent ***i*** croit que **< *i, p* >** est possible i.e. L'ensemble des pré-conditions est satisfait
- Agent ***i*** croit que ***j*** peut faire ***p***
- Agent ***i*** croit que ***j*** n'a pas déjà l'intention de faire ***p***

Effet

p a été effectuée

Performatif *REQUEST*

Formellement : $< i, \text{request}(j, P) >$

Pré-condition : $Pre(a)_{[i|j]} \wedge (B_i(Agent(j,a) \wedge \neg B_i(I_j(Done(a))))$

Effet : $Done(a)$

Avec $Pre(a)$: Pré-condition de a

$$Pre(a)_{[i|j]} = \{p \in Pre(a) / B_i(p)\}$$

Agent *i* requests agent *j* to open a file

```
(request        :sender i
                :receiver j
                :content "open \"db.txt\" for input"
                :language vb)
```

Différences entre *ACL* et *KQML*

- Syntaxe identique sauf pour les primitives
  (ex : « tell » pour KQML, « inform » pour ACL)

- différence dans les descriptions sémantiques:
  - Définition des primitives différentes
    - pré-àconditions et les post-conditions pour KQML
    - Faisabilité et effet pour ACL

  - Utilise un langage différent pour décrire les états mentaux des agents