Communication dans les *SMA*

Langages de communication d'agents

KQML

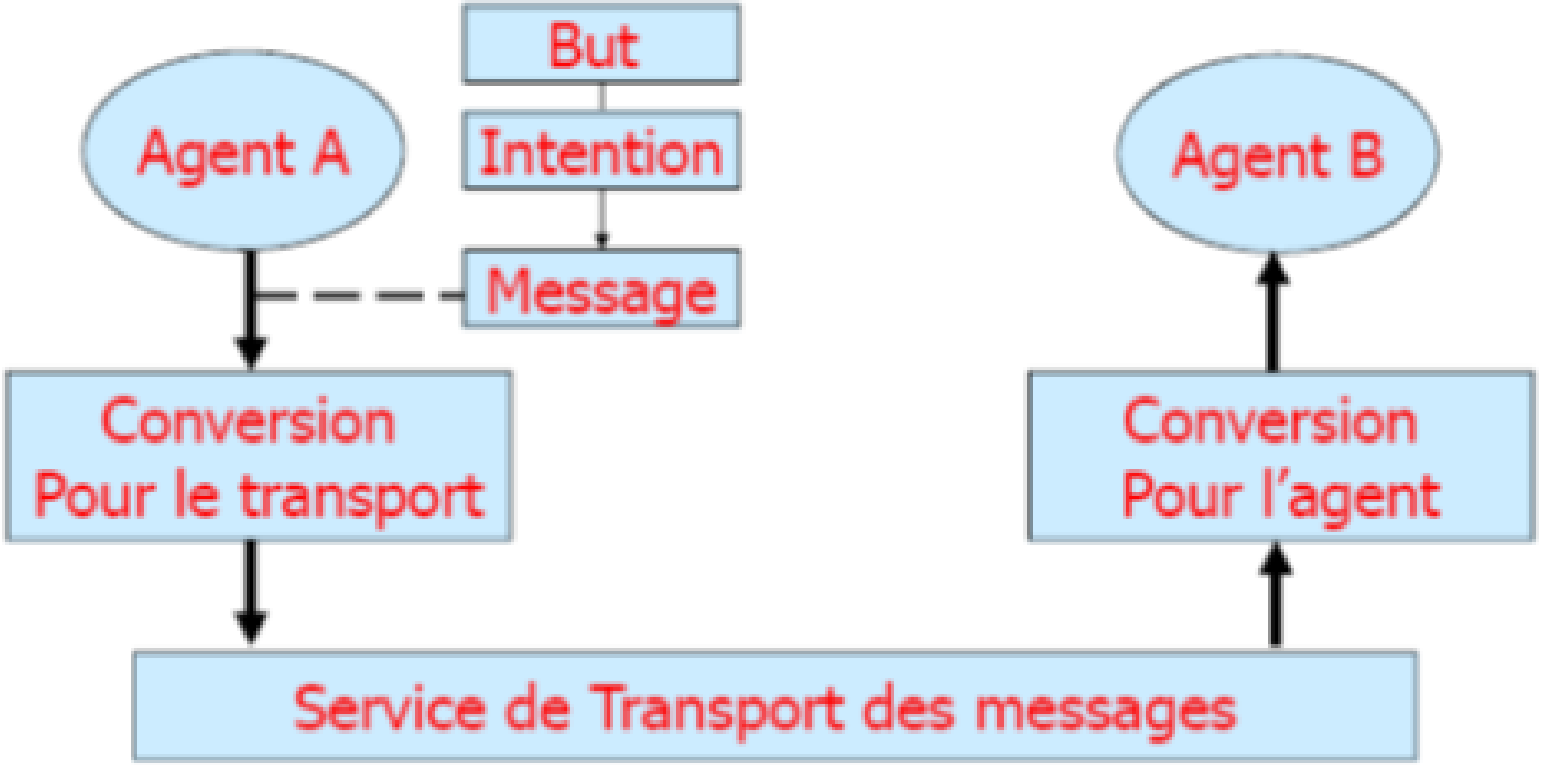
(**K**nowledge **Q**uery and **M**anipulation **L**anguage)

La communication : pourquoi?

- Les agents fonctionnent plus efficacement en groupe et peuvent réaliser ce qu'un agent ne pourrait faire seul;
- Pour collaborer, négocier et se coordonner, les agents ont besoin d'interagir;
- La communication est une forme d'interaction.
- Les agents ont besoin de partager leurs croyances, leurs désirs et leurs intentions.

La communication: Comment ?

- Utiliser un langage de haut niveau compréhensible par tous les agents d'un système
- Les agents utilisent des langages de communication : les ***ACL*** (**A**gent **C**ommunication **L**angages).



Communication entre agents

Protocoles de communication

permettent aux agents d'échanger et comprendre les messages.

Protocoles d'interaction

permettent aux agents les échanges structurés منظم des messages.

But de la communication

permet aux agents de:

- coordonner leurs actions et comportement سلوك , une propriété d'un *SMA* avec des agents agissant dans un environnement commun,
- essayer de modifier les états des autres agents,
- essayer de persuader يقنع les autres agents de faire des certaines actions.

Les langages de communication inter agents

Tout langage multi-agents est représenté par une structure de données comprenant les champs

- Émetteur
- Récepteur
- Langage utilisé
- L'ontologie
- Contenu du message

Une **ontologie** est :

- Utilisée pour identifier la source du vocabulaire
- Moyen pour le receveur d'interpréter le contenu du message

Les langages de communication inter agents

- L'utilisation d'un langage commun implique que tous les agents comprennent son vocabulaire sous tous ses aspects concernant:
  - **Syntaxe** : qui précise le mode de structuration des symboles;
  - **Pragmatique :** pour pouvoir interpréter les symboles;
  - **Ontologie :** pour pouvoir utiliser les mêmes mots d'un vocabulaire commun.

- De plus il est nécessaire que les agents sachent bien utiliser le vocabulaire pour atteindre leurs buts, éviter les conflits, coopérer pour exécuter leurs tâches et modifier l'état mental d'un autre agent.

KQML (**K**nowledge **Q**uery and **M**anipulation **L**anguage)

Objectif

Faire communiquer des agents en leur offrant un langage commun de manipulation des connaissances leur permettant d'interpréter les messages des autres agents, au moyen d'une syntaxe et d'une ontologie (ensemble de concepts) partagés.

KQML, *FIPA-ACL*,…

KQML (1993)

Langage né d'une Initative de l'*ARPA* dans le cadre d'un projet de développer des techniques et des méthodes permettant l'organisation de bases de connaissances à grande échelle qui soient partageables et réutilisables par des systèmes d'agents.

KQML (**K**nowledge **Q**uery and **M**anipulation **L**anguage)

KQML est un langage "extérieur" de haut niveau pour les agents, orienté sur l'échange des messages, indépendant de la syntaxe et de l'ontologie du contenu des messages. Il est aussi indépendant du langage utilisé pour coder le contenu des messages (. Prolog, STEP, SQL, KIF etc.).

Le langage ***KQML*** est fondé sur la **théorie** des **actes** de **langage**

Théorie des Actes de Langage (performatif)

❑ Parler (communiquer), c'est modifier l'état mental de ses interlocuteurs, donc c'est agir.

❑ Trois aspects (ou actes) d'un énoncé

- **Locutoire** : l'action de dire
- **Illocutoire** : l'action que souhaite l'auteur
- **Perlocutoire** : ce que comprends l'interlocuteur

Exemple:

Hamid dit à Riad «j'ai froid !»

- **Composante locutoire** : Hamid produit cette phrase
- **Composante illocutoire** : Hamid considère ce message comme une demande ou comme un ordre de fermer la fenêtre
- **Composante perlocutoire** : Riad ferme la fenêtre ou ne fait rien

Théorie des Actes de Langage (performatif)

Langage sémantique

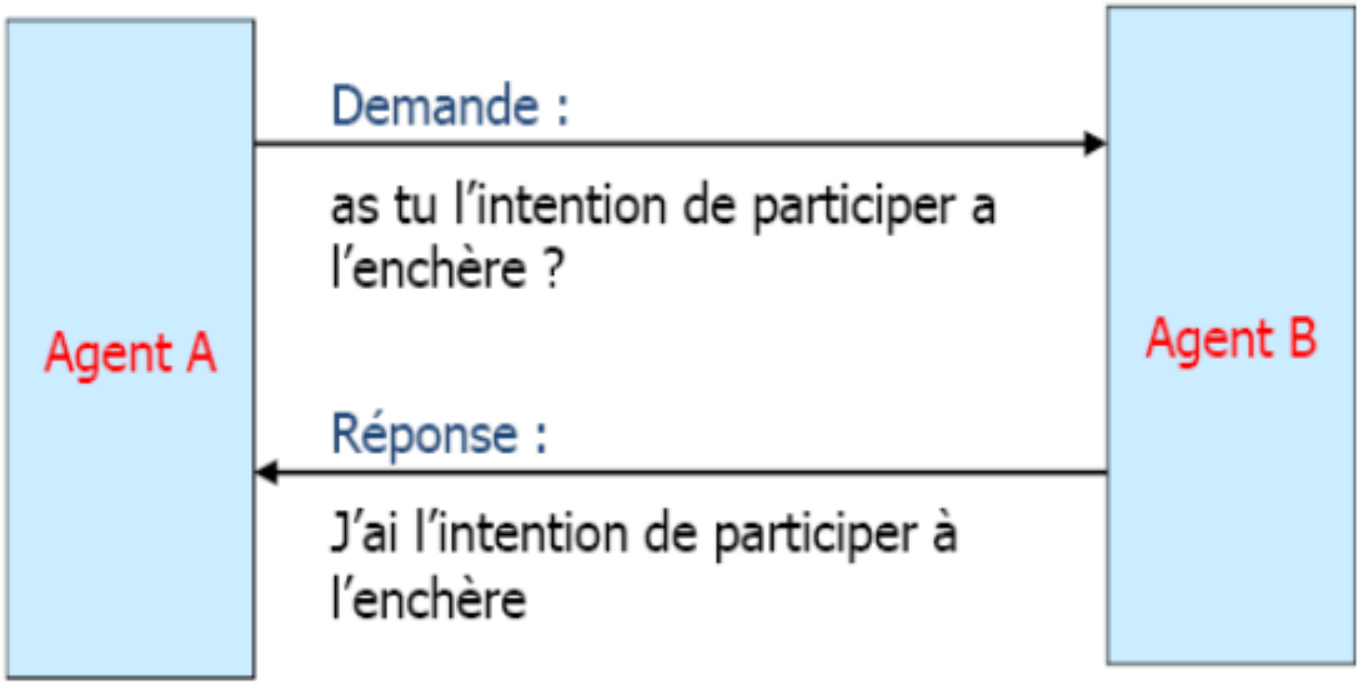
C’est l’expression d’un état mental de l’émetteur par le biais du message; Pour arriver à une reconnaissance par l’auditeur de cet état mental. Il définit les 3 attitudes mentales de l’agent (*Belief*, *Desire*, *Intention*) suivant l’acte utilisé :

- **Affirmation** = expression de la croyance de l’émetteur
- **Demande** = expression du désir (l’émetteur veut que le récepteur fasse une action)
- **Promesse** = expression de l’intention de l’émetteur d’exécuter une action

Théorie des Actes de Langage (performatif)

Interprétation

- Un acte peut être interprété différemment selon les agents et leurs contextes d'exécution
- Pour se comprendre, 2 agents doivent utiliser le même système d'interprétation des messages



Théorie des Actes de Langage (performatif)

Aspect illocutoire

Locutoire = support de la communication (couches syntaxe et connaissance)

Communication réussie Le destinataire interprète mon message correctement ⟹

Perlocutoire = Illocutoire

Contenu illocutoire

Acte de la forme **Force (Proposition)**

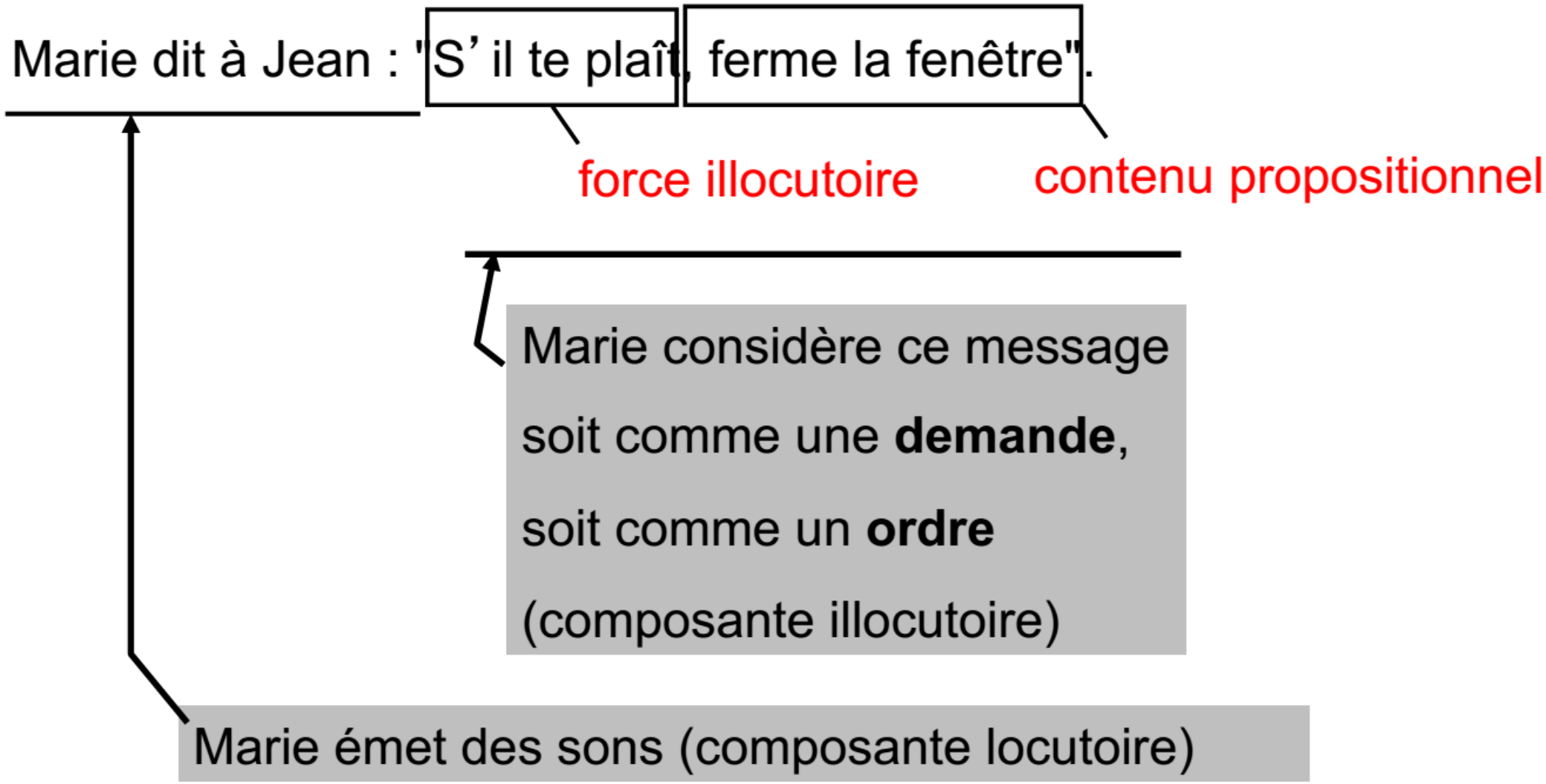
- force illocutoire ou performatif
- contenu propositionnel

Exemples: **affirme**(il_pleut), **demande**(il_pleut)...

➢ **Dans les SMA** Message = Performatif + Proposition

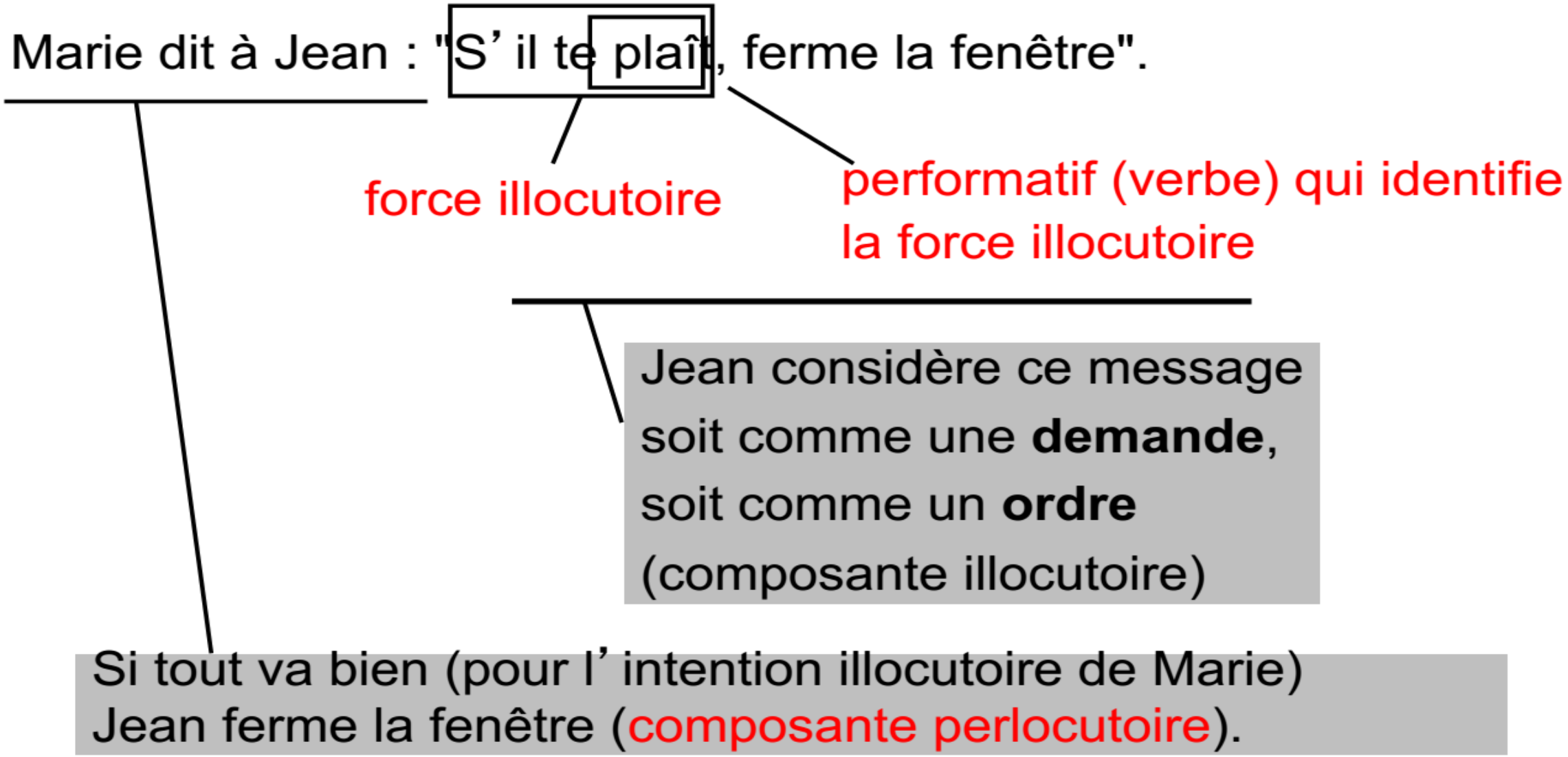
Théorie des Actes de Langage (performatif)

Exemple (Sender)



Théorie des Actes de Langage (performatif)

Exemple (Receiver)



Théorie des Actes de Langage (performatif): Searle 1969 , 5 types d'actes

- **Assertif** : servent à donner une information sur le monde en affirmant quelque chose (description, affirmation, …), exemple : père (pierre, marie).
- **Directif :** sont utilisés pour donner des directives au destinataire (demande, question, ordre...), exemple: quelle heure est-il ?, pousse-toi!, faire(action).
- **Promissif :** engagent les locuteurs à accomplir certains actes (promesse, vœux, menace…), exemple: je passerai demain, ok (action).
- **Déclaratif :** accomplissent un acte par le fait même de prononcer l'énoncé (définition, condamnation …) exemple: la séance est ouverte.
- **Expressif :** servent à donner au destinataire des indications concernant l'état mental du locuteur (excuse , félicitation, remerciement…), exemple: je suis heureux, bien_reçu .

Les couches de *KQML*

Le langage *KQML* est un langage à trois couches :

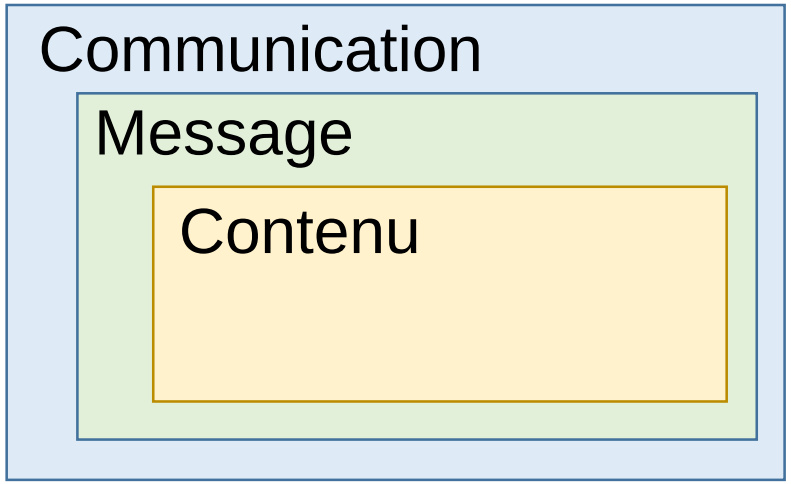


Figure : Les trois couches d'un message *KQML*

Les couches de *KQML*

1. La couche « Contenu »

- Il s'agit du contenu réel du message, il peut être écrit dans le langage de représentation du programme de l'agent.
- *KQML* peut transporter des messages écrits dans n'importe quel langage de représentation (ex : PROLOG, KIF, LISP, C, KQML (lui même), XML ...)
- Deux agents intelligents doivent s'accorder sur le langage de communication à utiliser.

Les couches de *KQML*

2. La couche « Communication »

Dans cette couche on retrouve des informations d'un niveau un peu plus bas permettant le bon fonctionnement de la communication telles que l'identité de l'émetteur ou celle du récepteur du message, ainsi qu'un identificateur unique pour le message.

- les agents échangent des paquets dans la couche communication (enveloppes autour du message qui spécifie des attributs de communication comme ceux concernant l'envoyeur et le destinataire)

Les couches de *KQML*

3. La couche « Message »

- Cette couche constitue le cœur du langage, elle sert à coder le message.
- Elle incorpore des argument décrivant le contenu du message telles que le langage utilisé, l'ontologie…
- Ces arguments permettent à *KQML* d'analyser, acheminer et délivrer les messages même si leur contenu lui est opaque et inaccessible.

KQML

Syntaxe du message

Niveau communication

Niveau message

Niveau contenu

```
(KQML-performatif
        : Emetteur <texte>
        : Récepteur <texte>
        : Langage  <texte>
        : Ontologie <texte>     ; vocabulaire utilisé dans le langage
        : Contenu <texte>       ; le message lui-même
```

Syntaxe du message

❖ **Les paramètres réservés :**

Chacun des trois couches présentées précédemment ajoute un ou plusieurs paramètres à un message *KQML*. La liste complète de ces paramètres est résumée dans le tableau .

Mot-clé	Signification
: content	Contenu de la performative
: sender	Emetteur actuel de la performative
: receiver	Récepteur actuel de la performative
: from	Emetteur initial dans une demande de transfert de message
: to	Destinataire final dans une demande de transfert de message
: reply-with	Référence de la réponse éventuelle
: in-reply-to	Référence attendue lors d'une réponse
: langage	Langage de représentation du paramètre content
: ontologie	L'ontologie supposée du paramètre : content

Tableau : liste des paramètres réservés des performatives en *KQML*

Performatives

36 performatifs répartis en 3 catégories :

- Les 18 performatives de **discours** (ask-if , ask-all, tell, describe, stream-all …)
- Les 11 performatives d'**interconnexion** (register, unregister, broadcast …)
- Les 7 performatives d'**exception** (error, sorry, standby …)

❑ **Discours :** servent à échanger des connaissances et des informations présentes dans la base de connaissance de l'agent

❑ **Interconnexion :** interconnexion entre les agents pour faciliter l'obtention des informations et connaissances

❑ **Exception :** servent à changer le déroulement normal des échanges

KQML **performatives**

Catégorie	Nom
Demande simple	evaluate, ask-if, ask-about, ask-one, ask-all.
Demande de réponses multiples	stream-about, stream-all, eos.
Réponse	reply, sorry.
Information générique	tell, achieve, cancel, untell, unachieve,
Générateur	standby, ready, next, rest, discard, generator
Définition de Compétences	advertise, subscribe, monitor, import, export
Gestion de réseau	register, unregister, forward, broadcast, route

KQML **performatives**

E : l'agent émetteur

R : l'agent récepteur

C : le contenu du message

BVC : la base virtuelle de connaissances (connaissances attribuées par chaque agent aux autres agents)

KQML performatives

Performative	Signification
Achieve	E veut que R rende quelque chose **vrai** dans leur environnement
Advertise	E fait connaître aux autres agents les capacités disponibles dans le traitement d'une performative
ask-about	E veut connaître toutes les propositions pertinentes dans les BVC de R
ask-if	E veut savoir si la réponse à la question précisée en C se trouve dans la BVC de R
ask-one	E veut que seulement R réponde à sa question C
Break	E veut que R interrompe le flux établi par pipe
Broadcast	**E veut que R transmette à son tour la performative á tous ses connexions**
broker-all	E veut que R collecte toutes les réponses á la performative
broker-one	E veut que R l'aide à répondre à une performative
Deny	la performative ne peut pas être appliquée à E
delete-all	E veut que R efface toutes les propositions qui s'apparie avec C de sa BVC
delete-one	E veut que R efface une proposition qui s'apparient avec C de sa BVC
Discard	E ne veut pas le reste des réponses de R à une interrogation
Eos	termine un flux de réponses (en anglais stream) à une interrogation (en anglais eos vient de 'end of stream')
Error	E considère le message précédent de R comme mal formé

KQML **performatives**

Evaluate	E veut que R évalue (simplifie) C
Forward	E veut que R transmette le message vers un autre agent
Generator	la même signification que standby de stream-all
Insert	E demande ã R d'ajouter C à sa BVC
Monitor	E veut que R actualise sa réponse à un stream-all
Next	E veut la réponse suivante de R à un flux d'une performative
Pipe	E veut que R transmette toutes les performatives futures à un autre agent
Ready	E est prêt pour répondre à la performative mentionnée précédemment par R
recommend-all	E veut tous les noms des agents qui peuvent répondre à C
recommend-one	E veut le nom d'un agent qui peut répondre à C
recruit-all	E veut que R 'recrute' tous les agents capables de lui répondre à C
recruit-one	E veut que R 'recrute' un agent capable de lui répondre à C
Register	E peut transmettre des performatives à un certain agent
Reply	communique une réponse attendue
Rest	E veut le reste des réponses de R à une interrogation nommée précédemment
Sorry	R ne peut pas fournir plus d'information

KQML

performatives

Standby	E veut que R soit prêt pour répondre à une performative
stream-about	une version réponse multiple de ask-about
stream-all	une version réponse multiple de ask-all
Subscribe	E veut que R actualise par la suite sa réponse
Tell	E affirme au R que C est dans la BVC de E
transport-address	E associe un nom symbolique à une adresse de transport
Unadvertise	l’action contraire de advertise
Unregister	l’action contraire de register
Untell	E affirme à R que C n'est pas dans la BVC de E

Exemple

Soit le message *KQML* : le message d'un agent *E* (émetteur) qui demande aux autres agents récepteurs (*R*) quel est le prix d'une imprimante à jet d'encre (la question est placée dans le contenu du message)

```
(ask-all
      :content (PRIX JET ?prix)
      :sender Ag1
      :reply-with prixJet
      :ontology imprimantes
      :language PROLOG )
```

- La *'performative'* est *ask-all* qui signifie que l'agent *E* désire que tous les agents *R* répondent à sa question. .

Exemples de *KQML*

- l'agent *A* veut connaître toutes les personnes définies comme étant des hommes.

```
(ask-all
      :sender   A
      :receiver B
      :language PROLOG
      :ontology phylosophie
      :content  homme (=x)
      :reply_with question1
      )
```

Exemples de *KQML*

- l’ agent *B* répond que Socrate est un homme.

```
(tell
    :sender B
    :receiver A
    :language PROLOG
    :ontology phylosophie
    :content homme(Socrate)
    :in_reply_to question1
    )
```

Exemples de *KQML*

- l'agent ***A1*** veut informer l'agent ***A2*** que le bloc ***A*** est sur le bloc ***B***

```
(tell
      :sender A1
      :receiver A2
      :language KIF
      :ontology BlockWord
      :content (And (Block A) (Block B) (On A B))
      )
```

Exemples de *KQML*

Exemple imbriqué

- ***A1*** demande à ***A3*** de transférer le message précédent à ***A2***

```
(forward
      :from A1
      :to A2
      :sender A1
      :receiver A3
      :language KQML
      :ontology kqml-ontology
      :content (tell
                        :sender A1
                        :receiver A2
                        :language KIF
                        :ontology BlockWord
                        :content (And (Block A) (Block B) (On A B))))
```

Exemples des dialogues *KQML*

- 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
        )
```

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

Exemples des dialogues *KQML*

- L'agent *A1* demande à l'agent *A2* toutes les informations concernant l'imprimante *HP-Jet*

```
(stream-about
        :sender A1
         :receiver A2
        :reply-with hpj
         :language KIF
         : ontology imprimantes
         :content HP-Jet
        )
```

Exemples des dialogues *KQML*

- *A2* lui répond par plusieurs messages qui se terminent avec un 'eos':

```
(tell
   :sender  A2
   :receiver  A1
   :in-reply-to  hpj
   :content (= (prix HP-Jet) (scalar 190 USD))
   :language KIF
   : ontology imprimantes
    )
```

```
(tell
   :sender A2
   :receiver A1
   :in-repl-to hpj
   :content (=(resolution HP-Jet) (scalar 300 dpi))
    :language KIF
    : ontology imprimantes
     )

(eos
     :sender A2
     :receiver A1
     :in-reply-to hpj
      )
```

Exemple de *KQML*

Dans une application en Java, l'agent *A* veut connaître la moyenne de 4 nombres

```
(ask-one
        :sender A
        :receiver B
        :language Java
        :ontology mathématique
        :content double m = moyenne_B (n1, n2, n3, n4)
        :reply_with question3
        )
```

Pour chaque performatif

- **Préconditions(Pre)**

  indiquent les états nécessaires dans les agents pour que le message puisse être créé.

- **Postconditions(Post)**

  décrivent les états des agents après le traitement du message

- **Completion**

  décrit le résultat attendu du message.

Exemple

Tell(A,B,X)

Tell(A,B,X)

- Pre(A): bel(A,X) ∧
  know(A,want(B,know(B,bel(A,X))∨know(B,¬bel(A,X))))
- Pre(B): intend(B,
  know(B,bel(A,X))∨know(B,¬bel(A,X)))
- Post(A): know(A,know(B,bel(A,X)))
- Post(B): know(B,bel(A,X))
- Completion: know(B,bel(A,X))

Discussion

Avantages

- Premier « standard » de communication
- Beaucoup d'applications supportent *KQML*
- Langage extensible :
  - Création de nouvelles performatives
  - Création de nouveaux paramètres
  - Création de nouvelles ontologies

Limites

- Des performatifs non-conformes à la théorie des actes du langage.
- ambiguïté et imprécision des performatifs,
- Ne prend pas en compte les conversations;
- certains sont inutiles et incohérents,
- absence de certains performatifs par exemple pas d'expressifs et de déclaratifs.

Ambiguïté et imprécision

- Deny(A,B,Tell(B,A,X))
  - Est-ce que A nie le fait que B ait dit ou
  - Est-ce que A nie le fait X ?

- Uninsert(A,B,X) (possible seulement après insert(A,B,X))
  - Post(B): ¬bel(B,X)
  - Que se passe-til si B peut inférer X à partir d'autre sources?