Communication

M1 Miage 2015–2016 *Systèmes MultiAgents*

Stéphane Airiau

LAMSADE

DAUPHINE
UNIVERSITÉ PARIS

Sources

- chapître 7 de “An Introduction to MultiAgent Systems” de Michael Wooldridge (2009)
- chapître 3 de “MultiAgent System” édité par Gerhard Weiss
- Chopra, Artikis, Bentahar, Colombetti, Dignum, Fornara, Jones, Singh, Yolum. *Research Directions in Agent Communication*, ACM Transactions on Intelligent Systems and Technology (TIST), Vol. V, No. N, July 2012, Pages 1–27.

Introduction

Les Ontologies jouent un grand rôle dans la communication agent, mais cet aspect ne sera pas développé dans ce cours.

- OWL ne possède pas d'aspect temporel
- OWL ne dit rien sur les aspects normatifs ou d'engagement

- autonomie : les agents agissent de façon indépendante
- hétérogénéité : les agents sont développés de façon indépendante

un **protocole** de communication définit

- comment les agents doivent communiquer entre eux
- un protocole aide à assurer l'interopérabilité :
  on peut changer un composant qui respecte un standard et l'ensemble devrait continuer à fonctionner
- thématique classique en systèmes distribués

Systèmes distribués

- ignore complètement l'autonomie et l'hétérogénéité
- les contraintes sont fixées à un bas niveau :
  - spécifie l'ordre des messages et les occurences

Systèmes distribués

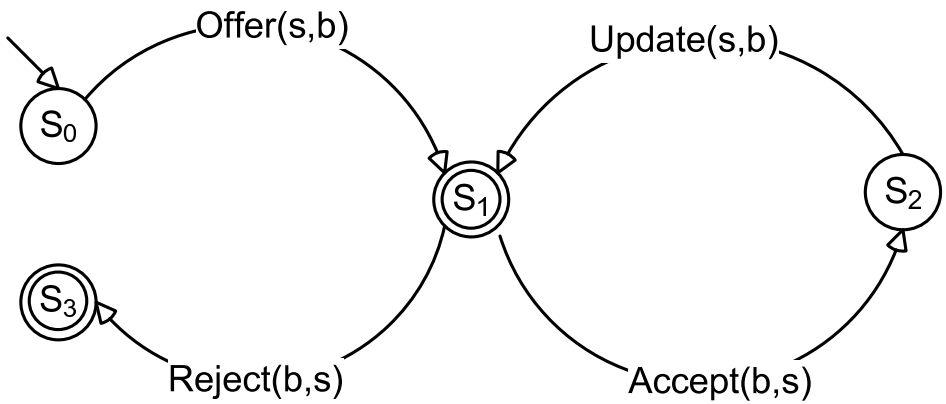
- les agents ont généralement leur propre processus d'exécution
  - s'ils doivent travailler ensemble : il faut communiquer !
- autonomie : on ne veut pas de protocole qui contraignent trop les intéractions car les agents sont libres !
- les protocoles doivent être indépendants de la méthode de raisonnement de l'agent.

dans les systèmes distribués classiques, il suffit souvent de

- préciser le squelette/schémas des messages échangés (ex : serveur smtp)
- les flots de communications légaux (ordre des messages, occurrences)

Exemple : représentation à l'aide d'un automate

- l'automate ne précise pas le sens des messages
- un sens des messages implicite va être un problème pour l'interopérabilité car le sens sera interprété de façon différente
- Développeurs devront définir comment interpréter les messages, ce qui réduira peut être l'hétérogénéité des agents
- ⇨ pas d'engagment *public* ou *social*



Systèmes distributés et génie logiciel

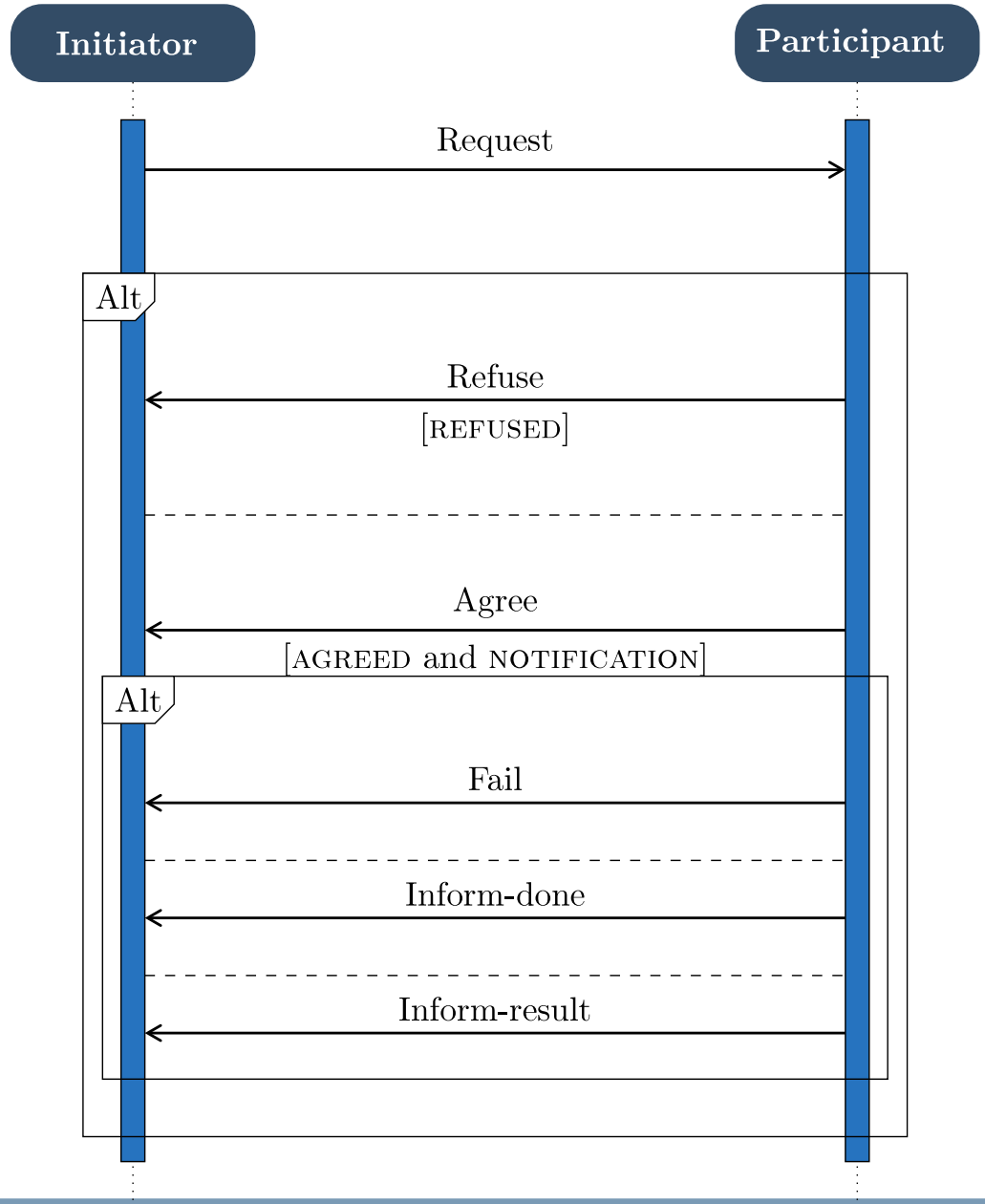
- mais il existe des outils formels pour la vérification des protocoles
- l'implémentation d'agents sera assez naturel

- automates finis
- Réseau de Petri
- diagramme d'états
- Pi-calculus
- logique temporelle
- outils pour les services web (orchestration de services)

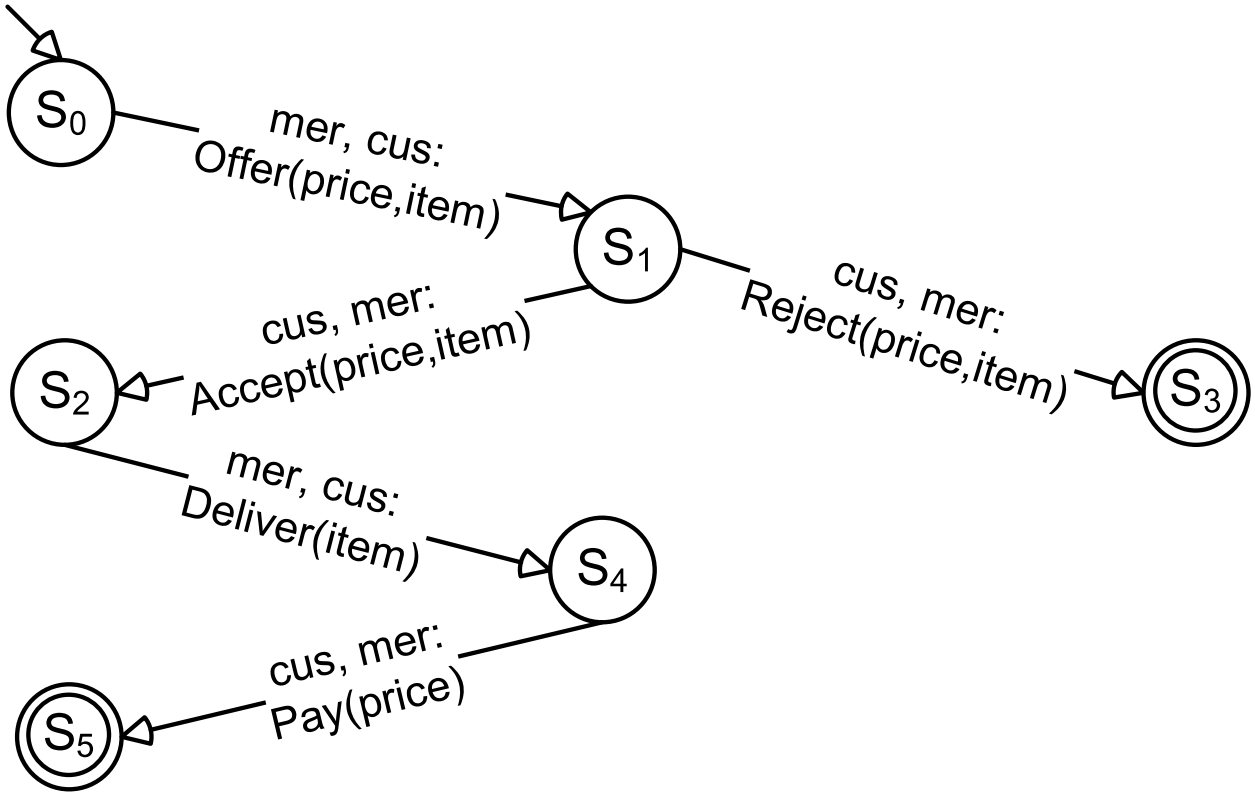
Diagrammes de séquence

- FIPA utilise des diagrammes de séquence UML pour spécifier des protocoles d'interaction :
  - spécifie le rôle de chaque agent
  - encadre le dialogue entre les rôles
- certains aspect sont devenus des parties du standard UML 2.0

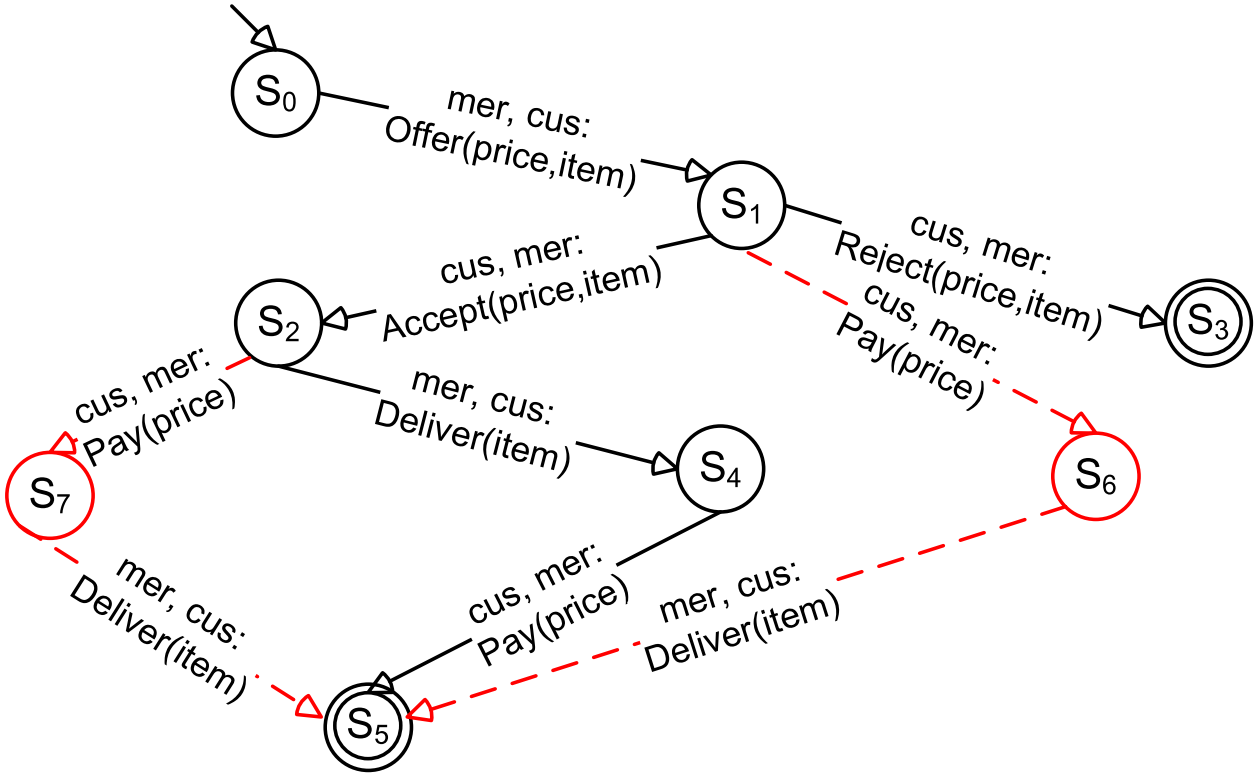
exemple : FIPA Request Interaction Protocol



Automates Finis



Automates Finis



produire des automates avec des chemins additionnels

Approches d'Intelligence Artificielle

- KQML : Knowledge Query and Manipulation Language
- KIF : Knowledge Interchange Format
- FIPA ACL : Agent Communication Language

Speech Acts

- théories pragmatiques du langage : des théories sur l'utilisation du langage ; comment le langage est utilisé pour atteindre leurs buts et leurs intentions
- Austin (1975) "How to *Do* Things with Words"

"Je baptise ce bateau le Titanic"
"Je parie un café qu'il va pleuvoir demain"
Ces phrases changent quelque chose : une nouvelle convention, un engagement pour le futur.
On ne peut dire que ces phrases sont vraies ou fausses

- convention : on s'attend à certaines réponses
- pré-conditions :
  - les circonstances doivent être adéquates
  - l'état mental de l'agent doit être adéquat : pensées/intentions/sentiments appropriés
- les participants doivent se comporter en accord avec ce qu'ils disent

différents types de d'acte de langage

- assertifs ou représentationfs : affirmer, nier, postuler
- directifs : essayer de faire faire quelque chose à l'interlocuteur, demande, conseil
- promissifs : promettre, faire voeu, garantir, parier
- expressifs féliciations, remerciement
- déclaratifs : nomination, baptême

généralement, deux composants :

un verbe performatifs (je demande, j'informe, etc...)

un contenu propositionel ("la porte est fermée")

exemple

- performative = request
  content = “the door is closed”
  speech act = “please close the door”
- performative = inform
  content = “the door is closed”
  speech act = “the door is closed !”
- performative = inquire
  content = “the door is closed”
  speech act = “is the door closed ?”

Langage de Communication

KQML Knowledge Query and Manipulation Language

Hypothèses :

- chaque agent maintient une base de croyance (KB)
- les agents sont coopératifs, sincères
- les croyances constituent une abstraction de l'implémentation de l'agent

deux parties :

- KQML pour exprimer les performatifs
  - `ask-if`
  - `perform`
  - `tell`
  - `reply`
- KIF : langage pour exprimer le contenu

A to B	`(ask-if (> (size chip1) (size chip2)))`
B to A	`(reply true)`
B to A	`(tell (= (size chip1) 20))`
B to A	`(tell (= (size chip2) 18))`

```
1  (stream-about
2    :sender A
3    :receiver B
4    :language KIF
5    :ontology motors
6    :reply-with q1
7    :content m1
8  )
9
10 (tell
11   :sender B
12   :receiver A
13   :in-reply-to q1
14   :content
15   (= (torque m1) (scalar 12 kgf))
16 )
```

Limites

- les performatifs ont été défini de manière informelle, plusieurs implémentations ont existé, et elles ne pouvaient pas communiquer ensemble !
- mécanisme de transport pas défini précisemment
  ⇨ problème d'interopérabilité
- la sémantique des messages n'a pas été définie de manière formelle
  ⇨ chaque développeur a fait sa propre interprétation
- pas de performatif pour promettre/s'engager
- le choix des performatifs était peut être un peu arbitraire

FIPA : Foundation for Intelligent Physical Agents (FIPA)

structure similaire à KQML

- ensemble de performatifs différent
- possède une sémantique formelle

```
1 (inform
2   :sender agent1
3   :receiver agent5
4   :content (price good200 150)
5   :language sl
6   :ontology hpl-auction
7 )
```

Performatifs

performative	passing info	requesting info	negotiation	performing actions	error handling
accept-proposal			✔		
agree				✔	
cancel		✔		✔	
cfp			✔		
confirm	✔				
disconfirm	✔				
failure					✔
inform	✔				
inform-if	✔				
inform-ref	✔				
not-understood					✔
propose			✔		
query-if		✔			
query-ref		✔			
refuse				✔	
reject-proposal			✔		
request				✔	
request-when				✔	
request-whenever				✔	
subscribe		✔			

`Inform` et `Request`

- pré-condition : ce qui doit être vrai pour que l'acte réussisse
- "effet rationnel" : ce que l'emetteur espère provoquer

`inform` : le contenu est une déclaration

- pré-condition
  - l'emetteur pense que le contenu est vrai
  - espère que le récepteur croira le contenu
  - l'emetteur ne croit pas que le récepteur soit conscient si le contenu est vrai ou faux

`request` : le contenu est une action

- pré-condition
  - l'emetteur espère que le récepteur effectuera l'action
  - l'emetteur croit que le récepteur est capable d'effectuer l'action
  - l'emetteur ne croit pas que le récepteur ait déjà l'intention d'effectuer l'action

Autre approche : Engagement

`C(debtor, creditor, antecedent, consequent)`

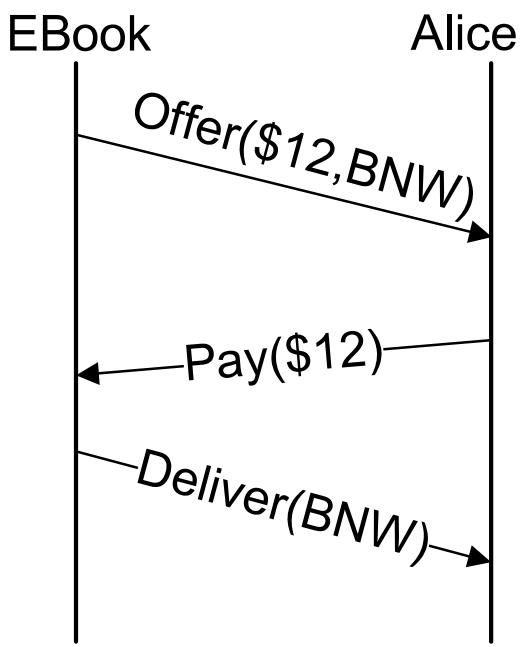
L'agent `debtor` s'engage auprès de l'agent `creditor` à ce que si l'`antecedent` est satisfait, il fera en sorte que `consequent` sera satisfait.

On peut alors créer d'autres opérations à partir de `C` :

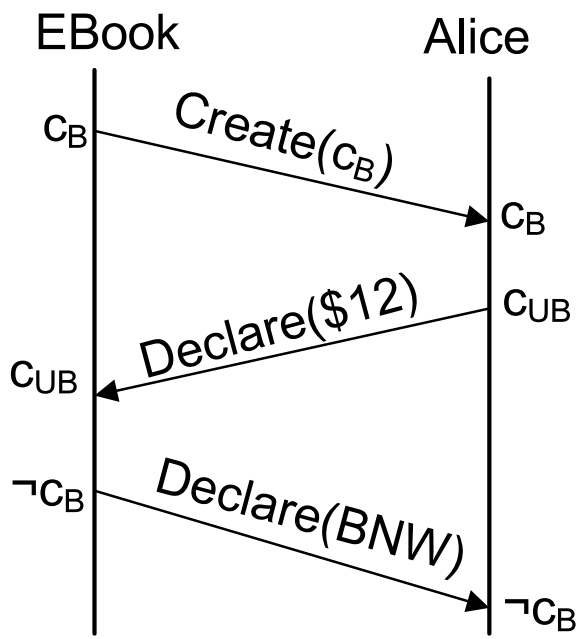
- `CREATE(x,y,r,u)` est effectué par `x` et cause `C(x,y,r,u)` d'être valable
- `CANCEL(x,y,r,u)` est effectué par `x` et cause `C(x,y,r,u)` de ne plus être valable
- `RELEASE(x,y,r,u)` est effectué par `y` et cause `C(x,y,r,u)` de ne plus être valable
- `DELEGATE(x,y,z,r,u)` est effectué par `x` et cause `C(z,y,r,u)` d'être valable
- `ASSIGN(x,y,z,r,u)` est effectué par `y` et cause `C(x,z,r,u)` d'être valable
- `DECLARE(x,y,r)` est effectué par `x` et pour informer que `r` est valable

Exemple

`Offer(mer,cus,price,item)` signifie `create(mer,cus,price,item)`
`Accept(cus,mer,price,item)` signifie `create(cus,mer,item,price)`
`Reject(cus,mer,price,item)` signifie `release(mer,cus,price,item)`
`Deliver(mer,cus,item)` signifie `declare(mer,cus,item)`
`Pay(cus,mer,price)` signifie `declare(cus,mer,price)`



Messaging



Meaning

Conclusion

- beaucoup de propositions, mais aucune n'est parfaite
- FIPA ACL est élégant, mais impossible de vérifier car il faudrait connaître l'état mental des agents
- de plus, on fait l'hypothèse que les agents sont coopératifs et bienveillants.
- les protocoles à base d'engagement sont séduisants, mais peut être pas assez expressif
- a-t-on vraiment besoin de créer un langage général ?