Modélisation des Systèmes Complexes

Systèmes Multi-Agents
Architectures d'agent

Alexandre Pauchet
INSA Rouen – Département ASI
BO.B.RC.18, pauchet@insa-rouen.fr

Caractéristiques d'un agent (rappel)

Un agent logiciel

- a un comportement et une prise de décision **autonome**
- est capable d'**interactions sociales**
- **perçoit** et **agit** sur son environnement (**caractère situé**)
- est **pro-actif** pour la réalisation de ses buts locaux (**adaptabilité**)

Développement de SMA (rappel)

Décomposition Voyelles [Demazeau, 95]

- **A**gents
- **E**nvironnements
- **I**nteractions
- **O**rganisations
- (**U**tilisateurs : agents humains dans les environnements mixtes)

Architectures d'agent (rappel)

Architecture réactive

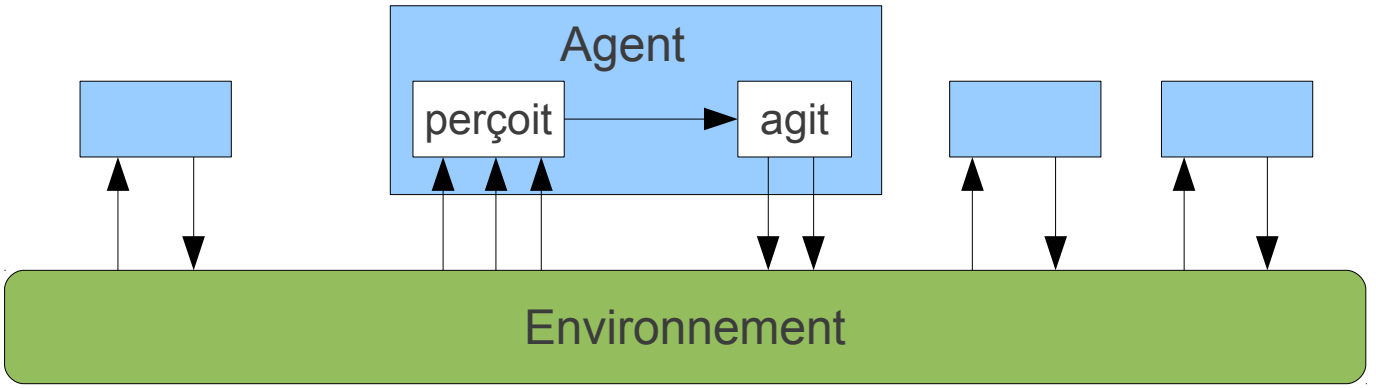
- Agit en réponse directe à une perception de son environnement (stimulus → action)
- Agents d'un grain fin, SMA à large échelle
- Souvent lié à un fonctionnement global émergent

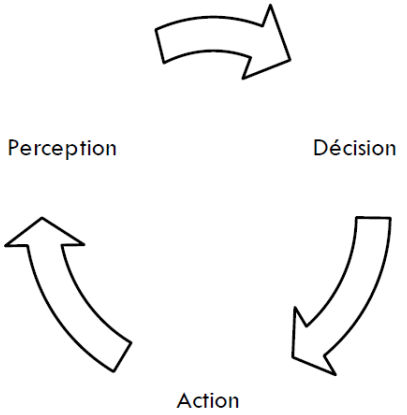
Architecture cognitive ou délibérative

- Agit en fonction d'un raisonnement interne élaboré (états mentaux → action)
- Agents de grain important, SMA de faible échelle
- Plus de contrôle sur le fonctionnement global du SMA

Agents réactifs

- Architecture simple implémentant un comportement réagissant à l'apparition d'événements
- Pas de représentation de l'environnement
- Pas d'historique des événements ou des comportements





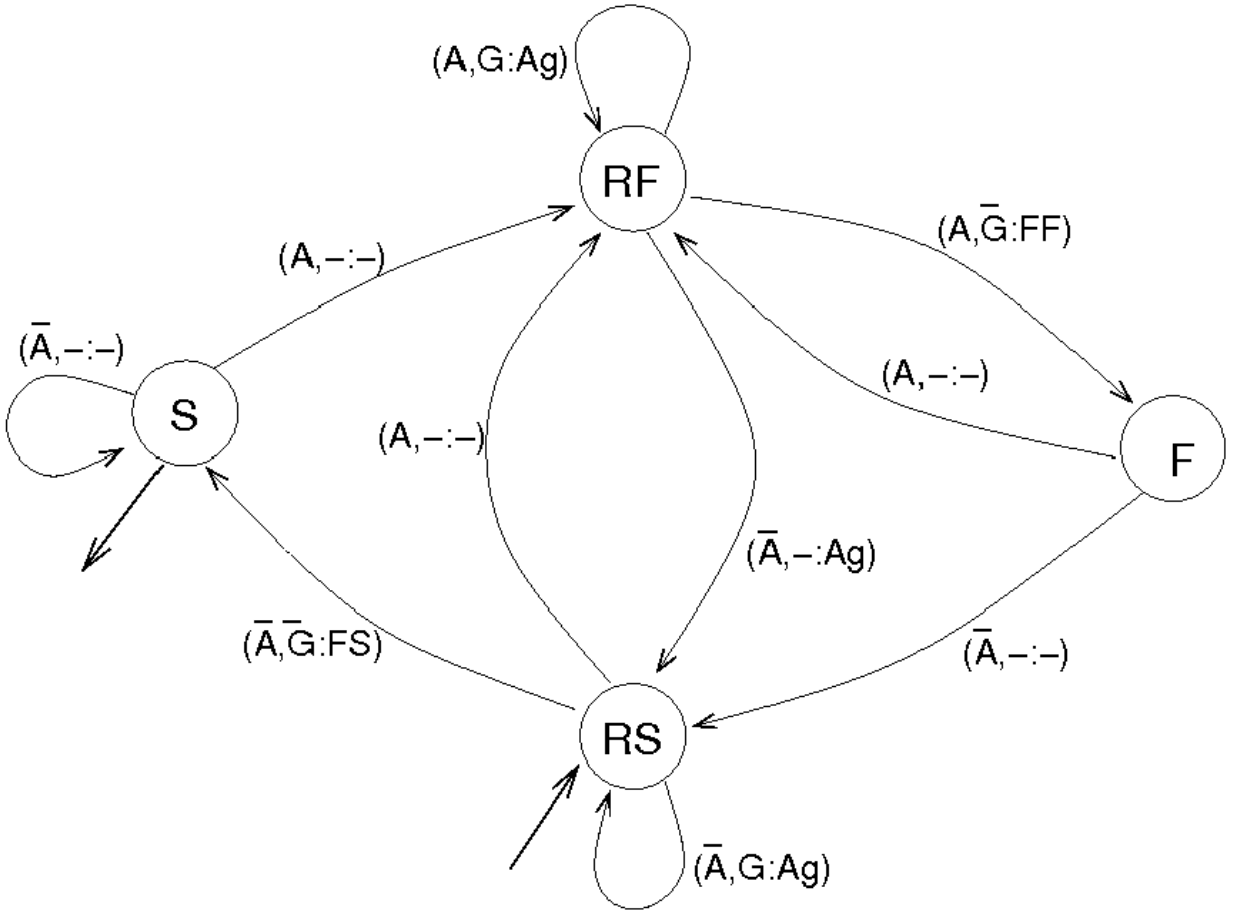
Cycle de contrôle réactif

Données manipulées

- Règles condition => action
- Ensemble de percepts

```
while (true) {
    percepts := see();
    state := interpret(percepts);
    rule := match(state,rules);
    execute(rule[action]);
}
```

Architecture d'un éco-agent (rappel)

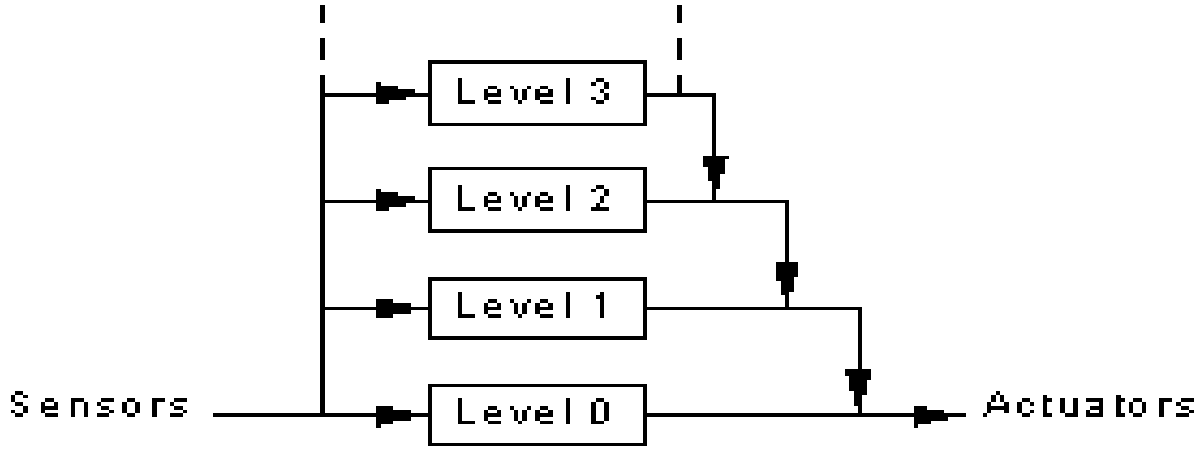


S : Satisfaction, RS : Recherche de Satisfaction, RF : Recherche de Fuite, F : Fuite

A/$\bar{A}$: Agressé / Non agressé, G/$\dot{G}$: Gêné / Non Gêné

FS : Faire Satisfaction, FF : Faire Fuite, Ag : Agresser un autre agent

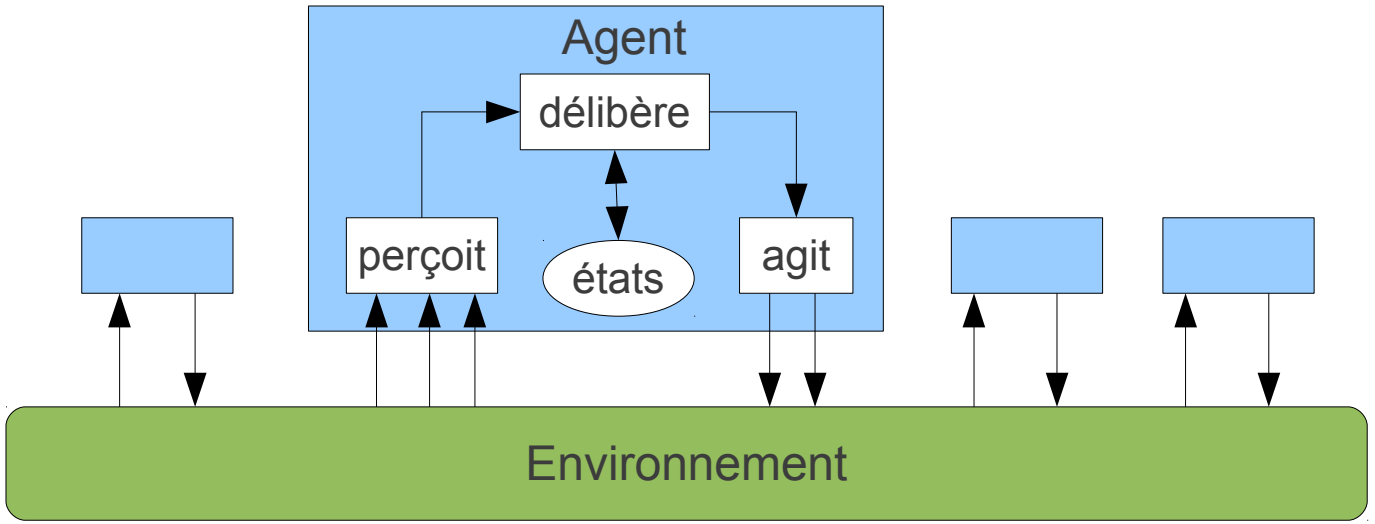
Subsomption architecture [Brooks, 96]



- Chaque couche interprète ses entrées et produit une réponse
- Possibilité de supprimer des entrées et d'inhiber des sorties

Agents cognitifs

- Le choix des actions fait l'objet d'une **délibération**
- **Représentation** explicite de l'environnement, des objectifs de l'agent et de ses capacités
- Un historique peut être utilisé pour prendre une décision, apprendre ou planifier une série d'actions



Cycle de contrôle délibératif

Données manipulées

- Ensemble d'états
- Ensemble de percepts
- Ensemble d'actions

```
states := initialise_state();
while (true) {
    percepts := see();
    states := update_states(percepts)
    action := deliberate(states);
    execute(action);
}
```

Exemple d'architecture d'agents cognitifs
Le modèle BDI

Le modèle BDI

Fondé sur un modèle cognitif de l'**intentionnalité** [Georgeff, 83] [Bratman, 90]

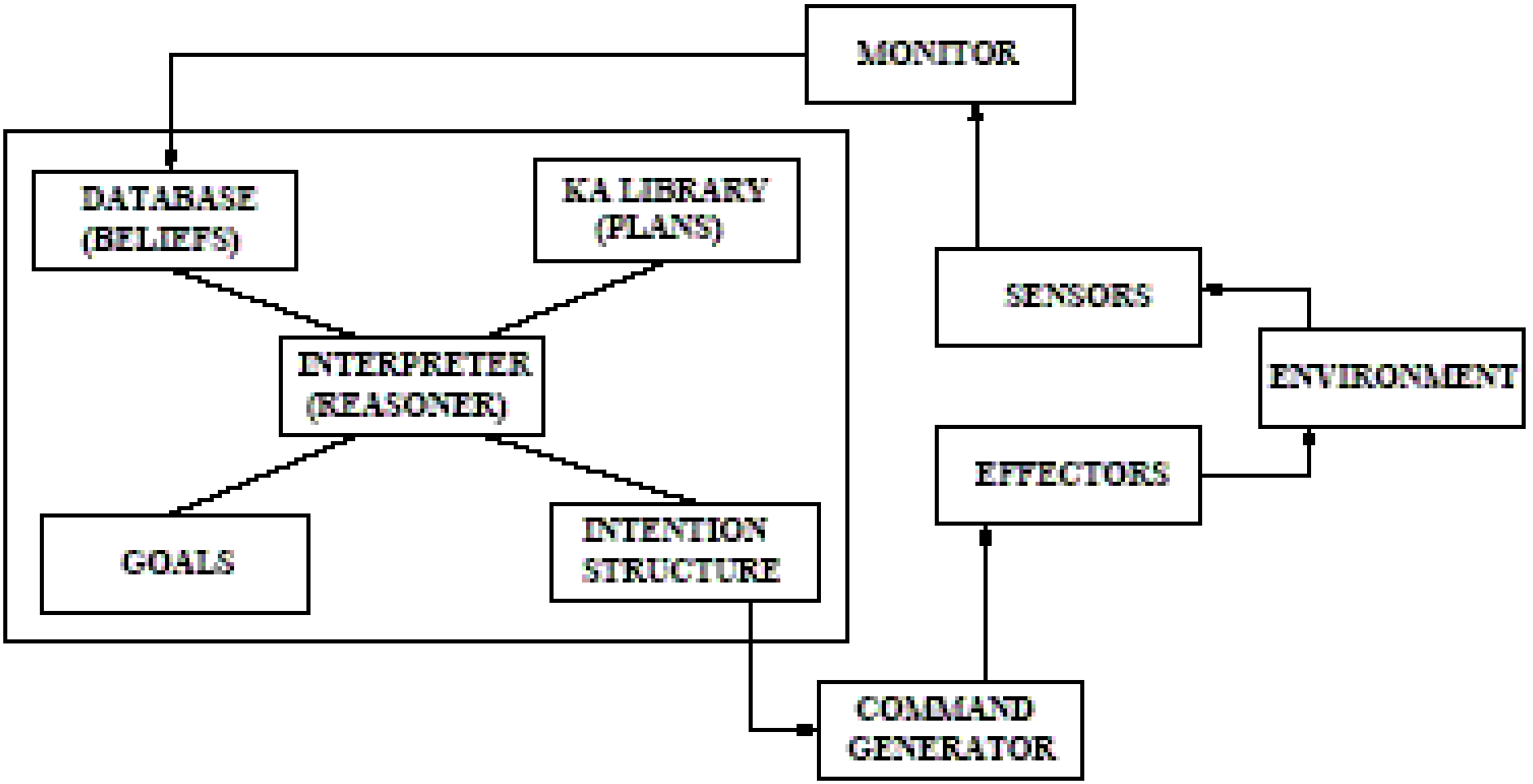
Un agent BDI est constitué

- d'un ensemble de croyances (**Beliefs**) sur lui-même et sur le monde (modalités ou prédicats)
- d'un ensemble de désirs (**Desires**) potentiellement conflictuels
- d'un ensemble d'intentions (**Intentions**) non conflictuelles
- de mécanismes de raisonnement pour la mise à jour des croyances, le choix des désirs et la génération d'intentions

Implémentations du modèle BDI

- Définition d'une architecture sur ce modèle de raisonnement (ex : PRS [Georgeff, 87])
- Formalisation en logique modale (ex : [Rao & Georgeff, 93])
- Langages de programmation d'agents (ex : Jason)

Procedural Reasoning System [Georgeff, 87]



Modèle formel BDI [Rao & Georgeff, 93]

Préambule

- un agent perçoit la situation courante comme un ***état du monde***
- un état du monde est constitué de faits vrais, faux ou inconnus représentés par des prédicats (ex : onTable(cube_A), not_clear(cube_B), ...)
- hypothèse de monde fermé vs. monde ouvert

Modèle formel BDI [Rao & Georgeff, 93]

Formules du modèles BDI

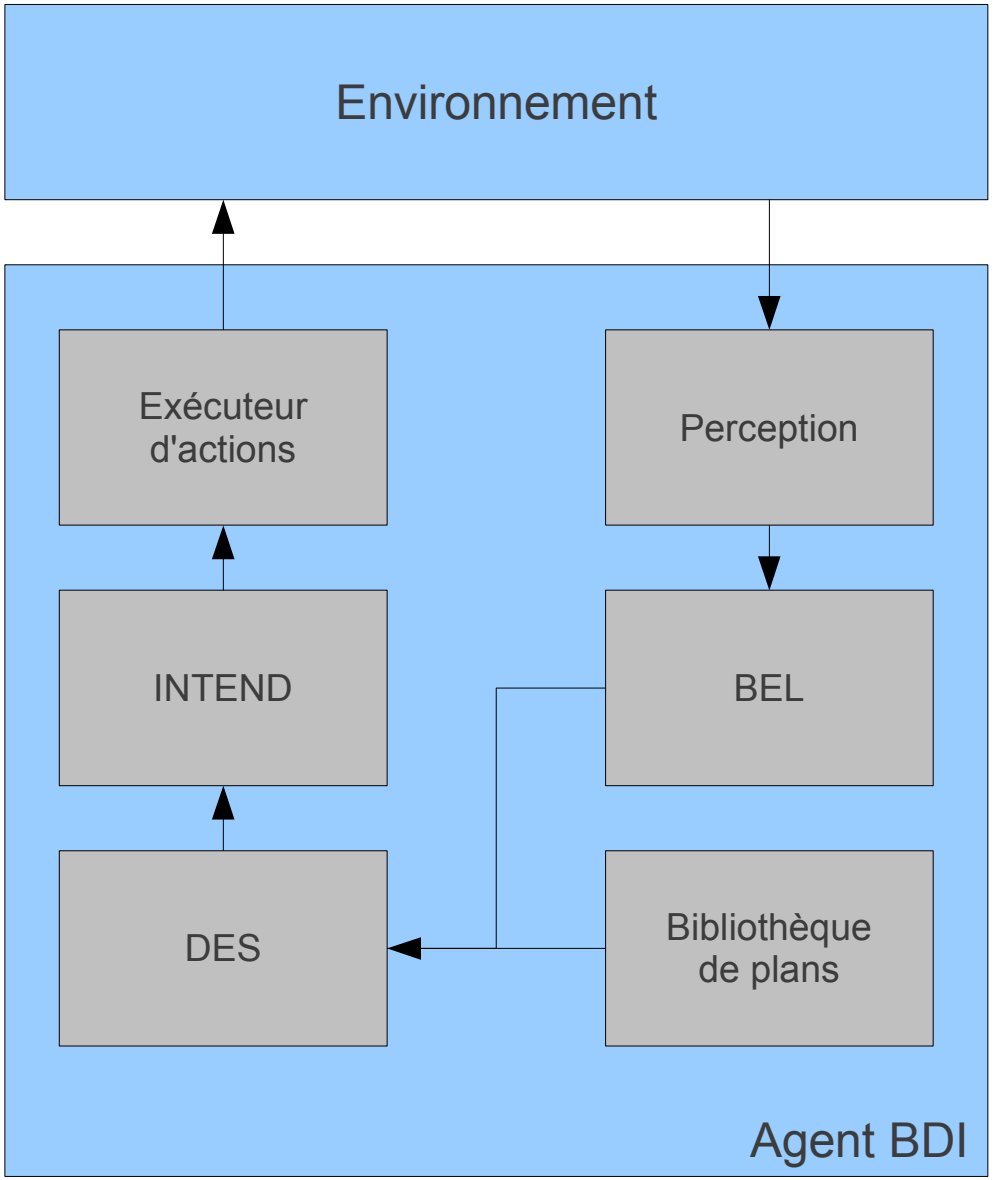
- Une formule d'état *s* est
  - Une proposition (un état du monde)
  - Une conjonction ou une négation de formules d'état (s1 AND s2, NOT s3)
  - BEL(*s*), DESIRE(*s*), INTEND(*s*)
- Une formule de chemin *p* est
  - Une formule d'état
  - Une conjonction ou une négation de formules de chemin
  - F *p*, G *p*, … des opérateurs temporels (*p* sera vrai au moins une fois, *p* sera toujours vrai, ...)

Modèle formel BDI [Rao & Georgeff, 93]

Exemples de prédicats BDI

- BEL(onTable(cube_A))
- BEL(NOT onTable(cube_A))
- INTEND(onTable(cube_B))
- BEL(does(take(?X)) AND onTable(?X) → NOT onTable(?X))
- INTEND(clear(cube_B))

Fonctionnement



Tant que (condition d'arrêt)

1. Obtenir de nouvelles perceptions
2. Mettre à jour les croyances
3. Mettre à jour la pile de désirs en fonction des plans "activables"
4. Mettre à jour la pile d'intentions
5. Exécuter la première intention de la pile d'intentions

Bibliothèque de plans

- La bibliothèque de plans encode un ensemble de sous-plans "activables" en fonction des croyances
- Un désir sera réalisé à l'aide d'un ensemble d'intentions encodées dans les sous-plans

La planification est pré-cablée !

Exemple : sous-plan pour peindre un tableau

Pre : `BEL(tableau-non-peint)`

Post : `DES(tableau-peint)`

Plan : `INTEND(prendre-pinceau) AND INTEND(tremper-pinceau) ...`

Opérations du monde des blocs

- pickup(x)

  PRE & DEL : ONTABLE(x), CLEAR(x), HANDEMPTY

  ADD : HOLDING(x)

- putdown(x)

  PRE & DEL : HOLDING(x)

  ADD : ONTABLE(x), CLEAR(x), HANDEMPTY

- stack(x,y)

  PRE & DEL : HOLDING(x), CLEAR(y)

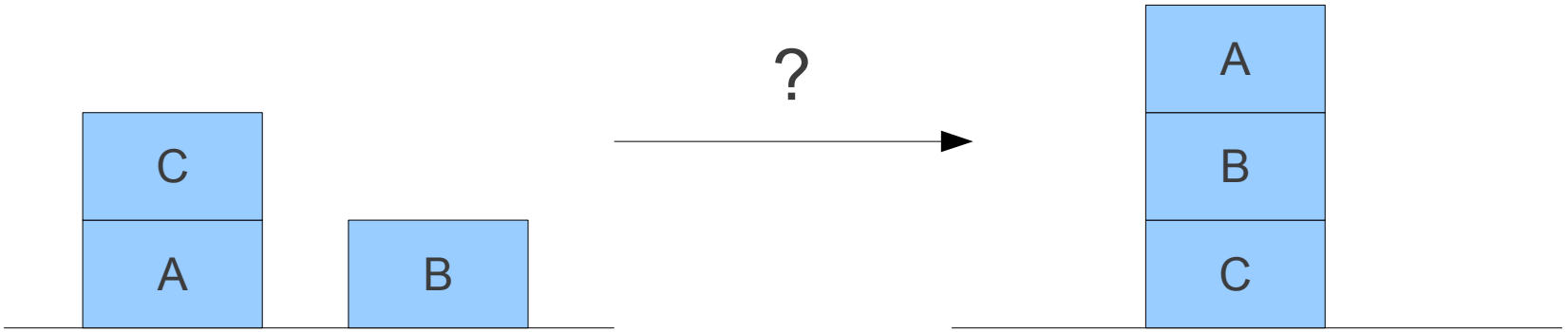
  ADD : HANDEMPTY, ON(x,y), CLEAR(x)

- unstack(x,y)

  PRE & DEL : HANDEMPTY, ON(x,y), CLEAR(x)

  ADD : HOLDING(x), CLEAR(y)

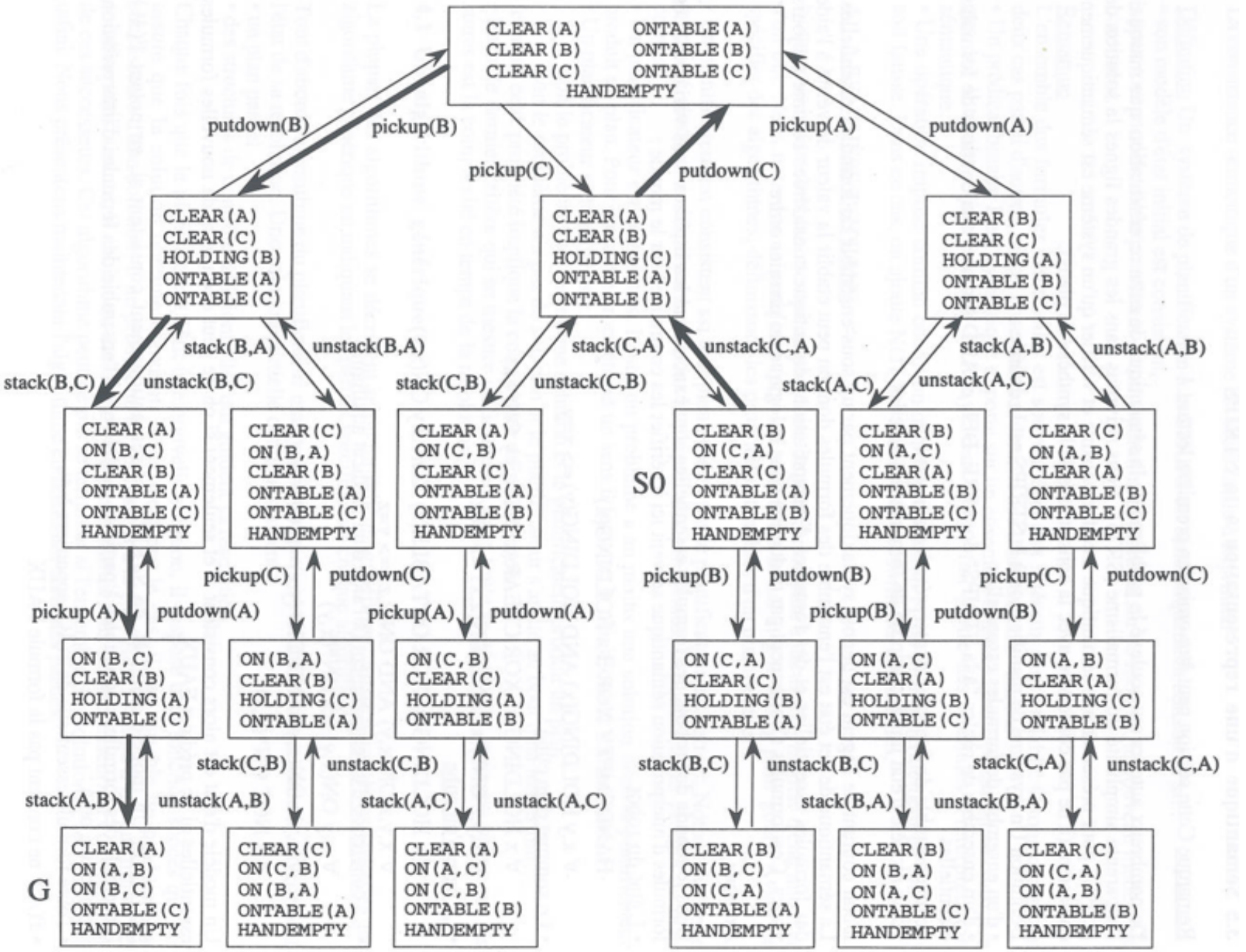
États initial et final



Une séquence possible :

unstack(C,A), putdown(C), pickup(B), stack(B,C), pickup(A), stack(A,B)

Graphe des états



Langage de programmation multi-agent

Un exemple de langage pour agents BDI : Jason

http://jason.sourceforge.net/Jason/Jason.html

- Développé en Java par R. Bordini et J. F. Hubner
- Inspiré par le formalisme AgentSpeak

Concepts de base

- Buts : *!goal*
- Actions internes : *.action(...)*
- Opérateurs d'ajout (+) et retrait (-) d'états
- Plan pour réagir à un événement : *event* <- *actions*

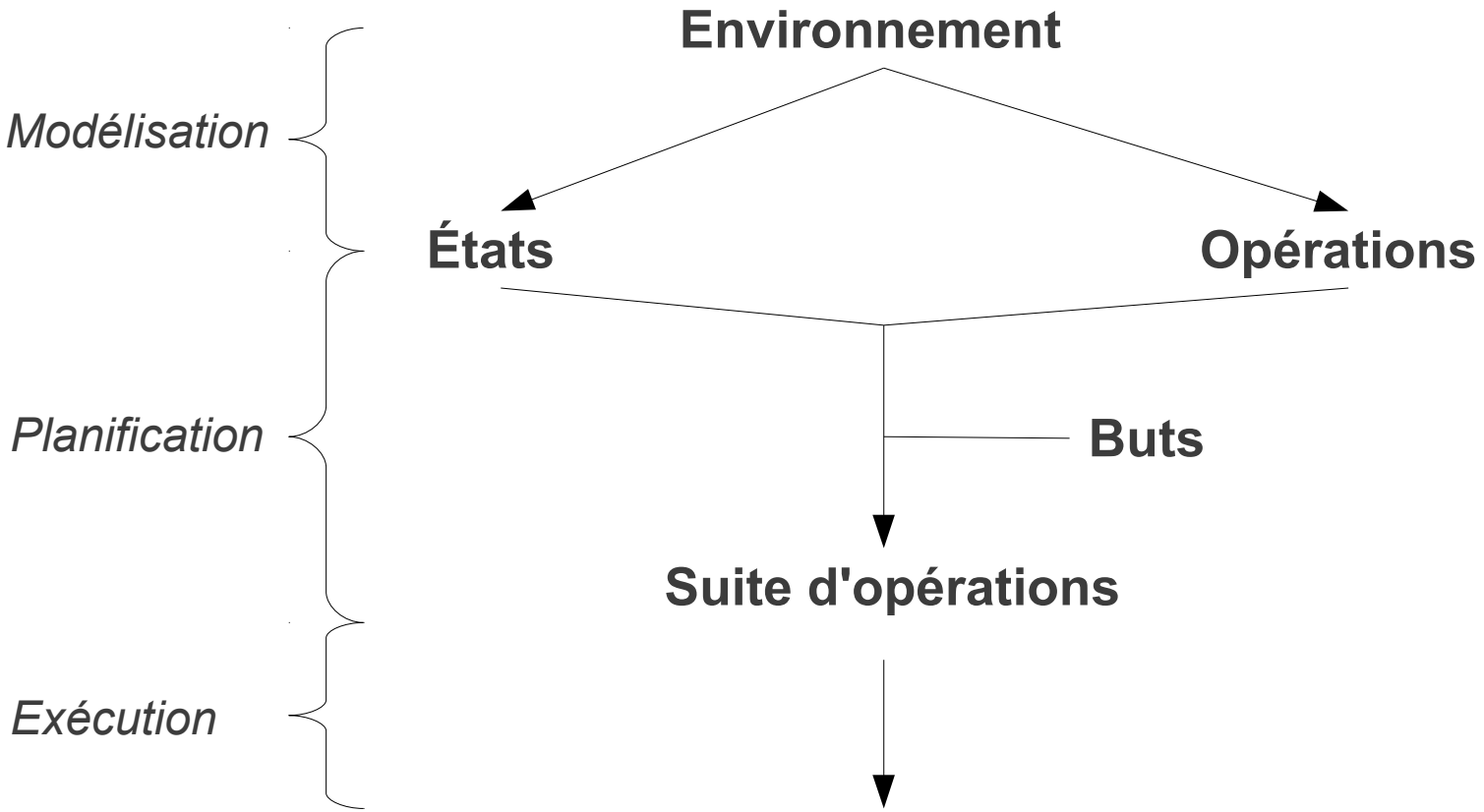
Exemple simple de code Jason

```
// Agent tom in project greeting.mas2j
!start.

+!start : true <- .send(bob,tell,hello).

+hello[source(A)]
  <- .print("I receive an hello from ",A);
     .send(A,tell,hello).
```

Planification

Planification

Déterminer une séquence d'actions à entreprendre pour atteindre un but donné



Algorithme de planification

La stratégie de parcours des solutions est implémentée ici

```
Si succès(solution initiale)
   Renvoyer(solution initiale)
Sinon
   Scons = {solution initiale}
   Saex = {solution initiale}
   TantQue Saex ≠ Ø faire
      solution = choisir(Saex)
      Si (solution_suivante = nouveau_successeur(solution) ≠ Ø) alors
         Si solution_suivante ∉ Scons alors
            Si succès(solution_suivante) alors
               Renvoyer(solution_suivante)
            FinSi
            Scons = Scons ∪ solution_suivante
            Saex = Saex ∪ solution_suivante
         FinSi
      Sinon
         Saex = Saex – {solution}
      FinSi
   FinTantQue
   Renvoyer(échec)
FinSi
```

Chaînage avant

Principe : partir de l'état initial et explorer les états produits par application successive des opérations

- Approche complète qui se termine si les modèles du monde sont finis
- Exploration en profondeur ou en largeur
- Complexité proportionnelle au nombre de modèles d'états

Chaînage arrière

Principe : partir du but pour retrouver l'état initial

Le planificateur STRIPS génère des solutions intermédiaires par 2 opérateurs

- **Décomposition** : si la solution considérée est composée, proposer des solutions dans l'ordre adéquat
- **Régression** : si la solution considérée est élémentaire, choisir une action qui y aboutit

Communications entre agents

Communiquer c'est agir !

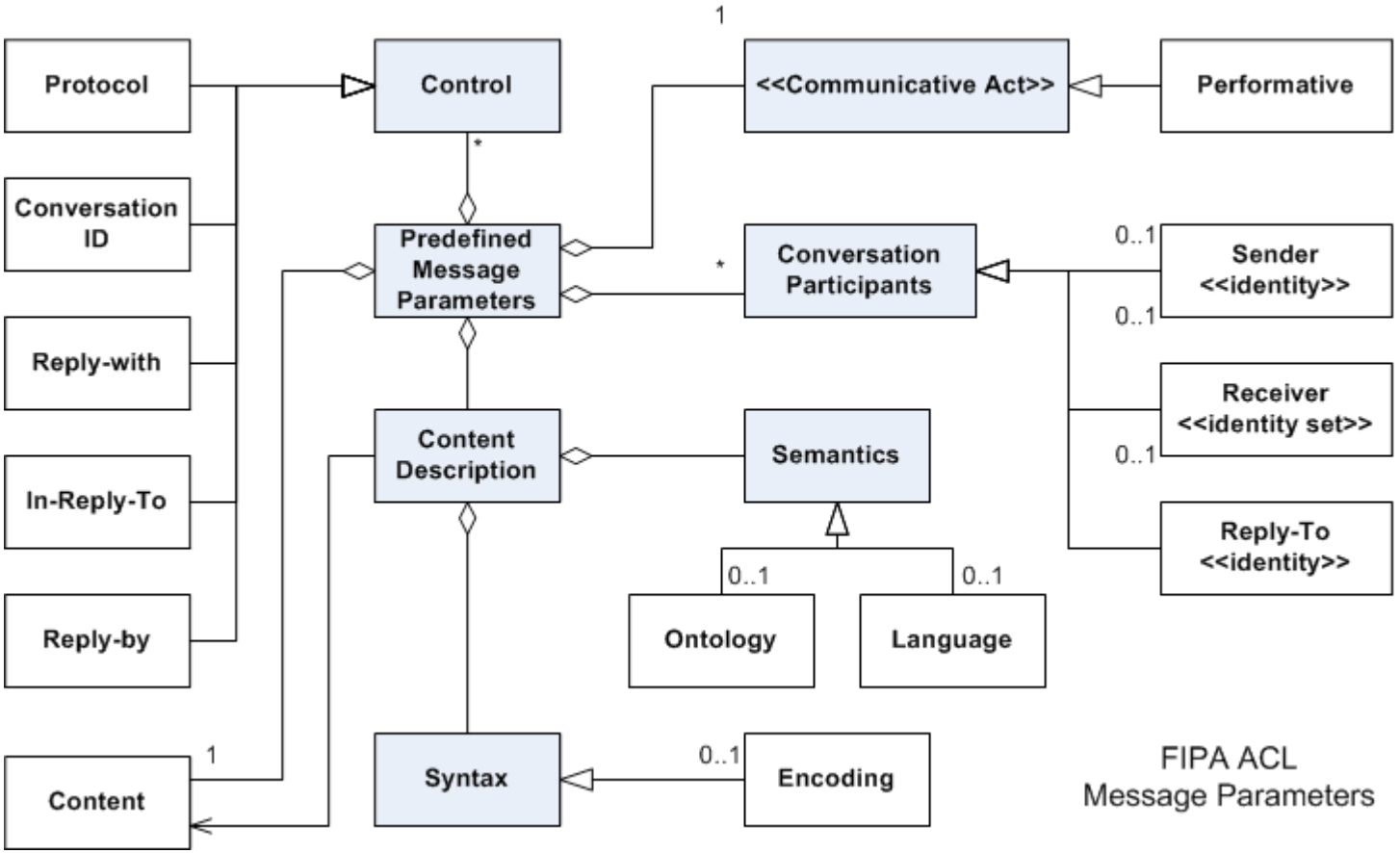
Théorie des actes de langages (Austin, Searle)

- Dimension locutoire
  - production de signes création l'acte de communiquer
- Dimension illocutoire
  - intention exprimée par le locuteur
- Dimension perlocutoire
  - effet sur l'allocutaire

Les états mentaux BDI peuvent formaliser ces dimensions

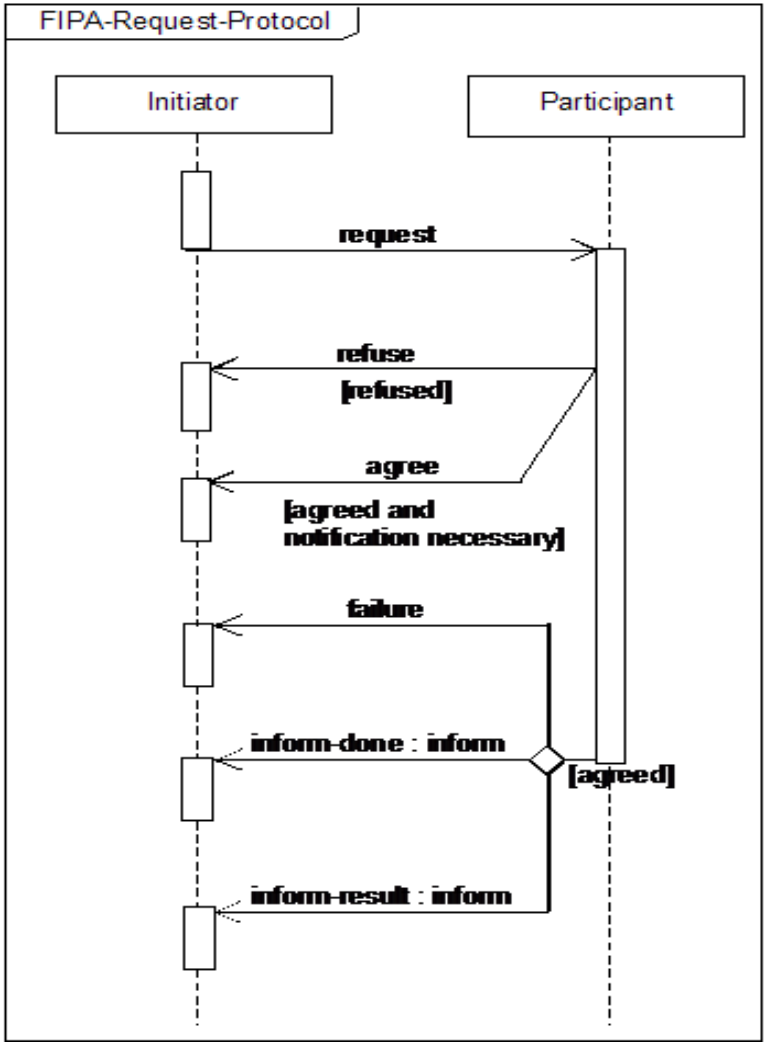
Messages FIPA-ACL

Agent Communication Language



Protocoles FIPA

FIPA Request Interaction Protocol



Exemples en FIPA-ACL

<i, inform(k,p)>

FP : $B_i p \wedge \neg B_i(\neg B_k X \vee \neg B_k \neg X \vee \neg U_k X \vee \neg U_k \neg X)$

RE : $B_k p$

<i,query-if(j,X)>

FP : $\neg B_i X \wedge \neg B_i \neg X \wedge \neg U_i X \wedge \neg U_i \neg X$

RE : Done(<j,inform(i,X)> $\vee$ <j,inform(i,¬X)>)