Intelligence Artificielle

Master 1ère année Informatique
Université Paris 7

Organisation

- `http://www.liafa.jussieu.fr/~haberm/cours/ia/`
  e-mail: Peter.Habermehl@liafa.jussieu.fr
- Contrôle des connaissances:
  Note finale première session = 2/3 Note examen + 1/3 Note contrôle continu
  Note finale deuxième session = examen
- Contrôle continu: ??
- TD: Alexandra d'Erfurth et Peter Habermehl
- Avertissement: Les transparents ne contiennent pas tout.

Bibliographie

- Ganascia, Jean-Gabriel. L'intelligence artificielle. Flammarion, 1993.
- I. Bratko, Programmation en Prolog pour l'intelligence artificielle, 2001
- J.M. Alliot et T.Schiex, Intelligence Artificielle et Informatique Théorique, Cépaduès Editions, 1993.
- N. Nilsson, Artificial Intelligence: A New Synthesis, Morgan Kaufmann, 1998.
- S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 2nd edition, 2002.

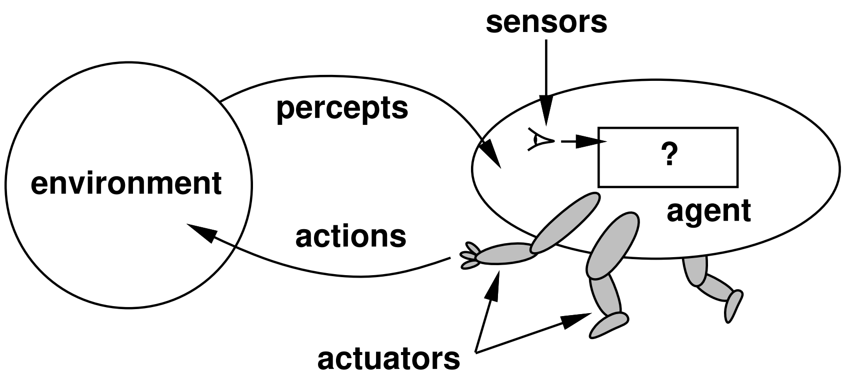
Plan du cours

- Les agents intelligents (du très simple au plus complexe)
- Les algorithmes de recherche
- Les algorithmes de jeux
- Les algorithmes d'apprentissage
  - Les algorithmes de classification
  - Les réseaux de neurones
  - L'apprentissage de langages formels

Qu'est-ce que l'Intelligence Artificielle ?

- Quatre vues:
  - Penser comme un humain
  - Agir comme un humain
  - Penser raisonnablement
  - Agir raisonnablement
- Agir raisonnablement: Faire la bonne chose
- Ici: Étudier des agents intelligents

Les agents (intelligents)

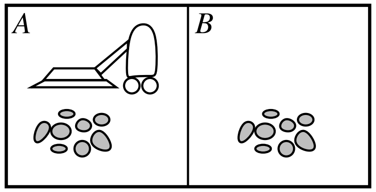


- Un agent est tout ce qu'il peut être vu comme quelque chose qui perçoit son environnement à travers de capteurs et agit sur son environnement à travers d'effecteurs.

Définitions

- Une perception est l'entrée perçu par l'agent à un moment donné.
- La séquence de perception est l'histoire complète de tout ce qu'un agent a perçu.
- Le comportement d'un agent est décrit par la fonction d'agent qui associe à chaque séquence de perception une action.
- Un programme d'agent est une implémentation de la fonction d'agent.

Exemple Agent Aspirateur



Séquence de perception	Action
[A,Propre]	Droite
[A,Sale]	Aspire
[B,Propre]	Gauche
[B,Sale]	Aspire
[A,Propre],[A,Propre]	Droite
[A,Propre],[A,Sale]	Aspire
⋮	⋮
[A,Propre],[A,Propre],[A,Propre]	Droite
⋮	⋮

Exemple Taxi automatisé

- Définir le cadre de la conception d'un agent intelligent: la description OEEP
- Objectifs: sûreté, atteindre destination, minimiser distance, maximiser profits, etc.
- Environnement: météo, routes, piétons, passager, etc.
- Effecteurs: permettent de freiner, accélérer, parler, etc.
- Perception: video, capteurs de moteurs, jauge d'essence, GPS, etc.

Agent raisonnable (doué de raison) idéal

- Les objectifs sont spécifiés par une mesure de performance donnée par une valeur numérique pour chaque passé de l'environnement.
- Action raisonnable: Action qui maximise la valeur attendu de la mesure de performance (en connaissant la séquence des perceptions jusqu'à présent et en tenant compte de la connaissance de l'agent)
- raisonnable $\neq$ omniscient
- raisonnable $\neq$ à succès
- Autonomie

Environnements

- accessible - inaccessible
- déterministe - non déterministe
- épisodique - non épisodique
- statique - semi dynamique - dynamique
- discret - continu
- agent seul - multiagent (compétitif, coopératif)

Fonction d'agent et programme

Un agent est complètement déterminé par la fonction d'agent qui étant donnée une séquence de perception renvoie une action.

Programme d'agent:

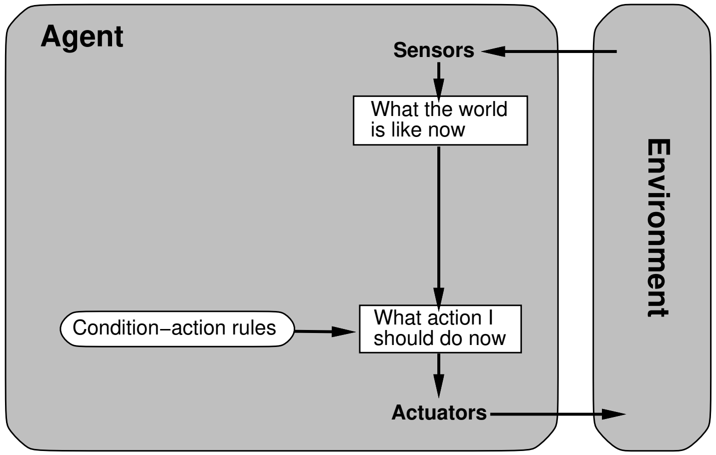
```
function SKELETON-AGENT(percept) returns action
   static: memory, the agent's memory of the world

   memory ← UPDATE-MEMORY(memory, percept)
   action ← CHOOSE-BEST-ACTION(memory)
   memory ← UPDATE-MEMORY(memory, action)
   return action
```

Le Programme le plus simple

function TABLE-DRIVEN-AGENT(*percept*) **returns** *action*
  static: *percepts*, a sequence, initially empty
    table, a table, indexed by percept sequences, initially fully specified

  append *percept* to the end of *percepts*
  action ← LOOKUP(*percepts*, *table*)
  return *action*

Agent réactif: Schéma



Agent réactif: Programme

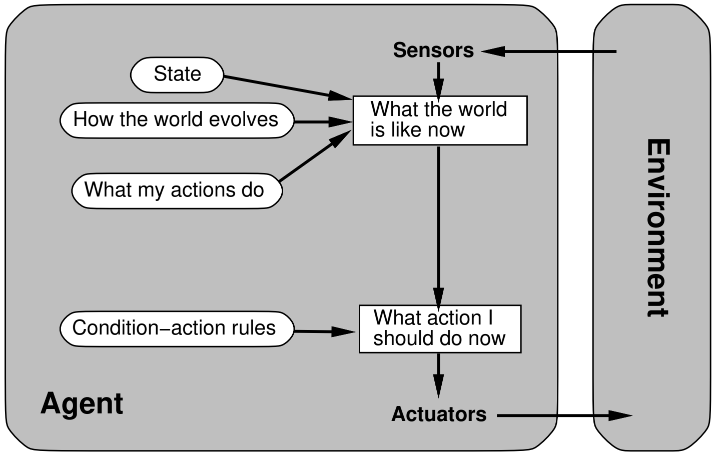
function SIMPLE-REFLEX-AGENT(*percept*) **returns** *action*
  static: *rules*, a set of condition-action rules

  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← RULE-ACTION[*rule*]
  return *action*

Programme pour l'agent aspirateur

```
function AGENT-REACTIF-ASPIRATEUR([endroit,status]) returns action

  if status = Sale alors return Aspire
  else if endroit = A then return Droite
  else if endroit = B then return Gauche
```

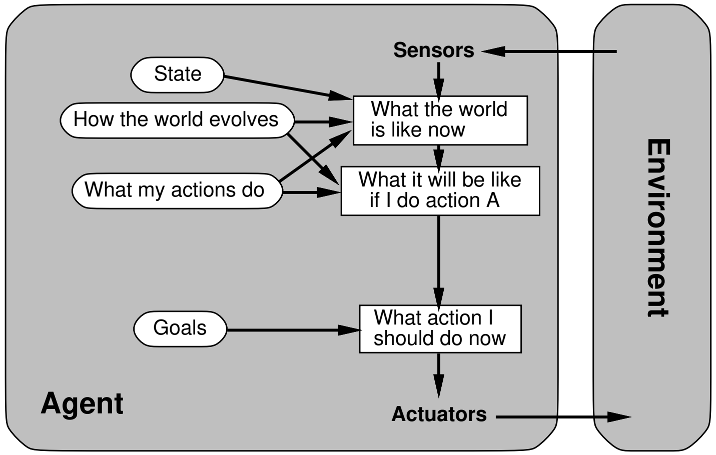

Agent réactif avec état: Schéma



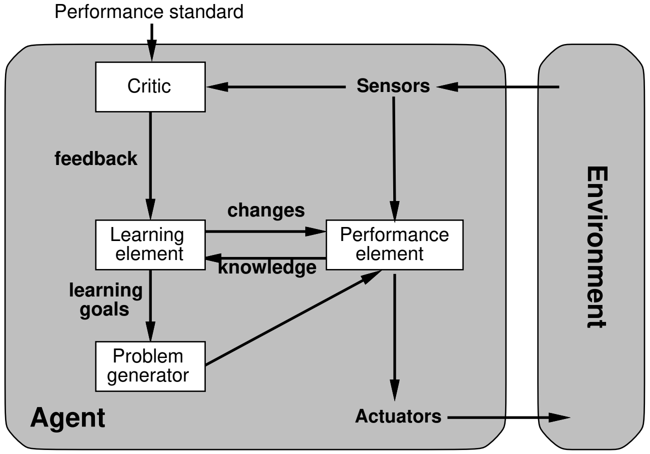
Agent réactif avec état: Programme

```
function REFLEX-AGENT-WITH-STATE(percept) returns action
   static: state, a description of the current world state
           rules, a set of condition-action rules

   state ← UPDATE-STATE(state, percept)
   rule ← RULE-MATCH(state, rules)
   action ← RULE-ACTION[rule]
   state ← UPDATE-STATE(state, action)
   return action
```


Agent avec objectifs explicites: Schéma

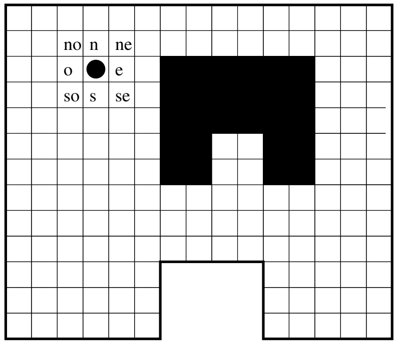


Agent qui apprend: Schéma



Exemple d'agent réactif

Contrôleur d'un robot dans un monde grillagé:



- Objectif: Le robot doit aller vers un mur et le suivre.
- Environnement: le monde grillagé
- Effecteurs: le robot peut aller dans 4 directions (ouest, est, nord, sud)
- Perception: le robot a 8 capteurs booléens (vrai indique qu'il y a un mur dans cette direction)
- À partir des valeurs des capteurs on doit décider quelle action faire.

Exemple d'agent réactif

- Pour faciliter la description des règles, on choisit d'abord des caractéristiques qu'on veut extraire des perceptions possibles. Ici, quatre valeurs booléennes données comme suit:
  - $x_1 := n \vee ne$, $x_2 := e \vee se$
  - $x_3 := s \vee so$, $x_4 := o \vee no$
- En utilisant les caractéristiques x_1, x_2, x_3, x_4 on écrit un système de productions (des règles condition-action):

$$\begin{array}{lcl} x_1\overline{x_2} & \Rightarrow & est \\ x_2\overline{x_3} & \Rightarrow & sud \\ x_3\overline{x_4} & \Rightarrow & ouest \\ x_4\overline{x_1} & \Rightarrow & nord \\ true & \Rightarrow & nord \end{array}$$

  Attention: On lit les règles du haut vers le bas.

  Dans quel sens le robot suit-il les murs ?