Quelle est la différence entre système complexe et système compliqué : Système compliqué : peut être décomposé pour être compris et étudié exemple : Moteur de voiture. Système complexe : perd son intelligibilité s’il est décomposé exemple : Ecosystème, chaîne alimentaire

Un syst compliqué, on peut le simplifier pour découvrir son intelligibilité. Un syst complexe, on doit le modéliser pour construire son intelligibilité.

Donnez trois propriétés qui distinguent un système d’intelligence classiques (tels que les systèmes experts) d’un système multi-agents : Dans les systèmes réactifs, l'intelligence est dans l'univers et non pas dans des systèmes désincarnés tels que les systèmes experts. L'intelligence émerge de l'interaction des agents avec leur environnement.

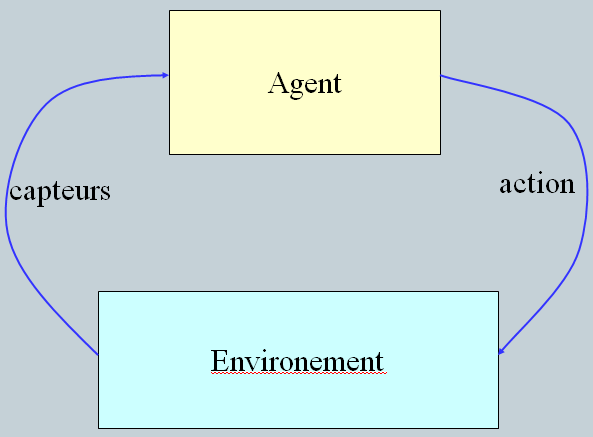
Quelle est la différence entre un système multi-bases de connaissance et un système multi-agents. système multi-bases de connaissance :

Distribuer le contrôle||Chaque agent a trois types de connaissances :C. du domaine ou expertise && C. de contrôle && C. de communication et d ’interaction

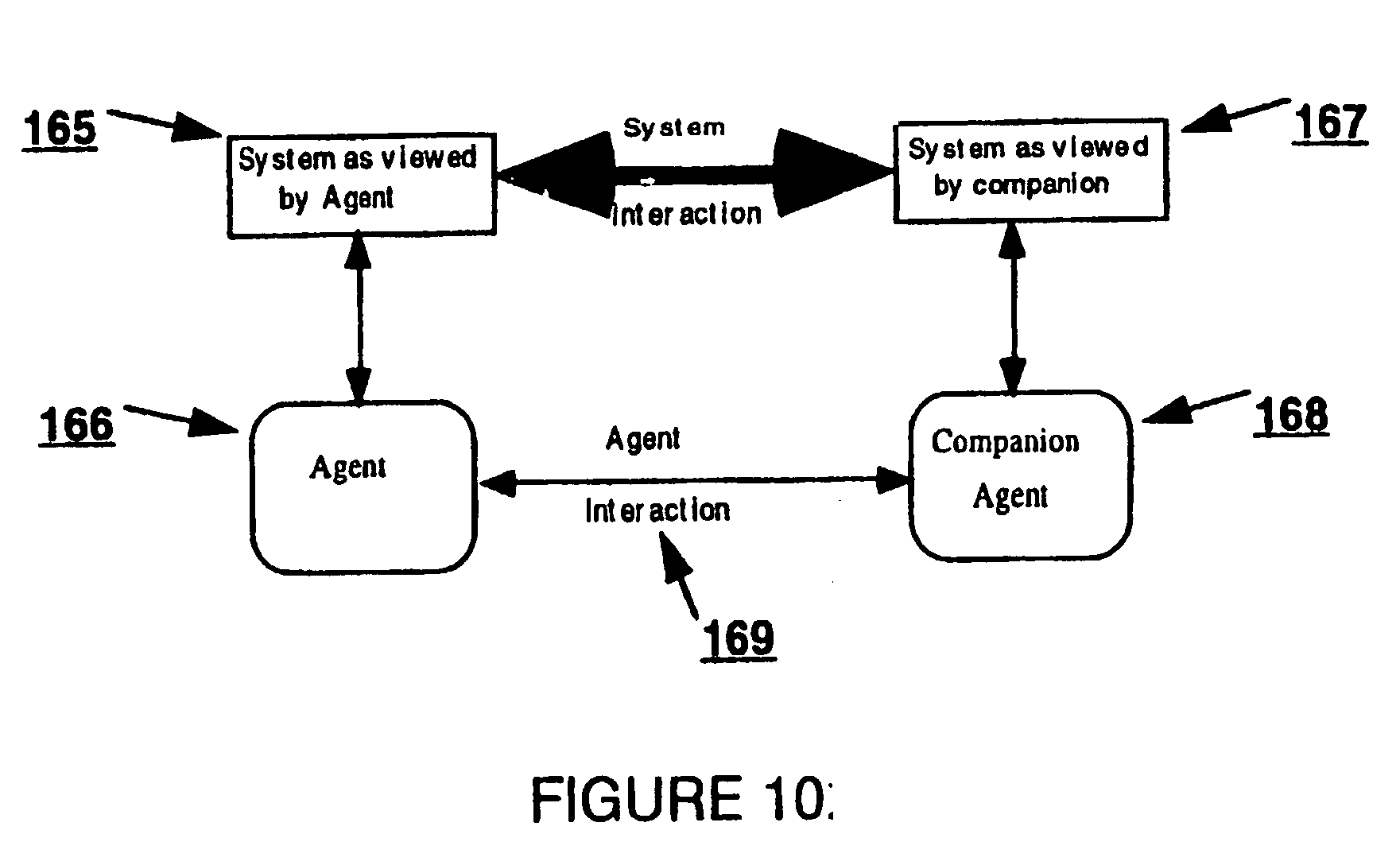
Donnez trois principales propriétés qui caractérisent un système multi-agents et qui le distinguent d’un système distribué : La cooperation / la resolution des problemes complexe/ sociable et adaptif

les logiques classiques ne sont pas appropriées dans le contexte des systèmes multi-agents : Logique classique (LC) insuffisante : En LC, la dénotation d’une expression dépend seulement de celle de ses sous expressions Ex: Jean croit p (indépendant de la valeur de p), Les agents ne s’attendent pas `a tous les eﬀets de leur intention

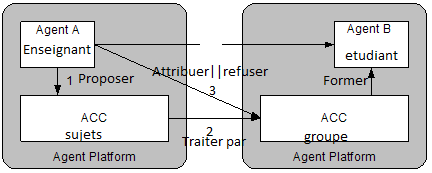
: Architecture d’agents FIPA-ACL



Décrivez la nouvelle architecture d’agents : Modèles d'agents hybrides : La nouvelle architecture permet une communication entre différente platform par le bais de ACC et l 'envoi des message par les ACL.



3. Ces agents doivent être dotés d’un interpréteur de messages FIPA-ACl. Donnez un exemple d’interpréteur (l’algorithme) pour l’architecture que vous venez de proposer : L'agent A veut informer l'agent B du temps qu'il fera demain, selon ses prévisions:   
(inform¤ :sender   (agent-identifier : name\_A)¤:receiver (set (agent-identifier :¤name\_B))¤:content¤"weather (tomorrow, raining)"¤:language Prolog)

: Mécanisme de coordination : . Modélisez ces différentes interactions en utilisant les performatifs FIPA-ACL et en choisissant un langage de représentation de connaissances :

Décrivez brièvement le comportement de chaque type d’agents : ce protocole est utilisé et normalisé par l’organisation FIPA.

Agent A: souhaitant sous traiter une tache qu’il doit accomplir et l’initiateur du contrat

AgentB : auquel on propose ce contrat

Décrivez en détail 2 exemples de messages :

L'agent A envoie à sont ACC local le message. L'ACC prend alors toutes les précautions pour envoyer le message à l'ACC distant correct en utilisant le MTP approprié. L'envoie du message à l'agent B sera alors géré par l'ACC distant.

L'agent A envoie directement son message à l'agent B en utilisant un mécanisme de transfert direct. Le transfert du message, l'adressage et les messages d'erreurs doivent alors être gérés par les agents. Ce mode de communication n'est pas couvert par la FIPA.

|  |  |
| --- | --- |
| turtles-own [energie] ;; pour savoir quand la tortue est prête  ;; pour se reproduire et quand elle doit mourir  breed [sheep a-sheep ]  to PREPARER  clear-all  prepare-patches  prepare-tortues  dessine-graphes  end    to prepare-patches  ask patches [ set pcolor green ]  end    to prepare-tortues  set-default-shape sheep "sheep"  create-sheep 10 ask turtles [ set color white set size 1.5  set label-color blue - 2  set energie random 2  setxy random-xcor random-ycor ]  end    to GO  bouge-tortues  mange-herbe  reproduit-tortue  teste-mort  repousse-herbe | dessine-graphes  end    to bouge-tortues    ask turtles [  right random 360  forward 1  set energie energie - 1 ;; quand la tortue se déplace, elle perd 1 unité d'énergie  ]  end    to mange-herbe  ask turtles [  if pcolor = green [  set pcolor black  set energie (energie + 10)  ]  ]  end    to reproduit-tortue  ask turtles [  if energie > 50 [  set energie energie  hatch 1 [ set energie energie ]  ]  ]  end    to teste-mort  ask turtles [  if energie <= 0 [ die ] ;; enlève la tortue si elle n'a plus d'énergie  ] |

|  |  |  |
| --- | --- | --- |
| globals [grass] ;; keep track of how much grass there is  breed [sheep a-sheep] ;; sheep is its own plural, so we use "a-sheep" as the singular.  breed [wolves wolf]  turtles-own [energy] ;; both wolves and sheep have energy  patches-own [countdown]  to setup  clear-all  ask patches [ set pcolor green ]  if grass? [  ask patches [  set countdown random grass-regrowth-time ;; initialize grass grow clocks randomly  set pcolor one-of [green brown]  ]  ]  set-default-shape sheep "sheep"  create-sheep initial-number-sheep ;; create the sheep, then initialize their variables  [  set color white  set size 1.5 ;; easier to see  set label-color blue - 2  set energy random (2 \* sheep-gain-from-food)  setxy random-xcor random-ycor  ]  set-default-shape wolves "wolf"  create-wolves initial-number-wolves ;; create the wolves, then initialize their variables  [  set color black  set size 2 ;; easier to see  set energy random (2 \* wolf-gain-from-food)  setxy random-xcor random-ycor  ]    set grass count patches with [pcolor = green]  reset-ticks  end | to go  if not any? turtles [ stop ]  ask sheep [  move  if grass? [  set energy energy - 1 ;; deduct energy for sheep only if grass? switch is on  eat-grass  ]  death  reproduce-sheep  ]  ask wolves [  move  set energy energy - 1 ;; wolves lose energy as they move    death  reproduce-wolves  ]  if grass? [ ask patches [ grow-grass ] ]  set grass count patches with [pcolor = green]  tick    end  to move ;; turtle procedure  rt random 50  lt random 50  fd 1  end  to eat-grass ;; sheep procedure  ;; sheep eat grass, turn the patch brown  if pcolor = green [  set pcolor brown  set energy energy + sheep-gain-from-food ;; sheep gain energy by eating  ]  end | to reproduce-sheep ;; sheep procedure  if random-float 100 < sheep-reproduce [  set energy (energy / 2)  hatch 1 [ rt random-float 360 fd 1 ]  ]  end  to reproduce-wolves ;; wolf procedure  if random-float 100 < wolf-reproduce [  set energy (energy / 2)  hatch 1 [ rt random-float 360 fd 1 ]  ]  end  to death ;; turtle procedure  ;; when energy dips below zero, die  if energy < 0 [ die ]  end  to grow-grass ;; patch procedure  ;; countdown on brown patches: if reach 0, grow some grass  if pcolor = brown [  ifelse countdown <= 0  [ set pcolor green  set countdown grass-regrowth-time ]  [ set countdown countdown - 1 ]  ]  end |