TD3 – Approche cognitiviste vs approche constructiviste

Intéressons-nous au problème suivant :

Un robot est dans une pièce et veut en sortir. Pour cela, il a la possibilité de se déplacer vers la porte et de sortir si la porte est ouverte et s'il ne tombe pas sur un obstacle ou en panne d'énergie. Il peut anticiper la panne d'énergie, en mesurant sa charge. Si la charge atteint un seuil minimal, il doit d'abord aller se recharger à sa base disponible dans la pièce. Si pendant son déplacement, il tombe sur un obstacle, il doit l'éviter. Si la porte est fermée, il doit prendre la clé sur la table. Pour prendre la clé sur la table, il doit se déplacer vers la table en évitant les obstacles et en s'assurant qu'il n'est pas en panne d'énergie.

EXERCICE 1 : GENERAL PROBLEM SOLVER

Modéliser le problème ci-dessus en utilisant GPS, i.e. définir la situation initiale (« Start »), la situation but (« Finish »), et l'ensemble des opérateurs (« ops ») qui permettent de résoudre le problème.

Pour chaque opérateur (ops), il faudra définir (« action », « precond », « adds » et « delete ») où

- « action » correspond à l'action à faire,
- « precond » correspond aux préconditions à vérifier pour pouvoir réaliser l'action,
- « add » et « delete » correspondent aux changements opérés par l'exécution de l'action sur la situation en cours (ajouts et suppression à faire sur la situation en cours pour exprimer les changements).

Dérouler ensuite à la main l'algorithme pour identifier la suite des opérations permettant de résoudre le problème.

========== Indices de correction ==========

Etat : {porte_ouverte, porte_fermee, devant_obs, en_panne, charge(X), possède_clef, devant_porte, devant_table, position(X,Y), est_dans_piece, est_dehors, direction(D)} avec D soir N, S,E,O

Etat intitial : {porte_fermee, charge(100%), position(X,Y), est_dans_piece}

Etat final : {est_dehors}

Opérateurs :

- se_charger :
  - precond : position(Xchargeur,Ychargeur)
  - delete : charge(X)
  - add : charge(100%)

- aller_vers_chargeur :
  - precond : non(devant_obs)
  - delete : position(X,Y), charge(X)
  - add : charge(X-x%), position (X+delta, X+gamme) selon direction(D) pour aller vers Xchargeur,Ychargeur
- aller_vers_porte
  - precond : non(devant_obs), charge(X) > seuil
  - delete : position(X,Y), charge(X)
  - add : charge(X-x%), position (X+delta, X+gamme) selon direction(D) pour aller vers Xporte,Yporte
- aller_vers_table
  - precond : non(devant_obs), charge(X) > seuil, non(possède_clef)
  - delete : position(X,Y), charge(X)
  - add : charge(X-x%), position (X+delta, X+gamme) selon direction(D) pour aller vers Xtable,Ytable
- eviter_obstacle
  - precond : devant_obs,
  - delete : position(X,Y), charge(X)
  - add : charge(X-x%), position (X+delta, X+gamme) selon direction(D) pour tourner de 90* à gauche
- avancer_devant // pour éviter obstacle
  - precond : non(devant_obs) , charge(X) > seuil
  - delete : position(X,Y), charge(X)
  - add : charge(X-x%), position (X+delta, X+gamme) selon direction(D) pour avancer droit devant
- prendre_clef
  - precond : devant_table , charge(X) > seuil
  - delete : charge(X)
  - add : charge(X-x%), possède_clef
- ouvrir_porte
  - precond : devant_porte , charge(X) > seuil, porte_fermee
  - delete : charge(X)
  - add : charge(X-x%), porte_ouverte
- sortir
  - precond : devant_porte , charge(X) > seuil, porte_ouverte
  - delete : charge(X), est_dans_piece
  - add : charge(X-x%), est_dehors,

EXERCICE 2 : SYSTEME A BASE DE REGLES

Exprimer ce même problème par un système à base de règles (si conditions alors actions) et donner un mécanisme de résolution pour ce système à base de règles.

========== Indices de correction ==========

Si (charge(X) < seuil) Alors aller_recharger

Si aller_recharger Alors avancer(DirectionChargeur)

Si devant_porte et porte_fermee et possedeClef Alors ouvrirPorte

Si non(possedeClef) Alors avancer(DirectionTable)

Si ouvrirPorte Alors actionnerBras

Si devant_porte et porte_ouverte Alors Avancer(D)

Si avancer(D) et obstacle Alors TournerGauche

Si avancer(D) et non(obstacle) Alors ActionnerRoueGauche et ActionnerRoueDroite

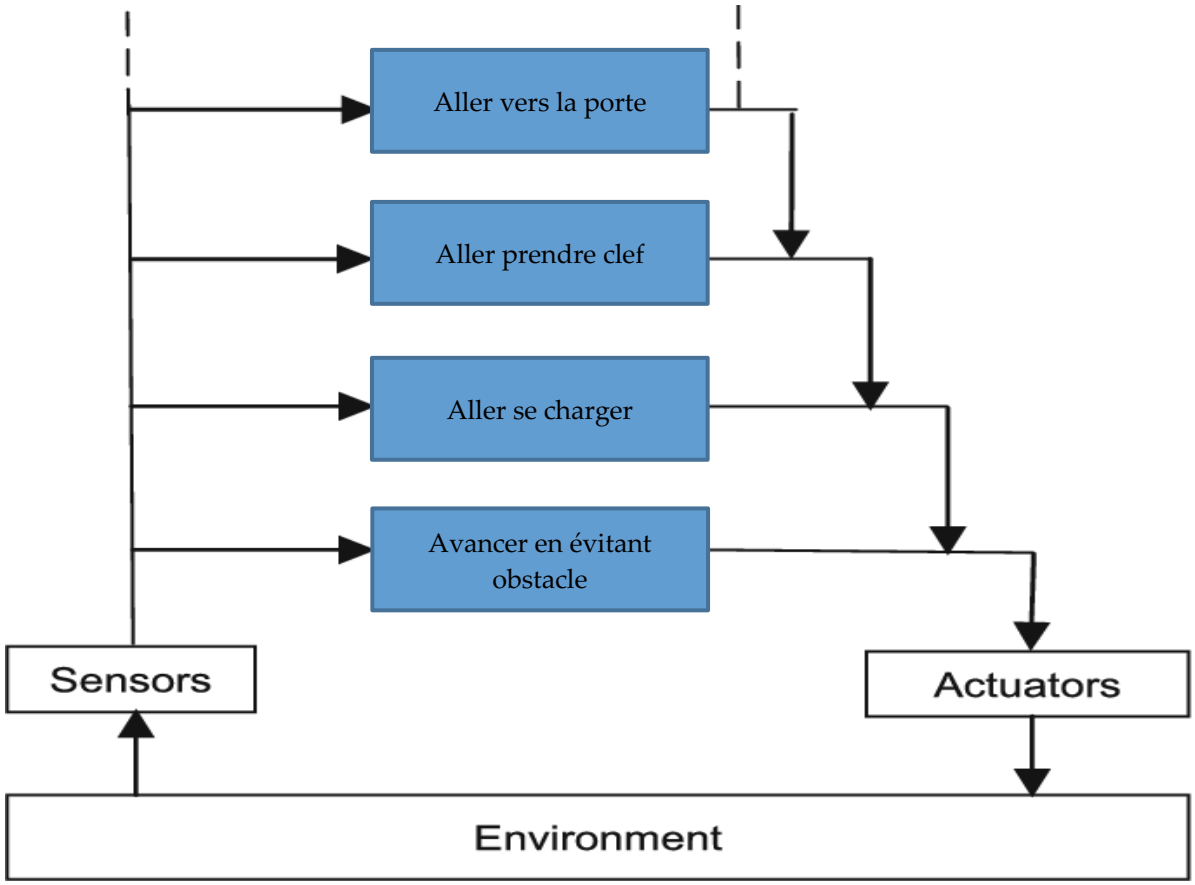
Si tournerGauche Alors ActionnerRoueDroite et avancer(D)

….

EXERCICE 3 : ARCHITECTURE DE SUBSOMPTION DE BROOKS (INTELLIGENCE WITHOUT REPRESENTATION)

Proposer une modélisation de ce même problème en suivant la proposition de Brooks (architecture de subsomption). On peut utiliser des seuils pour l'activation, inhibition d'un comportement.

========== Indices de correction ==========



EXERCICE 4 : PRENONS UN PEU DE RECUL…

Comparer et discuter les questions soulevées par chacune des approches (limites, avantages, questions soulevées).