INFORMATIQUE COGNITIVE

IFT-703

TP-2

ACT-R

Le Monde du *Wumpus*

Travail présenté à

M. André Mayers

Par

Julien Filion 07 177 770

Université de Sherbrooke, Département d'informatique

Novembre 2010

Table des matières

Introduction 1
Mémoire procédurale 1
  Description de la recherche de chemin 1
  Règle de production look-right-path 2
Mémoire déclarative 4
  Activation 4
    Activation de base 4
    Activation de propagation 5
    Association Partielle 6
    Bruit 6
  Temps de recouvrement 6
Annexe 1 : Processus de recherche de chemin 7
Annexe 2 : Trace pour retrouver un chunk EXPLORED-TILE 8
  Exemple 1 : Chunk qui correspond à la requête 8
  Exemple 2 : Chunk qui ne correspond pas à la requête 9
  Exemple 3 : Spreading activation 10

Introduction

Ce rapport est la suite d'un rapport précédant *Le Monde du Wumpus* effectué pour le cours d'Informatique cognitive. Le premier rapport étant plus général, celui-ci présente un complément d'information sur certaines parties spécifiques du projet. Ce rapport présente deux sujets principaux, la mémoire procédurale et la mémoire déclarative. Pour la mémoire procédurale, on détaille une partie du processus puis on examine une règle de production en détail. Alors que pour présenter la mémoire déclarative, on explique l'activation à l'aide d'exemple.

Mémoire procédurale

Le modèle implanté pour résoudre cette problématique possède une mémoire procédurale complexe et volumineuse. Pour cette raison, je ne vais détailler qu'une petite portion de la mémoire procédurale.

Description de la recherche de chemin

[ce paragraphe est tiré de la première version du rapport]
La recherche de chemin est une tâche qui peut être accomplie de plusieurs manières. En informatique, on utilise généralement A ou une recherche en largeur. Par contre, je ne crois pas qu'un humain utilise de telles techniques. Un humain aura tendance à diriger son attention en direction de son objectif et de revenir à des chemins qu'il n'a pas analysé lorsque son chemin est bloqué, d'une manière similaire à une recherche en profondeur.*

L'annexe 1 montre le processus suivi par l'agent pour trouver un chemin. Chaque boite rectangulaire représente une règle de production et les flèches représentent l'ordre d'exécution des procédures. Lorsqu'une règle de production X possède plusieurs flèches sortantes, cela signifie que plusieurs règles peuvent s'exécuter après l'exécution de X. Le choix de la prochaine règle est déterminé par le nouveau contexte et le système d'activation. Aussi, les rectangles qui contiennent la chaîne « <direction> » sont traduits par quatre règles de production, soit une pour chaque direction. Par exemple le rectangle *goal-reached-<direction>* représente les règles *goal-reached-right*, *goal-reached-left*, *goal-reached-up* et *goal-reached-down*. Les losanges ont été ajoutés pour simplifier le diagramme, ils permettent de limiter le nombre de flèches affiché. Une transition vers un losange peut être interprétée comme un ensemble de transition formé de toutes les transitions sortantes du losange. Finalement, les ovales représentent les processus qui précèdent et qui suivent la recherche de chemin. Ces processus ne seront pas détaillés dans ce rapport, pour plus d'information, se référer au rapport précédant.

Règle de production look-right-path

Pour mieux comprendre certaines parties du projet, je vais détailler la règle de production *look-right-path*. Cette règle est exécutée lors de la recherche de chemin, voir l'annexe 1 pour l'ordre d'exécution. Lorsque la règle peut être lancée, l'agent regarde une case à explorer pour trouver un chemin, elle sera nommée la case source. Aussi, le tampon Imaginal contient de l'information sur cette case, dont sa position. À cette étape, on cherche à savoir si les cases voisines de la case source peuvent être explorées pour la recherche de chemin. Une case peut être explorée si le joueur l'a déjà visité et si le processus de recherche de chemin ne l'a pas déjà exploré pour cette requête.

De plus, à cette étape, l'agent doit choisir dans quelle direction il porte son attention. Ce choix est fait en grâce à la sélection d'une des quatre règles de production *look-<direction>-path* en utilisant le module de résolution de conflit de ACT-R. Puisque les quatre règles ont une utilité de base de 0 et qu'aucun apprentissage n'est fait pour ce processus, le choix est fait aléatoirement. Une fois le traitement de cette direction terminé, il est possible qu'il choisisse une autre direction à traiter. Le caractère aléatoire de ce choix reflète la réalité, car un humain n'aura pas un ordre fixe d'évaluation des directions, ils seront évalués dans un ordre plus ou moins aléatoire.

En plus de fixer une direction où porter son attention, cette règle a aussi pour but de retrouver le *visual-location* à droite de la case source ce qui permettra aux règles de productions suivantes de traiter la case.

La
Figure 1 montre le code utilisée pour définir cette règle de production.

Les lignes 2 à 6 permettent de définir dans quel état doit être le buffer Goal pour que la règle puisse être lancée.

Les lignes 7 et 8 s'assurent que le buffer Visual-Location est prêt à être utilisé.

Les lignes 9 à 13 permettent d'obtenir la position horizontale (=x) et verticale (=y) de la case située à droite de la case source. J'ai dû utiliser un raccourci pour calculer la position de la nouvelle case en additionnant une constante à la position en X de la case source. Il est possible d'utiliser la fonction *:nearest* du module visuel pour corriger ce raccourci, mais des problèmes survenaient lorsqu'aucune case ne se trouve à droite.

La ligne 15 permet de conserver l'information de la case source dans le tampon Imaginal.

Les lignes 16 à 20 permettent de faire une requête au module *visual-location* pour trouver une case visitée par le joueur situé immédiatement à droite de la case source si elle existe. La ligne 20 permet de s'assurer que l'élément trouvé est une pièce déjà visitée et donc sécuritaire.

Les lignes 21 et 22 spécifient que l'agent analyse la pièce à droite, ce qui permet de sélectionner correctement les prochaines règles de productions.

```
(P look-right-path
        =goal>
                ISA             find-path
                state           searching
                right           search
                looking         nil
        ?visual-location>
                State           free
        =imaginal>
                ISA             Explored-tile
                x               =from-x
                y               =y
        !bind!                  =x (+ =from-x *grid-space*)
==>
        =imaginal>
        +visual-location>
                ISA             visual-location
                screen-x        =x
                screen-y        =y
                - value         "X"
        =goal>
                looking         right
)
```

Figure 1 : Règle de production *look-right-path*

Mémoire déclarative

La mémoire déclarative de l'agent contient trois catégories de chunk-type. Les chunks qui représentent les buts, les chunks qui représentent des données recueillies et des chunks qui représentent des valeurs de base (par exemple *right*, *done*, *start*, etc.). Voici un exemple de chunk-type de donné :

(chunk-type explored-tile location x y from-x from-y from-dir iteration world)

Ce type est utilisé lors du processus de recherche de chemin, il exprime une pièce explorée par le processus. La variable *location* pointe sur le *visual-location* de la pièce. Les variables *x* et *y* contiennent la position de la pièce. Les variables *from-x*, *from-y* et *from-dir* sont utilisées pour générer le chemin lorsque la recherche de chemin trouve la pièce voulue. Finalement les variables *iteration* et *world* spécifient dans quel contexte le chunk a été créé.

Activation

Pour ce modèle, le calcul de l'activation se fait en considérant l'activation de base, l'activation de propagation (spreading activation), l'activation d'association partielle et le bruit. De plus, l'Annexe 2 comporte des exemples de traces produites par ACT-R générés par la règle de production <*direction*>*-path-found* avec la requête suivante :

```
+retrieval>
        ISA             explored-tile
        x               =x
        y               =y
        iteration       =it
        world           =world
```

De plus le paramètre :er est activé pour permettre l'insertion de bruit aléatoire et le paramètre :esc est vrai pour activé le calcul subsymbolique.

Activation de base

Pour ce modèle, le paramètre *optimized learning* (:ol) est vrai, le paramètre *base level constant* (:blc) vaut 5 et le paramètre *base level learning* (:bll) vaut 0.9. L'équation utilisée pour le calcul de l'activation de base est donc le suivant :

$$B_i = \ln(n/(1-d)) - d * \ln(L) + \beta_i$$

Où β_i représente la valeur de :blc et *d* représente la valeur de :bll. Les paramètres :blc et :bll ont été choisie empiriquement pour produire un modèle similaire au comportement humain. Tous les chunks du modèle possèdent la même valeur pour β_i.

L'activation de base ne dépend pas du contexte, ni de la requête, c'est pourquoi les exemples 1 et 2 de l'Annexe 2 ont des valeurs d'activations de bases élevées, bien qu'un des deux chunks ne répond pas à la requête.

Activation de propagation

L'activation de propagation est activée grâce au paramètre :mas qui vaut 5.

Pour calculer l'activation de propagation S_i d'un chunk *i* on somme sur tous slots de tous les buffers. Si un slots *j* est le même que le chunk *i* ou que *j* est dans un des slots du chunk *i*, alors on utilise la formule suivante pour le calcul. Dans le cas inverse, l'activation de propagation est de 0.

$$S_i = \sum_k \sum_j W_{kj} S_{ji}$$

$$S_{ji} = S - \ln(fan_{ji})$$

$$fan_{ji} = \frac{1 + slots_j}{slotsof_{ji}}$$

Où la valeur de *S* correspond au paramètre :mas, $slots_j$ est le nombre de slots pour lesquels *j* est la valeur et $slotsof_{ji}$ est le nombre de slots de *i* qui ont la valeur *j*. De plus, W_{kj} représente la fraction du poids associé à chaque slot *kj*. Pour mon agent, seul le tampon Goal possède un poids non-nul.

On peut voir un exemple de l'activation de propagation à l'exemple 3 de l'annexe 1.

Dans l'exemple 1, la valeur de l'activation de propagation est de zéro, car le chunk *i* ne possède aucun chunk faisant partie des slots du tampon Goal. Par Contre, le chunk *i* possède un slot avec la valeur VISUAL-LOCATION2-0-0 et le tampon Goal possède un slot avec la valeur VISUAL-LOCATION2-0-3. Ces deux valeurs représentent le même visual-location mais n'ont pas été instanciées au même moment. Il aurait été intéressant de pouvoir les considérer comme équivalent pour avoir une valeur de propagation positive.

Un problème peut survenir avec cette formule si *S* est trop petit. En effet il est possible d'avoir des valeurs négatives si $\ln(fan_{ji})$ est plus grand que *S*. Ce qui signifie qu'un chunk peut se voir pénalisé s'il possède un slot égal à un slot d'un buffer. Par exemple, pour un *S* de 5, l'activation peut devenir négative si un chunk est la valeur de plus de 147 slots. Par contre pour un *S* de 2 ce nombre passe à 6.

Association Partielle

L'association partielle est activée grâce au paramètre :mp qui a une valeur de 10. La formule utilisée pour ce calcul est le suivant :

$$P_i = \sum_k PM_{ki}$$

Où *P* prend la valeur du paramètre :mp et M_{ki} est la mesure de similarité entre deux valeurs. Je n'ai pas modifié la mesure de similarité de ACT-R, donc la similarité est de 0 si les chunk sont égal et de -1 sinon. Par contre, il aurait été intéressant de modifier la similarité pour la position des pièces, ce qui permettrait d'activer des chunk représentant des pièces près de la requête.

L'exemple 1 possède une association partielle de 0, car le chunk correspond à la requête. Par contre, l'exemple 2 possède une association partielle de -10, car une des valeurs, la valeur de Y, ne correspond pas à la requête.

Bruit

Finalement, le bruit temporaire est activé grâce au paramètre :ans avec une valeur de 0.5. Ce qui correspond à une fonction logistique avec un écart-type d'environ 0.91. Le bruit permanent est aussi activé avec une valeur de 0.5 (paramètre :pas).

Temps de recouvrement

Pour calculer le temps de recouvrement, le paramètre :lf à été modifié à 0.4 plutôt que 1 pour réduire le temps de recouvrement.

Voici la formule utilisée pour calculer le temps de recouvrement d'un chunk :

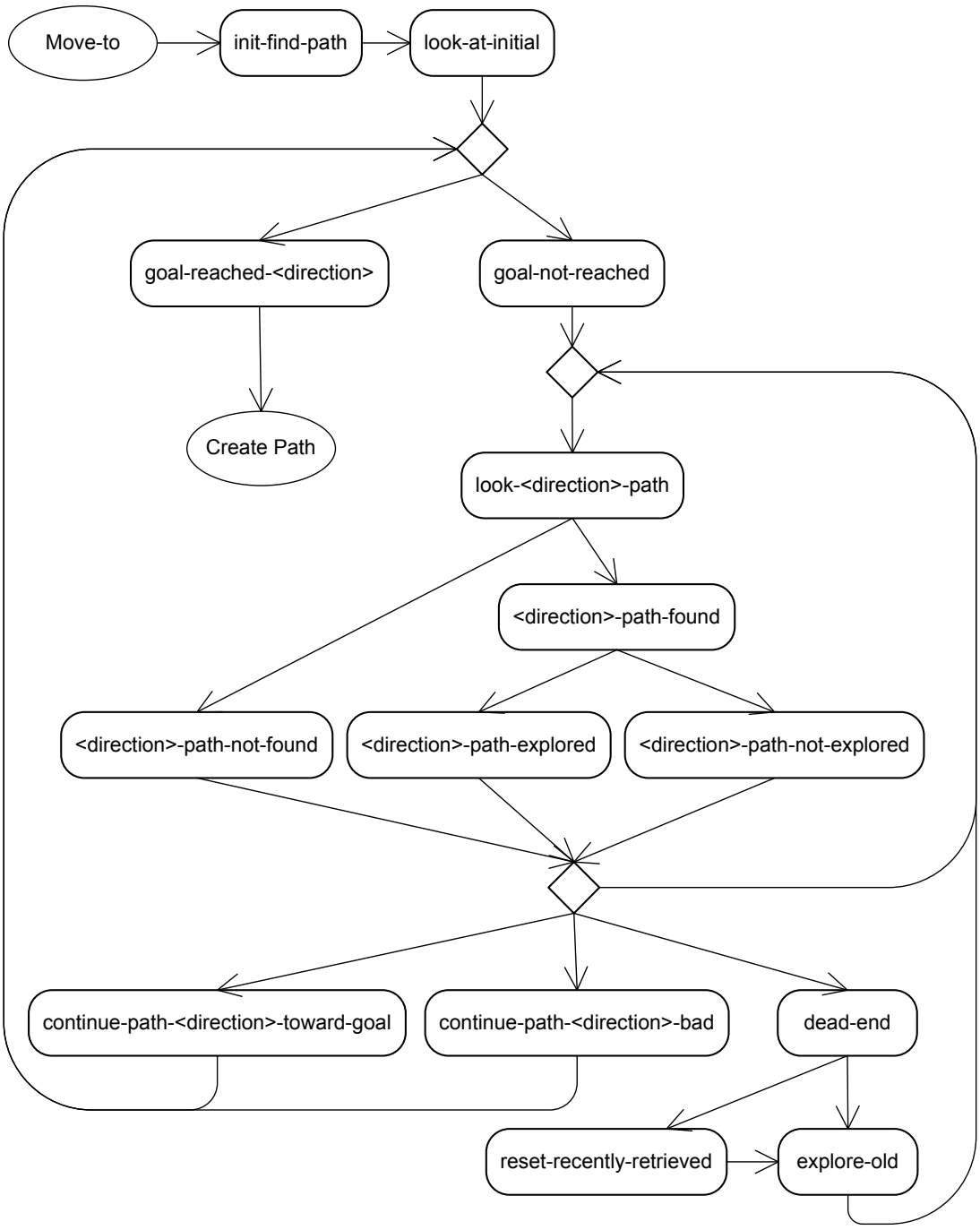
$$RT = Fe^{-(f*A_i)}$$

Si aucun chunk n'est trouvé, la formule suivante est utilisée :

$$RT = Fe^{-(f*\tau)}$$

La valeur par défaut de τ est de 0, ce qui implique que l'agent sait immédiatement si aucun chunk ne répond à sa requête, ce qui ne me semble pas réaliste pour mon modèle. C'est pourquoi j'ai modifié sa valeur à 1 à l'aide du paramètre :rt.

Annexe 1 : Processus de recherche de chemin



Annexe 2 : Trace pour retrouver un chunk EXPLORED-TILE

Exemple 1 : Chunk qui correspond à la requête

```
Computing activation for chunk EXPLORED-TILE2-0
Computing base-level
Starting with blc: 5.0
Computing base-level from 3 references NIL
  creation time: 2.6 decay: 0.9  Optimized-learning: T
base-level value: 2.3037786
Total base-level: 7.3037786
Computing activation spreading from buffers
  Spreading 1.0 from buffer GOAL chunk FIND-0
    sources of activation are: (SEARCHING
                                NOT-EXPLORED
                                SEEN
                                EXPLORED
                                NOTHING
                                VISUAL-LOCATION16-0-0
                                VISUAL-LOCATION2-0-3
                                VISUAL-LOCATION7-0-0
                                VISUAL-LOCATION9-0-1
                                LEFT)
    Spreading activation  0.0 from source SEARCHING level  0.1 times Sji 0.0
    Spreading activation  0.0 from source NOT-EXPLORED level  0.1 times Sji 0.0
    Spreading activation  0.0 from source SEEN level  0.1 times Sji 0.0
    Spreading activation  0.0 from source EXPLORED level  0.1 times Sji 0.0
    Spreading activation  0.0 from source NOTHING level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION16-0-0 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION2-0-3 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION7-0-0 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION9-0-1 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source LEFT level  0.1 times Sji 0.0
Total spreading activation: 0.0
Computing partial matching component
  comparing slot X
  Requested: = 35  Chunk's slot value: 35
  similarity: 0.0
  effective similarity value is 0.0
  comparing slot Y
  Requested: = 80  Chunk's slot value: 80
  similarity: 0.0
  effective similarity value is 0.0
  comparing slot ITERATION
  Requested: = 0  Chunk's slot value: 0
  similarity: 0.0
  effective similarity value is 0.0
  comparing slot WORLD
  Requested: = 0  Chunk's slot value: 0
  similarity: 0.0
```

```
  effective similarity value is 0.0
Total similarity score 0.0
Adding transient noise -0.41238248
Adding permanent noise -0.418
Chunk EXPLORED-TILE2-0 has an activation of: 6.473396
Chunk EXPLORED-TILE2-0 is now the current best with activation 6.473396
```

Exemple 2 : Chunk qui ne correspond pas à la requête

```
Computing activation for chunk EXPLORED-TILE0-0
Computing base-level
Starting with blc: 5.0
Computing base-level from 2 references NIL
  creation time: 0.6 decay: 0.9  Optimized-learning: T
base-level value: 1.4804764
Total base-level: 6.4804764
Computing activation spreading from buffers
  Spreading 1.0 from buffer GOAL chunk FIND-0
    sources of activation are: (SEARCHING
                                NOT-EXPLORED
                                SEEN
                                EXPLORED
                                NOTHING
                                VISUAL-LOCATION16-0-0
                                VISUAL-LOCATION2-0-3
                                VISUAL-LOCATION7-0-0
                                VISUAL-LOCATION9-0-1
                                LEFT)
    Spreading activation  0.0 from source SEARCHING level  0.1 times Sji 0.0
    Spreading activation  0.0 from source NOT-EXPLORED level  0.1 times Sji 0.0
    Spreading activation  0.0 from source SEEN level  0.1 times Sji 0.0
    Spreading activation  0.0 from source EXPLORED level  0.1 times Sji 0.0
    Spreading activation  0.0 from source NOTHING level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION16-0-0 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION2-0-3 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION7-0-0 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION9-0-1 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source LEFT level  0.1 times Sji 0.0
Total spreading activation: 0.0
Computing partial matching component
  comparing slot X
  Requested: = 35  Chunk's slot value: 35
  similarity: 0.0
  effective similarity value is 0.0
  comparing slot Y
  Requested: = 80  Chunk's slot value: 50
  similarity: -1.0
  effective similarity value is -10.0
  comparing slot ITERATION
  Requested: = 0  Chunk's slot value: 0
```

```
  similarity: 0.0
  effective similarity value is 0.0
  comparing slot WORLD
  Requested: = 0  Chunk's slot value: 0
  similarity: 0.0
  effective similarity value is 0.0
Total similarity score -10.0
Adding transient noise -0.19897808
Adding permanent noise 1.619
Chunk EXPLORED-TILE0-0 has an activation of: -2.0995018
Chunk EXPLORED-TILE0-0 has the current best activation -2.0995018
```

Exemple 3 : Spreading activation

```
Computing activation spreading from buffers
  Spreading 1.0 from buffer GOAL chunk FIND-0
    sources of activation are: (SEARCHING
                                EXPLORED
                                SEARCH
                                SEARCH
                                SEEN
                                VISUAL-LOCATION38-0-0
                                VISUAL-LOCATION24-0-0
                                VISUAL-LOCATION1-0-2
                                VISUAL-LOCATION25-0-0
                                DOWN)
    Spreading activation  0.0 from source SEARCHING level  0.1 times Sji 0.0
    Spreading activation  0.0 from source EXPLORED level  0.1 times Sji 0.0
    Spreading activation  0.0 from source SEARCH level  0.1 times Sji 0.0
    Spreading activation  0.0 from source SEARCH level  0.1 times Sji 0.0
    Spreading activation  0.0 from source SEEN level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION38-0-0 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION24-0-0 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION1-0-2 level  0.1 times Sji 0.0
    Spreading activation  0.0 from source VISUAL-LOCATION25-0-0 level  0.1 times Sji 0.0
    Spreading activation  0.28027755 from source DOWN level  0.1 times Sji 2.8027754
Total spreading activation: 0.28027755
```