

RAPPORT IA02 : Le Motus

DOAN Nhat-Minh GI01

Juin 2019

Table des matières

1	Introduction	2
2	Explication du programme	2
2.1	Initialiser des variables d'environnement du compilateur GNU Prolog sous Windows 10	2
2.2	Listes des prédicats built-in	3
2.3	Listes des prédicats définis	3
2.4	Fonctionnement du code	6
2.4.1	Mode d'humain	6
2.4.2	Mode d'ordinateur	6
2.5	Mode d'emploi	6
2.6	Résultat	6
2.6.1	Mode d'humain	6
2.6.2	Mode d'ordinateur	7
2.6.3	Performance	9
3	Conclusion	10
4	Rendu	10

1 Introduction

Le but de ce projet est de réaliser le jeu motus avec le langage prolog. Le jeu contient deux modes : mode d'humain et mode d'ordinateur. Le mot est lit depuis le clavier par saisir un *atom*.

2 Explication du programme

2.1 Initialiser des variables d'environnement du compilateur GNU Prolog sous Windows 10

Dans le cadre de ce projet, les dictionnaire ".pl" ont été choisies (sauf le "dico.pl"). Le plus grosse fichier est "D10.pl" est il dépasse la taille limite des variables d'environnement du compilateur dont une redéfinition est nécessaire.

- TRAILSZ : 16384 -> 50798592 Kb.
- CSTRSZ : 16384 -> 50798592 Kb.
- GLOBALSZ : 16384 -> 50798592 Kb.
- LOCALSZ : 16384 -> 50798592 Kb.
- MAX_ATOM : 32768 -> 33849344 atoms.

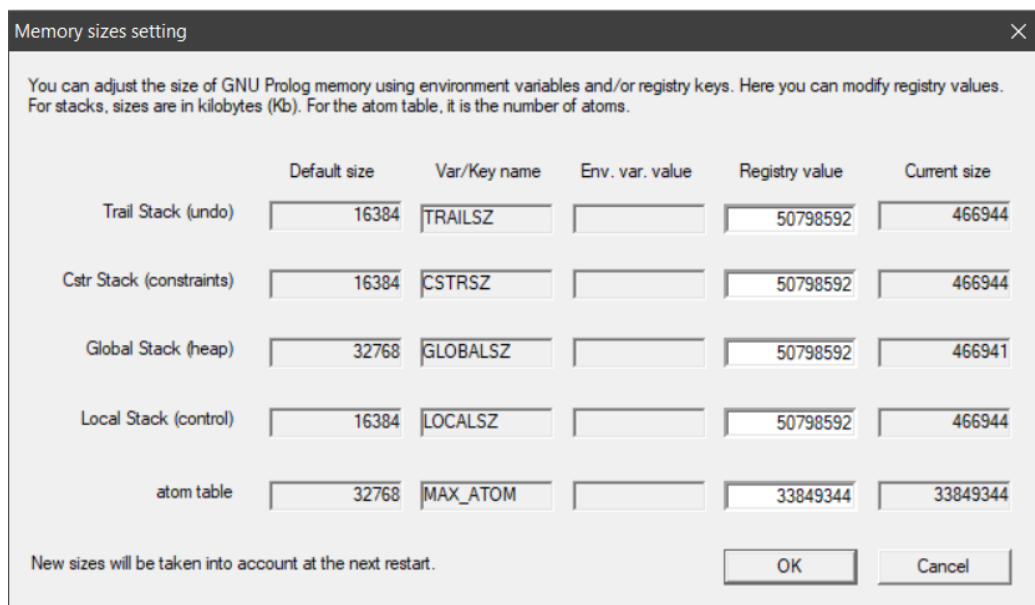


FIGURE 1 – Variables d'environnement.

2.2 Listes des prédicats built-in

références : GNU Prolog Manual.

- *append*(List1, List2, List12) succeeds if the concatenation of the list List1 and the list List2 is the list List12.
- *permutation*(List1, List2) succeeds if List2 is a permutation of the elements of List1.
- *atom_codes*(Atom, Codes) succeeds if Codes is the list of code ASCII of the successive characters of the name of Atom.
- *consult*(File).
- *atom_length*(Atom, Length) succeeds if Length unifies with the number of characters of the name of Atom.
- *number_atom*(Number, Atom) succeeds if Atom is an atom whose name corresponds to the characters of Number.
- *nth*(N, List, Element) succeeds if the Nth argument of List is Element.
- *read_atom*(Atom) succeeds if Atom unifies with the next atom read from the stream associated with the current stream.
- *member*(Element, List) succeeds if Element belongs to the List.
- *random*(Base, Max, Number) unifies Number with a random number such that $\text{Base} \leq \text{Number} < \text{Max}$.
- *sublist*(List1, List2) succeeds if all elements of List1 appear in List2 in the same order.
- *reverse*(List1, List2) succeeds if List2 unifies with the list List1 in reverse order.
- *delete*(List1, Element, List2) removes all occurrences of Element. in List1 to provide List2.
- *length*(List, Integer) succeeds if Integer is the length of List.

2.3 Listes des prédicats définis

- *majuscule*(L1,L2) vraie si chaque élément de la liste *L2* est la code ASCII (majuscule) de ceux dans *L1*
- *delete1_list*(L1,L2,L3) vraie si *L3* est obtenu par effacer les éléments de *L1* dans *L2* (une seule occurrence par chaque élément de *L1*).
- *consulte_fichier*(L) vraie si le fichier ".pl" est consulte, avec L la liste des code ASCII du première lettre et la longueur (ex : "A5.pl" - L=[65,53]).
- *information_mot*(Mot,UCodes,Lenth,CodeAtomLenth,Codechar1) vraie si
 - Mot : le mot (sous type *atom*)
 - UCodes : la liste des codes ASCII majuscules de chaque lettre dans

- le *Mot*
- *Lenth* : la longueur du mot
- *CodeAtomLenth* : le code ASCII de la longueur
- *Codechar1* : le code ASCII de la première lettre
- *valide*(*Mot*) vraie si un mot (*atom*) est dans la dictionnaire
- *countt*(*E,L,N*) vraie si *N* est le nombre d'occurrences de l'element *E* dans la liste *L*.
- *afficher1*(*CodeChar1,Lenth*) vraie si la première présentation du mot est imprimé sur l'écran (ex : "A.....") avec *CodeChar1* est le code ASCII de la première lettre et *Lenth* est la longueur du mot
- *lire_mot*(*MotCherche,UCodesMotCherche,Lenth,Codechar1*) vraie si
 - *UCodesMotCherche* et la liste des codes ASCII (majuscule)
 - *Lenth* est la longueur
 - *Codechar1* est le code ASCII de la première lettre du *MotCherche* (sous type *atom*) qui est lit depuis clavier
- *verifier_charNot*(*L1,L2*) vraie si tous les éléments dans *L2* n'appartiennent pas dans *L1*.
- *choisir_mot*(*CodeChar1,Lenth,ListeChar,ListeCharNot,CodesMot*) vraie si
 - *CodesMot* est la liste des codes ASCII majuscules du mot choisi dans la dictionnaire
 - *CodeChar1* est le code ASCII majuscule de la première lettre du mot
 - *Lenth* est la longueur du mot
 - *ListChar* est la liste des codes ASCII des lettres appartenant dans le mot choisi
 - *ListeCharNot* est la liste des codes ASCII des lettres n'appartenant pas dans le mot choisi
- *correct*(*L1,L2*) vraie si la liste *L1* est unifiée avec *L2*.
- *bcbp*(*L1,L2,L3*) vraie si pour chaque élément *E1,E2,E3* de la liste *L1,L2,L3* :
 - *E1 = E2 = E3* ou *E3 = 43* (on a choisi un code ASCII dont la lettre n'appartient pas dans l'alphabet).
 - (ex : *L1*=[1,3,2,1], *L2*=[1,1,2,4] , *L3* =[1,43,2,43]). Le but est d'appeler le prédicat *delete*(*L3,43,Lbcbp*) pour obtenir *Lbcbp*=[1,2]. *bcbp* - "bonne char bonne position".
- *bcmp*(*CodesMot,L1,L2,Lrest,L3,L4,Lbcbp*) vraie si
 - *L1* et *CodesMot* est la liste des codes ASCII majuscules du mot secret
 - *L2* est la liste des codes ASCII majuscules du mot proposé
 - *Lrest* est la liste des codes ASCII des lettres qui appartiennent

- dans le mot mais pas dans la liste Lbcbp
- L3 est pour obtenir L31 qui est la liste des codes ASCII des lettres qui appartiennent dans le mot mais pas en vraie position (on ne considère pas des lettre qui sont déjà dans Lbcbp) (ex ci-dessous)
- L4 est pour obtenir L41 qui est la liste des codes ASCII des lettres qui appartiennent dans L2 mais pas dans L1 (et CodesMot aussi) (ex ci-dessous)
- Lbcbp est la liste "bonne char bonne position"
(ex : L1=[1,3,2,1,5,4], L2=[1,1,2,4,5,7] , Lrest=[3,1,5,4], L3=[43,1,43,4,43,43], L4=[43,43,43,43,43,7], Lbcbp=[1,2]). Le but est d'appeler le prédicat *delete*(L3,43,L31) pour obtenir L31=[1,4] et *delete*(L4,43,L41) pour obtenir L41=[1,4]. bcmp - "bonne char mauvaise position".
- *check*(L1,L2,L3,L4) vraie si
 - L1 est la liste des codes ASCII majuscules du mot secret
 - L2 L2 est la liste des codes ASCII majuscules du mot proposé
 - L3 est la liste des codes ASCII majuscules qui doivent apparaître dans le mot secret
 - L4 est la liste des codes ASCII majuscules qui ne doivent pas apparaître dans le mot secret
 - les positions sont imprimées sur l'écran (par le prédicat *afficher*)(ex : !. ?.. ?)
- *afficher*(L1,L2,Lrest,Lbcbp) vraie si
 - L1 est la liste des codes ASCII majuscules du mot saisi
 - L2 L2 est la liste des codes ASCII majuscules du mot proposé
 - Lrest est la liste des codes ASCII des lettres qui appartiennent dans le mot mais pas dans la liste Lbcbp
 - Lbcbp est la liste "bonne char bonne position"
 - les positions sont imprimées sur l'écran (ex : !. ?.. ?)
- *start* est le prédicat pour afficher l'interface commune.
- *menu*(Command) est le prédicat pour afficher l'interface d'humain (Command = h) ou or d'ordinateur (Command = o).
- *humain*(L) est le prédicat pour traiter les autres prédicats lorsque l'humain joue avec L : la liste des codes ASCII majuscules du mot secret.
- *ordinateur*(Mot) est le prédicat pour traiter les autres prédicats lorsque l'ordinateur joue avec Mot : le mot secret (sous type *atom*).

2.4 Fonctionnement du code

2.4.1 Mode d'humain

Pour choisir un mot, le programme va choisir au hasard le code ASCII de la première lettre et la longueur du mot, de plus, on considère que un mot doit avoir une voyelle (\neq la première lettre, on peut le vérifier par le prédicat *choisir_mot*). Après, on ne consulte que le fichier correspondant (ex : 'A5.pl') et appelle le prédicat *humain*(CodesMotSecret). Ensuite, si la longueur du mot est 5(6->10) on donne au joueur 5(6) opportunités pour trouver le mot secret, on vérifie bien la validité du mot (si le joueur se trompe en saisissant le mot, on refaire la lecture).

2.4.2 Mode d'ordinateur

Dans ce projet, on a choisi le niveau AI naïve améliorée. Pour choisir un mot, l'utilisateur va donner le mot secret, on vérifie bien la validité du mot (si le joueur se trompe en saisissant le mot, on refaire la lecture). Après avoir appelé le prédicat *ordinateur*(Mot), l'ordinateur va choisir au hasard le code ASCII de la première lettre et la longueur du mot, le mot est choisi par le prédicat *choisir_mot*. Ensuite, si la longueur du mot est 5(6->10) on donne au joueur 5(6) opportunités pour trouver le mot secret. Le mot proposé dans un tour est différent contre tous les mots précédents et après chaque tour, on génère une liste qui contient les codes ASCII qui doivent apparaître dans le mot et une liste qui contient les codes ASCII qui ne doivent pas apparaître dans le mot.

2.5 Mode d'emploi

1. configurer des variables d'environnement.
2. "[motus]."
3. "start."
4. <caractere_clavier>+<entrée> ("o"=ordi,"h"=humain,"x"=exit)
5. <saisir_mot>+<entrée> (ex : "facile"+<entrée>) (tous les caractères sont minuscules et ne pas avoir le "." à la fin)

2.6 Résultat

2.6.1 Mode d'humain

Pour le mode d'humain, les résultats sont ci-dessous. on considère que le mot proposé par le joueur peut être dupliqué (mais il faut être valide).

```

GNU Prolog console
File Edit Terminal Prolog Help
| ?- start.
M O T U S !
-----
Qui joue ?
  o - ordinateur
  h - humain
  x - exit
Command (Syntax: [caractere] + [entree]):
h

Humain pret !
-----
7
compiling G:/References/BR/GI01/IA02/test prolog/R7.pl for byte code...
G:/References/BR/GI01/IA02/test prolog/R7.pl compiled, 2417 lines read - 1136196 bytes written,
warning: G:/References/BR/GI01/IA02/test prolog/R7.pl:1: redefining procedure dico/3
        G:/References/BR/GI01/IA02/test prolog/J8.pl:1: previous definition
R.....
Votre proposition :ranger
La taille du mot ou la 1ere caractere ou la validite n'est pas bonne ! re-essayez:
Votre proposition :rangent
RANGENT
!!....?
Votre proposition :rangent
RANGENT
!!....?

```

FIGURE 2 – L'écran.

```

GNU Prolog console
File Edit Terminal Prolog Help
Votre proposition :ranger
La taille du mot ou la 1ere caractere ou la validite n'est pas bonne ! re-essayez:
Votre proposition :rangent
RANGENT
!!....?
Votre proposition :rangent
RANGENT
!!....?
Votre proposition :rangent
RANGENT
!!....?
Votre proposition :rajoute
RAJOUTE
!!..?!.
Votre proposition :racines
RACINES
!!.....
Votre proposition :reignes
La taille du mot ou la 1ere caractere ou la validite n'est pas bonne ! re-essayez:
Votre proposition :renards
RENARDS
!!....
Pas correct !
Le mot secret est :RABATTU

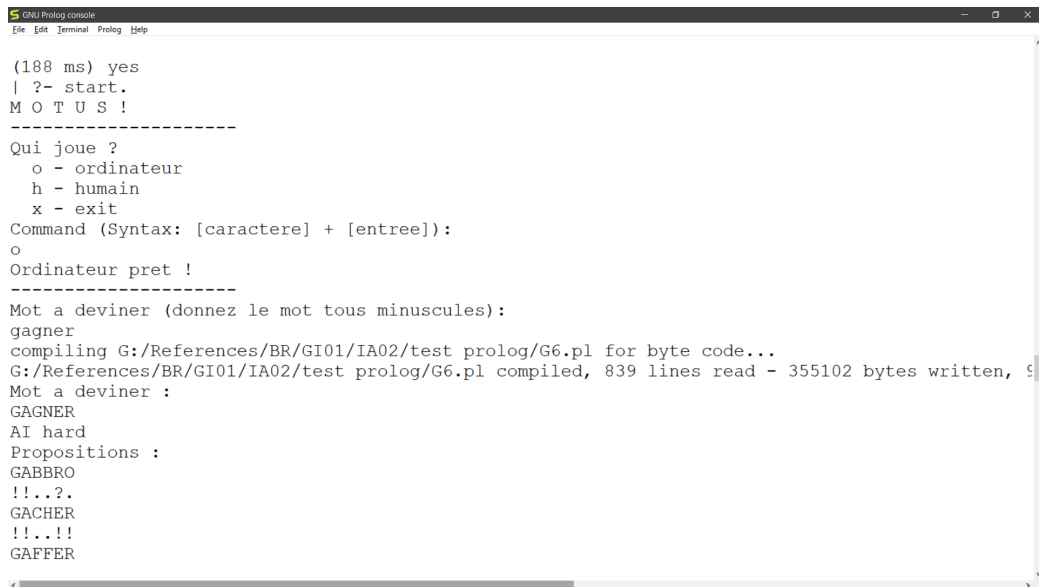
(235 ms) yes

```

FIGURE 3 – L'écran.

2.6.2 Mode d'ordinateur

Pour le mode d'humain, les résultats sont ci-dessous. Pour le mot "gagner" : Le cas réussi :



```
GNU Prolog console
File Edit Terminal Prolog Help

(188 ms) yes
| ?- start.
M O T U S !
-----
Qui joue ?
  o - ordinateur
  h - humain
  x - exit
Command (Syntax: [caractere] + [entree]):
o
Ordinateur pret !
-----
Mot a deviner (donnez le mot tous minuscules):
gagner
compiling G:/References/BR/GI01/IA02/test prolog/G6.pl for byte code...
G:/References/BR/GI01/IA02/test prolog/G6.pl compiled, 839 lines read - 355102 bytes written, 9
Mot a deviner :
GAGNER
AI hard
Propositions :
GABBRO
!!!?..
GACHER
!!!!!!
GAFFER
```

FIGURE 4 – L'écran.



```
GNU Prolog console
File Edit Terminal Prolog Help

Ordinateur pret !
-----
Mot a deviner (donnez le mot tous minuscules):
gagner
compiling G:/References/BR/GI01/IA02/test prolog/G6.pl for byte code...
G:/References/BR/GI01/IA02/test prolog/G6.pl compiled, 839 lines read - 355102 bytes written, 9
Mot a deviner :
GAGNER
AI hard
Propositions :
GABBRO
!!!?..
GACHER
!!!!!!
GAFFER
!!!!!!
GAGERA
!!!??..
GAGEUR
!!!?..!
GAGNER
!!!!!!
C'est facile ! - dit Ordi.

(110 ms) yes
| ?- start.
```

FIGURE 5 – L'écran.

Le cas échec :

```

GNU Prolog console
File Edit Terminal Prolog Help
| ?- start.
M O T U S !
-----
Qui joue ?
  o - ordinateur
  h - humain
  x - exit
Command (Syntax: [caractere] + [entree]):
o
Ordinateur pret !
-----
Mot a deviner (donnez le mot tous minuscules):
gagner
compiling G:/References/BR/GI01/IA02/test prolog/G6.pl for byte code...
G:/References/BR/GI01/IA02/test prolog/G6.pl compiled, 839 lines read - 355102 bytes written, 9
warning: G:/References/BR/GI01/IA02/test prolog/G6.pl:1: redefining procedure dico/3
          G:/References/BR/GI01/IA02/test prolog/R7.pl:1: previous definition
Mot a deviner :
GAGNER
AI hard
Propositions :
GAMAYS
!!....
GABARE
!!...??
GACHER

```

FIGURE 6 – L'écran.

```

GNU Prolog console
File Edit Terminal Prolog Help
-----
Mot a deviner (donnez le mot tous minuscules):
gagner
compiling G:/References/BR/GI01/IA02/test prolog/G6.pl for byte code...
G:/References/BR/GI01/IA02/test prolog/G6.pl compiled, 839 lines read - 355102 bytes written, 9
warning: G:/References/BR/GI01/IA02/test prolog/G6.pl:1: redefining procedure dico/3
          G:/References/BR/GI01/IA02/test prolog/R7.pl:1: previous definition
Mot a deviner :
GAGNER
AI hard
Propositions :
GAMAYS
!!....
GABARE
!!...??
GACHER
!!...!!
GAFFER
!!...!!
GAGERA
!!!??.
GAGEUR
!!!?.!
C'est dur ! - dit Ordi.

(188 ms) yes

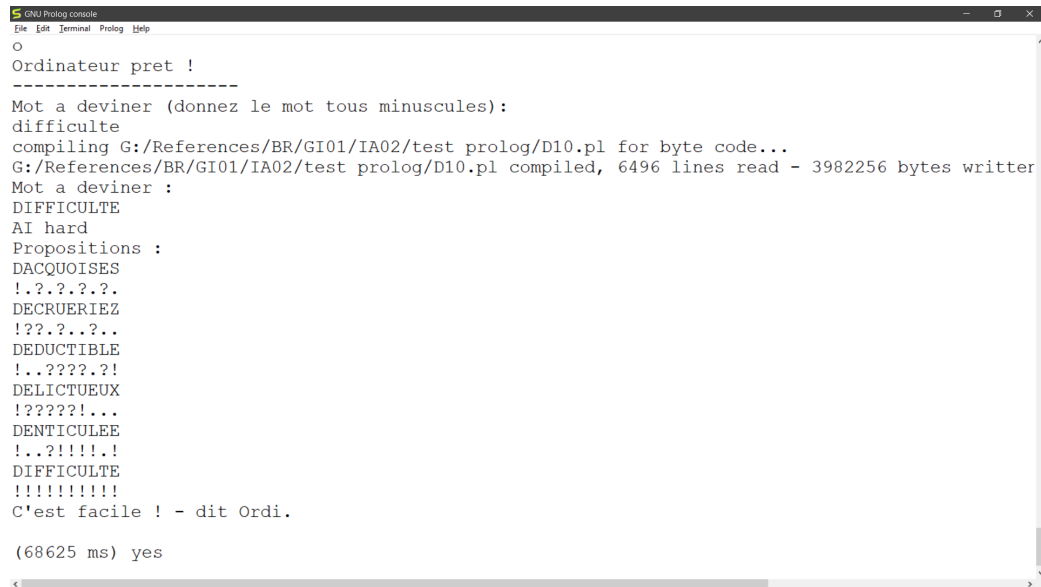
```

FIGURE 7 – L'écran.

2.6.3 Performance

Le fait d'utiliser des fichiers "`_pl`" risques des blocages dans le console à cause de la limite de la taille des variables d'environnement ou il faut attendre une unité de temps. Pour le fichier le plus grosse "`D10.pl`", le resultat est ci-

dessous.



```
GNU Prolog console
File Edit Terminal Prolog Help
O
Ordinateur pret !
-----
Mot a deviner (donnez le mot tous minuscules):
difficulte
compiling G:/References/BR/GI01/IA02/test prolog/D10.pl for byte code...
G:/References/BR/GI01/IA02/test prolog/D10.pl compiled, 6496 lines read - 3982256 bytes written
Mot a deviner :
DIFFICULTE
AI hard
Propositions :
DACQUOISES
!.?.?.?.?.
DECRUERIEZ
!?.?.?.?.
DEDUCTIBLE
!..?????!
DELICTUEUX
!?????!...
DENTICULEE
!..?!!!!!!
DIFFICULTE
!!!!!!
C'est facile ! - dit Ordi.

(68625 ms) yes
```

FIGURE 8 – 68625 ms!.

3 Conclusion

En réalisant ce projet, j'ai mieux compris le langage Prolog ainsi que le récursivité. Le AI Motus peut être développé encore pour obtenir un résultat mieux précis et vide. On peut aussi améliorer la performance de ce programme.

4 Rendu

- Le dossier comprend :
- la dictionnaire : "A5.pl" -> "Z10.pl"
 - le motus.pl
 - le rapport format PDF