Sy-Toan NGO
Marion CHAN-RENOUS-LEGOUBIN

IA01 – RENDU TP03

Conduite d'expertise d'un SE d'ordre 0+

Introduction

Le but de ce TP est de réaliser le développement d'un système expert de sa phase d'expertise à sa phase d'utilisation. Pour se faire, il nous est donc demandé de formaliser une problématique d'un domaine au choix pouvant être traitée par un SE d'ordre 0+. Il est ensuite nécessaire de déterminer la base de règles de notre SE en fonction des différentes sources d'expertise trouvées. Puis, nous programmerons notre SE et le testerons.

Question 1

Nous avons choisi la problématique suivante :

- Quel est le sport le plus adapté à pratiquer selon les différentes caractéristiques d'une personne ?

Question 2

Notre base de règles est composée de 48 règles.

Les règles R1 à R6 concernent la taille d'une personne :

```
(R1 ((> taille 190) (sexe masculin)) (taille grande))

(R2 ((> taille 174) (sexe feminin)) (taille grande))

(R3 ((< taille 163) (sexe masculin)) (taille petite))

(R4 ((< taille 152) (sexe feminin)) (taille petite))

(R5 ((>= taille 152) (<= taille 174) (sexe feminin)) (taille
moyenne))

(R6 ((>= taille 163) (<= taille 187) (sexe masculin)) (taille
moyenne))
```

Afin de déterminer ces règles, nous nous sommes basés sur les courbes de croissance en fonction de l'âge et de la taille de l'individu. La taille d'un individu peut donc prendre trois valeurs : petite, grande et moyenne, en fonction de la taille (en cm) et de l'âge (en années) entrés par l'utilisateur.

Source : http://www.auxologie.com/croissance/

Les règles R7 à R11 concernent les qualités physiques d'une personne :

```
(R7 ((>= puissance 5) (>= resistance 5)) (qualite force))

(R8 ((>= fatigue 5) (>= duree 5)) (qualite endurance))

(R9 ((>= reactivite 5) (>= explosivite 5)) (qualite vitesse))

(R10 ((>= habilete  5) (>= precision 5)) (qualite adresse))

(R11 ((>= amplitude 5) (>= flexibilite 5)) (qualite souplesse))
```

Afin de déterminer ces règles, nous avons fait plusieurs recherches sur internet. Nous avons tiré nos règles de sources telles que :

http://www.gymsante.eu/blog/les-qualites-physiques-du-sportif-2294/

http://sante.lefigaro.fr/mieux-etre/sports-activites-physiques/condition-physique/quelles-sont-qualites-physiques

Les règles R12 à R17 exploitent les caractéristiques mentales d'une personne afin de déterminer si l'individu est plus apte à la pratique d'un sport collectif ou individuel.

```
(R12 ((>= extraversion 5) (>= ouverture 5)) (pratique individuel))

(R13 ((>= extraversion 5) (>= consciencieux 5)) (pratique individuel))

(R14 ((>= ouverture 5) (>= consciencieux 5)) (pratique individuel))

(R15 ((< extraversion 5) (< ouverture 5)) (pratique collectif))

(R16 ((< extraversion 5) (< consciencieux 5)) (pratique collectif))

(R17 ((< consciencieux 5) (< ouverture 5)) (pratique collectif))
```

Afin d'établir ces règles, nous nous sommes basés sur l'étude suivante :

https://idosi.org/mejsr/mejsr9(4)11/17.pdf : "*Comparaison of Personal Traits between Individual and Team Athletes"*

Les règles R18 à R20 concernent l'IMC d'une personne, afin de définir la silhouette de celle-ci, petite, moyenne ou large.

```
(R18 ((> IMC 25)) (silhouette large))

(R19 ((>= IMC 18.5) (<= IMC 25)) (silhouette moyenne))

(R20 ((< IMC 18.5)) (silhouette fine))
```

Afin d'obtenir l'IMC d'une personne, en fonction de son poids (en kg) et de sa taille (en cm) nous avons défini une fonction spéciale :

```
(defun calculIMC ()

  (let ((IMC 0) (taille (cadr (assoc 'taille *BF*))) (poids (cadr (assoc 'poids *BF*))))

    (setq IMC (/ poids (* (/ taille 100) (/ taille 100))))

    (push (list 'IMC IMC) *BF*)

    IMC

    )

  )
```

Cette fonction opère directement sur la base de faits *BF* et ajoute immédiatement la valeur de l'IMC sous la forme (IMC valeur) dans la base de faits.

Pour analyser l'IMC nous avons utilisé cette source :

http://www.calculersonimc.fr/faites-le-test.html

Les règles R21 à R48 permettent de déterminer le sport idéal en fonction des caractéristiques établies par les règles précédentes.

```
     (R21 ((pratique individuel) (qualite endurance) (qualite
vitesse)) (sport athletisme))

     (R22 ((pratique collectif) (taille moyenne) (qualite adresse))
(sport basketball))

     (R23 ((pratique collectif) (taille grande) (qualite adresse))
(sport basketball))

     (R24 ((pratique collectif) (silhouette fine) (qualite
endurance)) (sport football))

     (R25 ((pratique collectif) (silhouette moyenne) (qualite
endurance)) (sport football))

     (R26 ((pratique individuel) (qualite force) (qualite
endurance)) (sport natation))

     (R27 ((pratique collectif) (silhouette moyenne) (qualite force)
(qualite endurance)) (sport rugby))

     (R28 ((pratique collectif) (silhouette large) (qualite force)
(qualite endurance)) (sport rugby))

     (R29 ((pratique individuel) (qualite souplesse)) (sport danse))

     (R30 ((pratique individuel) (silhouette fine) (qualite
adresse)) (sport equitation))

     (R31 ((pratique individuel) (silhouette moyenne) (qualite
adresse)) (sport equitation))

     (R32 ((pratique individuel) (qualite force) (qualite adresse))
(sport tennis))

     (R33 ((pratique individuel) (qualite adresse) (sport ski)))

     (R34 ((pratique individuel) (silhouette fine) (qualite
endurance)) (sport cyclisme))

     (R35 ((pratique individuel) (silhouette moyenne) (qualite
endurance)) (sport cyclisme))

     (R36 ((pratique individuel) (silhouette fine) (qualite
adresse)) (sport escrime))

     (R37 ((pratique individuel) (silhouette moyenne) (qualite
adresse)) (sport escrime))

     (R38 ((pratique individuel) (silhouette fine) (qualite
souplesse)) (sport gymnastique))

     (R39 ((pratique individuel) (silhouette moyenne) (qualite
souplesse)) (sport gymnastique))
```

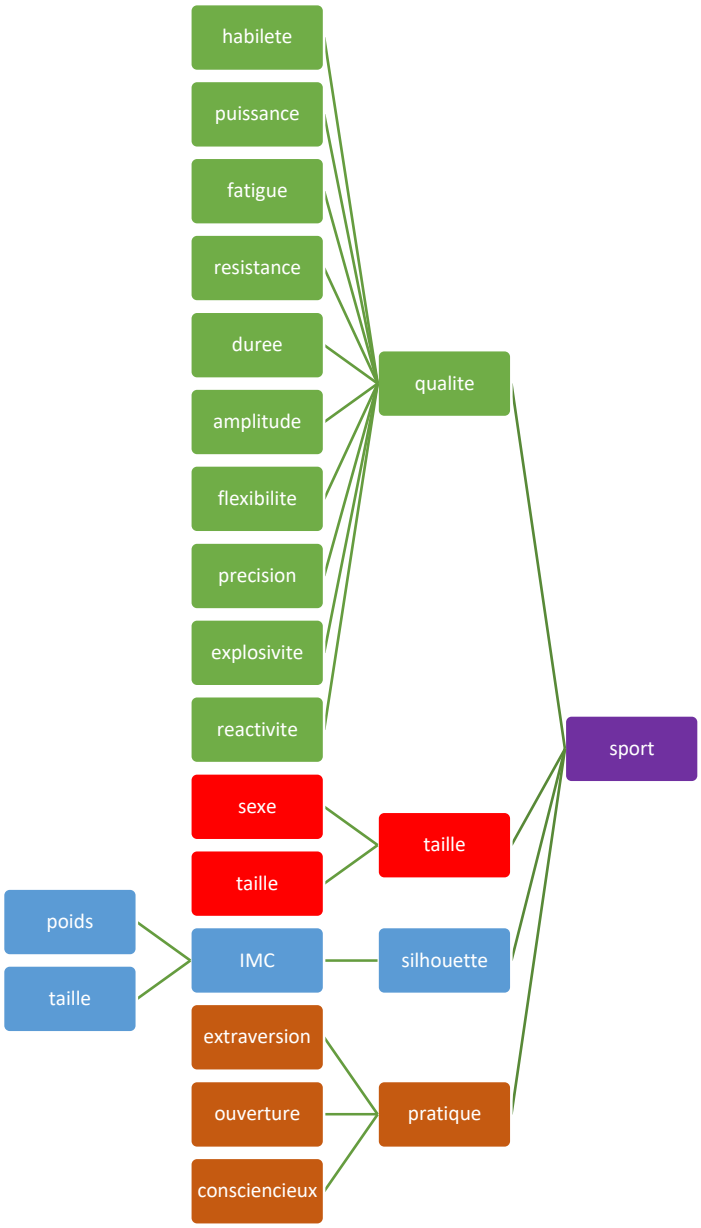
```
(R40 ((pratique individuel) (silhouette fine) (qualite force)
(qualite endurance)) (sport aviron))

(R41 ((pratique individuel) (silhouette moyenne) (qualite
force) (qualite endurance)) (sport aviron))

(R42 ((pratique individuel) (silhouette moyenne) (qualite
force)) (sport lutte))

(R43 ((pratique individuel) (silhouette large) (qualite force))
(sport lutte))

(R44 ((pratique individuel) (silhouette moyenne) (qualite
adresse)) (sport BMX))

(R45 ((pratique individuel) (silhouette fine) (qualite
adresse)) (sport BMX))

(R46 ((pratique individuel) (qualite adresse)) (sport golf))

(R47 ((pratique collectif) (silhouette moyenne) (qualite
adresse) (qualite endurance)) (sport handball))

(R48 ((pratique collectif) (silhouette large) (qualite adresse)
(qualite endurance)) (sport handball))
```

Nous avons utilisé deux sources principales d'expertise pour ces règles :

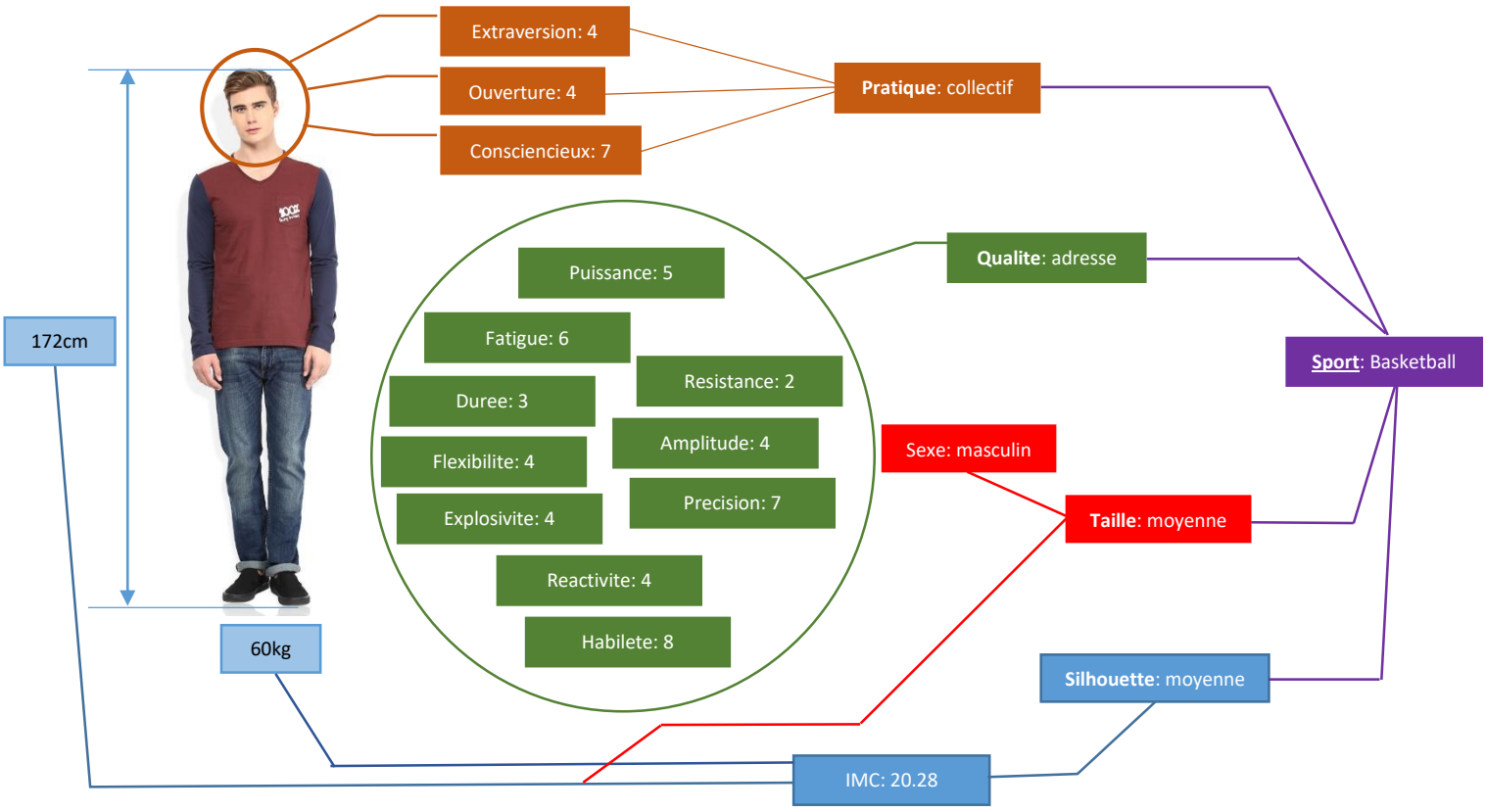
https://www.citesport.com/fr/sport/sport.html

http://kutchuk.com/vivelesport/Pratiquer.htm

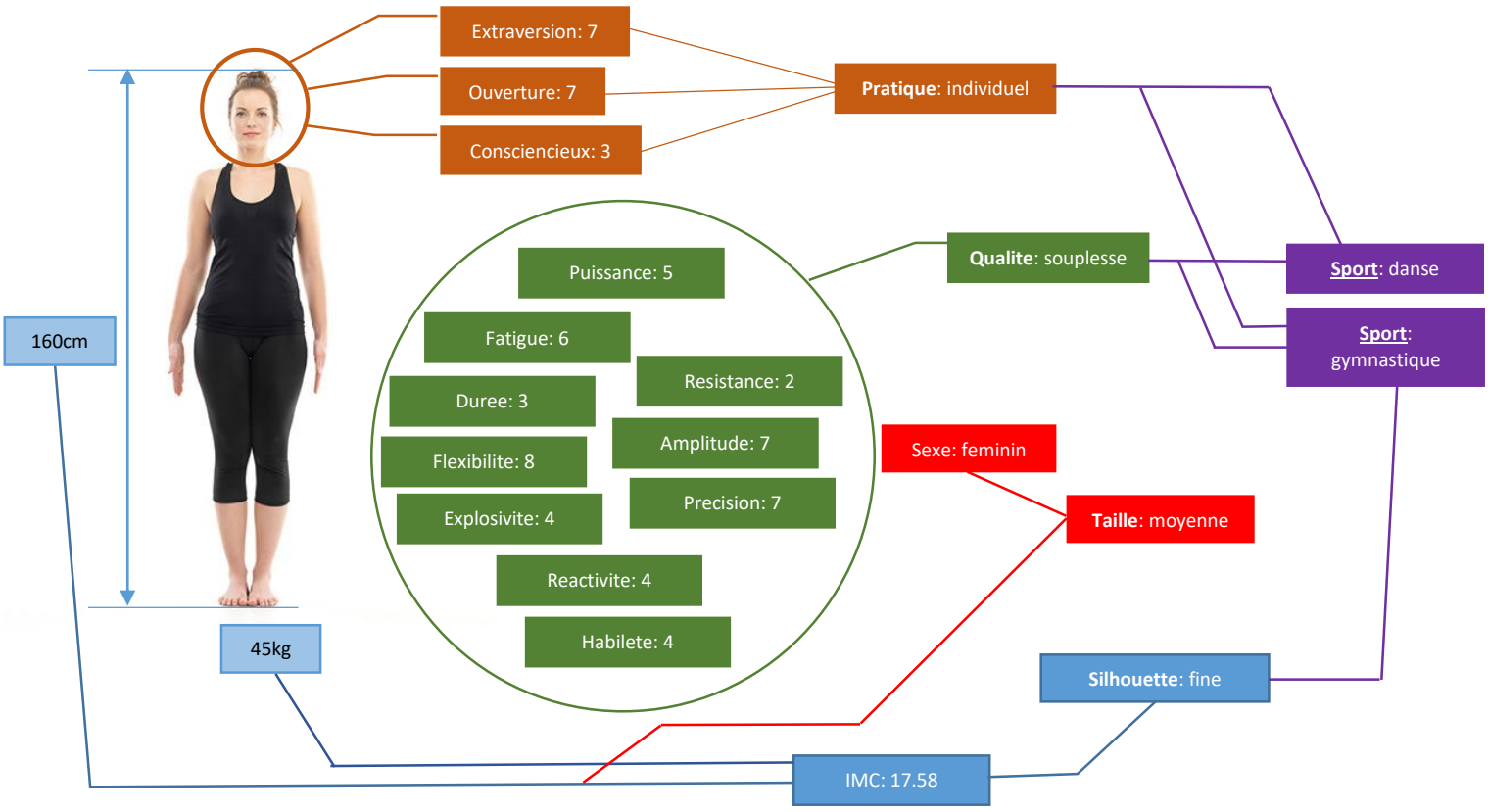
Arbre de déduction



Jeux d'essai


Extraversion: 4
Ouverture: 4
Consciencieux: 7
Pratique: collectif
Puissance: 5
Qualite: adresse
Fatigue: 6
Resistance: 2
Duree: 3
Amplitude: 4
Flexibilite: 4
Precision: 7
Explosivite: 4
Reactivite: 4
Habilete: 8
Sport: Basketball
Sexe: masculin
Taille: moyenne
172cm
60kg
Silhouette: moyenne
IMC: 20.28

Jeux d'essai


Extraversion: 7
Ouverture: 7
Consciencieux: 3
Pratique: individuel
Puissance: 5
Fatigue: 6
Resistance: 2
Duree: 3
Amplitude: 7
Flexibilite: 8
Precision: 7
Explosivite: 4
Reactivite: 4
Habilete: 4
Qualite: souplesse
Sport: danse
Sport: gymnastique
160cm
45kg
Sexe: feminin
Taille: moyenne
Silhouette: fine
IMC: 17.58

Question 3

a) Afin d'exploiter les faits et les règles nous avons choisi la représentation suivante :
Faits : (attribut valeur)
Exemples : `(taille grande), (qualite vitesse), (sport football).`
Règles : (numéro de la règle (liste de prémisses) conclusion)
Nous avons utilisé cette représentation avec une clé correspondant au numéro de la règle afin de pouvoir récupérer facilement les prémisses ou la conclusion d'un règle à partir de son numéro grâce à la fonction assoc.
Exemple : `(R1 ((> taille 190) (sexe masculin)) (taille grande))`

b) Nous avons choisi d'offrir deux possibilités à l'utilisateur de notre système expert. La première est celle de pouvoir tester si la personne est apte à pratiquer un sport donné en fonction de ses caractéristiques. Pour se faire, nous avons choisi de programmer un moteur d'inférence **en chainage arrière, en profondeur d'abord.** La deuxième possibilité consiste à donner pour une personne la liste des sports pour laquelle elle est apte selon la liste des sports possibles dans les règles de notre système expert. Dans ce deuxième cas, nous avons programmé un moteur d'inférence **en chainage avant, en largeur d'abord.**

Fonctions outils :

- Fonction premisses (regle) : fonction qui renvoie les prémisses d'une règle en fonction du numéro de règle fourni en paramètre

```
(defun premisses (regle)
  (let (result)
    (setq result (cadr (assoc regle *BR*)))
    )
  )
```

- Fonction conclusion (regle) : fonction qui renvoie la conclusion d'une règle en fonction du numéro de règle fourni en paramètre

```
(defun conclusion (regle)
  (let (result)
    (setq result (caddr (assoc regle *BR*)))
```

```
    )
  )
```

- Fonction verifier_dans_BF (but) : fonction qui vérifie si le but à atteindre est dans la base de faits

```
(defun verifier_dans_BF (but)
  (member but *BF* :test #'equal)
  )
```

- Fonction regles_candidates () : fonction qui renvoie les règles applicables en fonction de la base de faits

```
(defun regles_candidates ()
  (let ((result nil))
    (dolist (var *BR* result)
      (if (eq t (verifier_premisses (car var)))
(push (car var) result))
     )
    )
  )
```

- Fonction verifier2 (premisse) : fonction qui vérifie les prémisses dont la longueur est 2 : c'est-à-dire les prémisses de la forme (attribut valeur)

```
(defun verifier2 (premisse)
           (let ((ok nil))
           (loop for f in *BF*
               while (null ok) do
                 (if (and (eq (car f) (car
premisse))(eq (cadr f) (cadr premisse))) (setq ok
t))
                )
           ok
             )
           )
```

- Fonction verifier3 (premisse) : fonction qui vérifie les prémisses dont la longueur est 3 : c'est-à-dire les prémisses dont il faut tester la valeur de l'attribut, sous la forme (signe mathématique attribut valeur)

```
(defun verifier3 (premisse)
           (let ((ok nil) (tmp nil))
           (loop for f in *BF*
               while (null ok) do
                 (cond
                  ((and (eq (car f) (cadr premisse))
(numberp (cadr f))) (setq tmp (list (car premisse)
(cadr f)(caddr premisse))) (setq ok (eval tmp)))
                  (t nil)
```

```
                    )
                  )
              ok
                )
        )
```

- Fonction verifier_premisses (regle) : fonction qui vérifie toutes les prémisses d'une règle en fonction de sa longueur

```
(defun verifier_premisses (regle)
  (let ((ok t) (listepremisses (premisses regle)))
    (loop for p in listepremisses
        while ok do
          (cond
           ((eq (length p) 2) (setq ok (verifier2
p)))
           ((eq (length p) 3) (setq ok (verifier3
p)))
           (t nil)
           )
          )
      ok
      )
  )
```

- Fonction poser_question() : fonction permettant de poser des questions à l'utilisateur afin de remplir la base de faits

```
(defun poser_question()
  (let ((result) (value) (symbol))
    (terpri)
    (princ "Sexe (masculin/feminin): ")
    (setq value (read))
    (setq symbol 'sexe)
    (push (list symbol value) result)
    (terpri)
    (princ "Taille (cm): ")
    (setq value (read))
    (setq symbol 'taille)
    (push (list symbol value) result)
    (terpri)
    (princ "Poids (kg): ")
    (setq value (read))
    (setq symbol 'poids)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note à votre puissance physique
(de 1 a 10) : ")
    (setq value (read))
    (setq symbol 'puissance)
    (push (list symbol value) result)
    (terpri)
```

```
    (princ "Donnez une note à votre resistance durant
l'effort (de 1 a 10) : ")
    (setq value (read))
    (setq symbol 'resistance)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note à votre capacité à resister à
la fatigue (de 1 a 10) : ")
    (setq value (read))
    (setq symbol 'fatigue)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note à votre reactivite (de 1 a
10) : ")
    (setq value (read))
    (setq symbol 'reactivite)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note à votre explosivite (de 1 a
10) : ")
    (setq value (read))
    (setq symbol 'explosivite)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note à votre habilete (de 1 a 10)
: ")
    (setq value (read))
    (setq symbol 'habilete)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note à votre precision (de 1 a 10)
: ")
    (setq value (read))
    (setq symbol 'precision)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note à votre amplitude de
mouvements possible (de 1 a 10): ")
    (setq value (read))
    (setq symbol 'amplitude)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note à votre flexibilite (de 1 a
10): ")
    (setq value (read))
    (setq symbol 'flexibilite)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note sur votre capacité à produire
un effort sur de longues durees (de 1 a 10): ")
    (setq value (read))
    (setq symbol 'duree)
```

```
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note sur votre extraversion (de 1
a 10): ")
    (setq value (read))
    (setq symbol 'extraversion)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note à votre ouverture d'esprit
(de 1 a 10) : ")
    (setq value (read))
    (setq symbol 'ouverture)
    (push (list symbol value) result)
    (terpri)
    (princ "Donnez une note à votre capacité à etre
consciencieux (de 1 a 10) : ")
    (setq value (read))
    (setq symbol 'consciencieux)
    (push (list symbol value) result)
    (setf *BF* result))
    (calculIMC)
    )
    )
```

- Fonction menuSE() : le menu de notre système expert

```
(defun menuSE()
  (let ((option) (value) (enquete) (result))
  (terpri)
  (poser_question)
  (terpri)
  (princ "Option 1 : Determiner si une personne est faite
pour un sport donné")
  (terpri)
  (princ "Option 2 : Donner la liste des sports possibles
pour une personne")
  (terpri)
  (princ "Option 1 ou 2: ")
  (setq option (read))
  (if (equal option 1)
      (progn
        (terpri)
        (princ "Votre sport: ")
        (setq value (read))
        (setq enquete (list 'sport value))
        (setq result (verifier enquete))
        (if (equal result 1) (princ "Oui!") (princ
"Non!"))
        )
    (chainage_avant2)
        )
    )
```

```
        )
```

Moteur d'inférence en chainage arrière en profondeur d'abord :

```
(defun VERIFIER (but)
  (let ((OK 0))
    (if (or (verifier2 but) (verifier3 but)) (setq OK 1)
      (progn
       (setq EC '())
        (dolist (element *BR*) (if (and (eq (car (conclusion
(car element))) (car but))
                                        (eq (cadr (conclusion
(car element))) (cadr but)))
                                    (push element EC)))
       (dolist (Ri EC)
         (if (equal OK 0)
           (setq OK (VERIFIER_ET Ri))
           )
         )
        ))
    OK
    ))
(defun VERIFIER_ET (regle)
  (let ((OK 1) (premisses))
    (setq premisses (cadr regle))
    (dolist (Pre premisses)
      (if (equal OK 1)
        (setq OK (VERIFIER Pre))
        )
      )
    OK
    ))
```

Explication :

Le moteur d'inférence en chainage arrière en profondeur d'abord se décompose en deux fonctions. La fonction vérifier permet d'extraire les règles qui ont en conclusion le but à prouver. Pour chaque prémisse de la règle extraite, on appelle la fonction verifier_et. Celle-ci appelle la fonction vérifier pour chaque prémisse.

Moteur d'inférence en chainage avant en largeur d'abord :

```
(defun chainage_avant2 ()
  (let ((conflits (regles_candidates)))
    (loop while (not (null conflits)) do
          (dolist (var conflits)
            (push (conclusion var) *BF*)
            (setq *BR* (delete var *BR* :test 'equal :key
'car))
            )
          (setq conflits (regles_candidates))
          )
    (dolist (var *BF*)
      (if (eq 'sport (car var)) (print var))
      )
    )
  )
```

Explication :

Cette fonction permet d'obtenir pour une personne donnée (base de faits remplie préalablement) la liste des sports praticables par celle-ci. Tout d'abord, nous déclarons une variable locale « conflits » et nous l'initialisons avec la liste des règles applicables grâce à la fonction-outil « regles_candidates ». Puis, nous faisons une boucle tant que, avec comme condition de sortie (not (null conflits), c'est-à-dire lorsqu'il reste des règles applicables à la base de faits. A chaque itération dans la boucle tant que, on itère sur la liste des règles applicables contenues dans la variable conflits. Nous avons choisi d'effectuer la recherche en largeur d'abord. Ainsi, pour chaque règle applicable, on ajoute la conclusion de celle-ci à la base de faits avec la fonction push, et on supprime la règle appliquée grâce à la fonction delete. Ensuite, on affecte à la variable conflits la listes des nouvelles règles applicables à partir des nouveaux faits ajoutés à la base des faits.
A la sortie de la boucle tant que, on filtre la base des faits pour afficher à l'écran les sports possibles pour la personne.

c) Comparaison des deux moteurs :

Conclusion

Notre système expert terminé remplit les fonctions que nous souhaitions au départ. Nous avons utilisé une gamme de sport étendue (une vingtaine de sports différents) ce qui permet des résultats XXX. Les deux options qu'offrent le menu de notre programme exploitent deux moteurs d'inférence différents ce qui nous a permis d'apprendre à implémenter ces deux types de moteurs

et d'en discerner les différences. Les inférences sont basées sur trois aspects majeurs d'une personne : ses capacités physiques, caractéristiques mentales et sa condition physique. Nous avons utilisé plusieurs sources d'expertise pour programmer les règles de ce système expert, nous pouvons donc affirmer que les résultats de ce système sont cohérents.

Notre programme pourrait être amélioré dans le futur. En effet, nos règles se basent sur l'appréciation qu'une personne a de soi-même : il note ses capacités physiques et mentales sur une échelle de 1 à 10. Il serait ainsi intéressant de trouver des critères objectifs, avec par exemple des tests physiques réels et ainsi d'insérer les résultats obtenus dans notre système expert pour constituer la base de faits. Il aurait également été possible d'ajouter de nombreux critères pour les sports : la localisation (en intérieur ou extérieur), le type de contact entre les participants, le coût de la pratique du sport…