**IA01-TP03**

**Conduite d’expertise d’un système expert d’ordre 0+**

Maxime DILY, Simon LAURENT

Université

de

Technologie

de

Compiègne

Ce document fait office de compte rendu du TP03.

Il a pour but d’expliquer les différentes étapes du développement de notre système expert

De la réflexion du sujet jusqu’à la programmation de ce dernier en détaillant les difficultés rencontrées

Bonne dégustation oenologique

Automne 2015

Responsable UV :

Marie-Helene ABEL

Responsable TD/TP :

Dominique LENNE

Table des matières

I. Énoncé du sujet 3

II. Formalisation de la problématique 4

1. Définition des termes 4

2. Choix du sujet 4

3. Introduction 5

III. Représentation des connaissances 5

1. Base de connaissance 5

2. Base de fait 6

3. Base de règles 6

4. Jeux d’essais 6

*5.* Arbre de déduction 6

IV. Programmation du Système expert 7

1. Représentation Lisp 7

2. Moteur d’inférences 7

3. Fonctions de services 8

a. Fonctions principales 8

b. Fonctions de gestion de la base de fait 9

c. Fonction d’affichage 10

d. Programme à exécuter 10

4. Résultats 11

a. Les différentes étapes de fonctionnement 11

b. Analyse des étapes 13

V. Conclusion 14

Annexe 15

# Énoncé du sujet

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

Dates de remise :

* -  Lundi 30 novembre 2015 à 18H pour la réponse à la question 1.
* -  Lundi 4 janvier 2016 à 18H pour la réponse aux questions 2 et 3.
* -  Démonstration et présentation orale lors du dernier TD.
* L’objet du TP03 est de réaliser le développement d’un SE de sa phase d’expertise à sa phase d’utilisation. A cette fin, vous devez :
  1. Formalisez une problématique d’un domaine au choix (un qui vous passionne) qui puisse être traitée par un SE d’ordre 0+. Justifiez votre choix et faites-le valider par votre chargé de TD.
  2. Déterminez les connaissances nécessaires au SE : explicitez votre base de règles (donnez vos sources). Présentez l’arbre de déduction associé et donnez des jeux d’essais.
  3. Programmez votre SE
     1. Justifiez la représentation Lisp choisie pour exploiter les faits et la base de  règles.
     2. Développez, justifiez et commentez le moteur d’inférences choisi: chaînage  avant (ou arrière) en profondeur d’abord, chaînage avant (ou arrière) en largeur  d’abord.
     3. Testez votre moteur et commentez les résultats. Une comparaison avec un  deuxième moteur développé serait un plus.
* Documents à produire :
*   Un rapport écrit comportant les réponses aux points précédents et présentant des  scénarios d’utilisation.
*   Un fichier comportant le code lisp de votre SE avec les scénarios d’utilisation (à  envoyer par courriel).
*   Une courte présentation orale s’appuyant sur des transparents et une démonstration  sont attendues au cours du dernier TD.

# Formalisation de la problématique

## Définition des termes

Le sujet de ce dernier TP est de réaliser un système expert d’ordre 0+. Nous allons tout d’abord expliciter ce que l’on nous demande, c'est-à-dire ce qu’est un système expert, puis l’ « ordre 0+ ».

* Système expert : système permettant la résolution de problèmes difficiles, en particulier remplacer un expert humain, ou au moins l’aider dans sa démarche d’expertise. Un système expert repose sur trois composantes : la base de faits, la base de règles et le moteur d’inférence. Il existe trois types de systèmes experts définis par leur ordre : 0, 0+ ou 1. Le choix de l'ordre détermine le formalisme des règles.
* Ordre 0+ : Les règle seront sous la forme d’un couple attribut valeur.

La première partie de ce TP, consistait à choisir un sujet qui pouvait correspondre à un système expert d’ordre 0+.

## Choix du sujet

Notre première idée fut de modéliser le très célèbre jeu Akinator pour en faire une version automobile. Cependant, ce sujet était trop facile et peu intéressant. Nous avons dans un second temps choisit de modéliser un système expert pour le choix des UVs en Génie Informatique à l’UTC. Néanmoins, nous fîmes face à un refus car ce sujet était trop rependu. Il avait déjà été fait à de nombreuses reprises. Ainsi nous avons décidé de faire un système expert pour l’œnologie.

Cependant, nos premiers tests permettant la résolution ne convergeaient pas vers la structure d'un système expert à proprement parler. En effet nous étions parti vers une structure de base de donnée. Suite à un rendez-vous avec Mme ABEL et à ses conseils, nous avons légèrement modifier notre système afin qu’il réponde aux critères. Finalement nous traiterons d’un système d'aide à l’achat d’un vin. Dans le but de cadrer le sujet nous limiterons notre système à la Bourgogne, le Champagne et le Bordelais. En effet, nous avons la chance d’avoir un terroir viticole de plus de 130 000 vins différents.

Notre base de règles est modélisée selon un formalisme d’ordre 0+. Cette dernière fonctionnera à partir des réponses aux questions entrées par l’utilisateur. Nous basons notre expertise sur le fameux « Guide Hachette des Vins 2016 ». Quand à notre moteur d’inférence, il fonctionnera en chainage avant. Le but de ce système expert est qu’il vous retourne une liste de vins en fonction de vos critères.

## Introduction

A l’approche des fêtes de noël vous décidez de vous mettre en quête des cadeaux à offrir à vos proches. Problème, cette année vous ne savez pas quoi offrir à votre père. En effet ce dernier dans la cinquantaine possède déjà tout pour être heureux. Alors après avoir passé en revue l’ensemble des derniers produits Hi-Tech et autres objets matériels qui finiront au grenier ou en dépôt vente, vous décidez de lui offrir l’une des bonnes choses de ce monde, une bouteille de vin. Nouvelle difficulté, votre père étant un épicurien vous ne pouvez pas acheter la première bouteille venue. Vous décidez donc de vous en remettre à un caviste. Néanmoins, cela ne vous suffit pas. Vous avez peur de l’arnaque. C’est là que vous réalisez que vous avez à disposition un système expert pour vous aider dans cette noble quête. Bonne dégustation.

# Représentation des connaissances

## Base de connaissance

Afin d’effectuer notre expertise, nous avons modéliser notre base de connaissances en nous basant sur le Guide Hachette des Vins 2016 (http://www.hachette-vins.com/). Nous avons donc commencé par saisir à la main nos connaissances en les structurant sous forme de a-list de la façon suivante :

* *((Nom CARRUADES\_DE\_LAFITE) (Prix 105) (Annee 2009) (Couleur ROUGE) (Petillant NON) (Geographie BORDELAIS) (Appellation PAUILLAC) (Bio NON) (Note 2) (Medaille NON) (Plat AU\_VERRE VIANDE\_NOIRE VIANDE\_ROUGE VIANDE\_BLANCHE) (Garde 0) (Cepage CABERNET))*

Malheureusement, il n’était pas possible de travailler correctement avec une telle structure. Cependant afin de les structurer correctement, nous devions parser le fichier.txt générer à la main pour obtenir un autre format transitoire. Pour effectuer cela, nous avons utilisé le cours de SR01 sur le python afin de créer un script faisant le travail pour nous. Nous avons donc obtenu un format de la forme :

* *CARRUADES\_DE\_LAFITE,105,2009,ROUGE,NON,BORDELAIS,PAUILLAC,NON,2,NON,AU\_VERRE VIANDE\_NOIRE VIANDE\_ROUGE VIANDE\_BLANCHE,0,CABERNET,*

Afin d’aboutir à notre format de donnée, nous avons de nouveau fait un script en python qui nous a générer les 176 vins de notre base de connaissance comme suit :

* *(setq vin1 (make-instance '$vin :nom "CARRUADES\_DE\_LAFITE" :prix 105 :annee 2009 :couleur "ROUGE" :petillant "NON" :geographie "BORDELAIS" :appellation "PAUILLAC" :bio "NON" :note 2 :medaille "NON" :plat "AU\_VERRE VIANDE\_NOIRE VIANDE\_ROUGE VIANDE\_BLANCHE" :garde "NON" :cepage "CABERNET"))*

Remarque : Le code des scripts sera joint dans un fichier.py

## Base de fait

Nous avons chois de représenter la base de fait « \*BF\* » comme une liste de faits *(fait\_1 fait\_2…)*, où chaque fait correspond à un couple (attribut valeur). Exemple :

* *((GARDE "NON") (MEDAILLE "NON") (NOTE 2)…)*

## Base de règles

La base de règles « \*BR\* » est représentée sous forme de liste de règles (R1R2...RN). Une règle R contient une liste de critères et un nom de vin. Exemple :

* *R1 : (((PRIX 105) (ANNEE 2009) (COULEUR "ROUGE") (PETILLANT "NON") (GEOGRAPHIE "BORDELAIS") (APPELLATION "PAUILLAC") (BIO "NON") (NOTE 2) (MEDAILLE "NON") (GARDE "NON") ...) "CARRUADES\_DE\_LAFITE")*

Une règle n’est déclenchée que si que si ses critères sont satisfaits (relativement à la base de fait). Seulement alors, la base de fait est mise à jour avec les critères. Une fois la base de fait remplie, on la compare avec la base de connaissance afin d’aboutir à la liste des vins correspondant à nos critères.

## Jeux d’essais

* Vous voulez un vin rouge entre 10€ et 20€

*\*BF\* :((GARDE "NON") (MEDAILLE "NON") (NOTE 0) (BIO "NON") (PETILLANT "NON") (ANNEEMAX 2015) (ANNEEMIN 1990) (PRIXMAX 20)*

*(PRIXMIN 10) (GEOGRAPHIE ("CHAMPAGNE" "BORDELAIS" "BOURGOGNE")) ...)*

* Vous voulez un vin bio

*\*BF\* :((GARDE "NON") (MEDAILLE "NON") (NOTE 0) (BIO "OUI") (PETILLANT "NON") (ANNEEMAX 2015) (ANNEEMIN 1900) (PRIXMAX 1000)*

*(PRIXMIN 0) (GEOGRAPHIE ("CHAMPAGNE" "BORDELAIS" "BOURGOGNE")) ...)*

* Vous voulez un vin blanc pétillant de 2013 de bourgogne pour 5€

*\*BF\* :* *:((GARDE "NON") (MEDAILLE "NON") (NOTE 0) (BIO "NON") (PETILLANT "NON") (ANNEE 2013) (PRIX 5) (GEOGRAPHIE ("BOURGOGNE")) (COULEUR ("BLANC"))*

## Arbre de déduction

Arbre de déduction : tous les noeuds sont des règles.

# Programmation du Système expert

## Représentation Lisp

## Moteur d’inférences

En chainage avant, toutes les règles possibles sont déclenchées jusqu’à atteinte du but. Ou l’impossibilité́ de l’atteindre: cela se produit s’il n’y a plus de règles à déclencher avant que le but ne soit atteint.

Ce modèle est intéressant car il permet de visualiser la progression du raisonnement jusqu’au but. Cependant elle effectue un trop grand nombre d’opérations : toutes les branches de l’arbre possible sont testées. Afin de réduire cette complexité, notre moteur sort dès qu’une règle ne peut pas être appliquée.

* *CheckR :*

*(defun CheckR (R)*

*//Le moteur reçoit une règle pour la tester*

*(if (not (member (cadr R) \*Resultat\*))*

*//Vérifie qu’un vin n’est pas déjà dans la base de résultat. Optimisation pour réduire la complexité*

*(let ((flag T) (Conditions (car R)))*

*//Initialisation du drapeau d’arrêt à true*

*//Initialisation deCconditions avec toutes les règles de R1*

*(loop for C in Conditions while (equal flag T) do*

*//Boucle sur chaque sous règle de Condition. Cette boucle opère tant que le drapeau est à true. Optimisation de la complexité.*

*(if (null (Verifier C)) (setq flag NIL))*

*//Appel à Verifier qui peut modifier le drapeau si une sous règle est fause en le passant ) NIL*

*)*

*(if (equal flag T) (addVin (cadr R))))*

*//Ajout du vin dans résultat*

*)*

*)*

## Fonctions de services

### Fonctions principales

* *TrouveVin :*

*(defun TrouveVin()*

*(setq \*Resultat\* NIL)*

*//Initialisation de la base de résultat*

*(InitBF)*

*//Initialisation de la base de faits*

*(loop for R in \*BR\* do (CheckR (eval R))*

*)*

*(Affichage \*Resultat\*)*

*)*

Cette fonction est le cœur du programme. En effet elle va faire fonctionner le moteur d’inférence sur chaque règle de \*BR\*.

Nous avons choisi de créer une base de résultat afin de faciliter le fonctionnement de notre moteur d’inférence et l’affichage des résultat.

* *addVin :*

*(defun addVin (vin)*

*(push vin \*Resultat\*)*

*)*

Cette fonction ajoute le vin répondant aux critères dans la base de résultat. *InitBF*

* *InitBF :*

*Le code de la fonction est en annexe afin d’alléger la lecture du rapport.*

Cette fonction va permettre de renseigner quelques critères de base de notre vin dans la base de fait. Nous avons du utiliser *« (push (list 'couleur listeCouleur) \*BF\*)) »* afin que l’utilisateur puisse rentrer un choix multiple de couleurs (de même pour la géographie).

### Fonctions de gestion de la base de fait

* *Vérifier :*

*(defun Verifier (C)*

*(cond*

*((equal (car C) 'prix) (AskPrice C))*

*((equal (car C) 'annee) (AskMillesime C))*

*((equal (car C) 'petillant) (AskPetillant C))*

*((equal (car C) 'bio) (AskBio C))*

*((equal (car C) 'note) (AskNote C))*

*((equal (car C) 'medaille) (AskMedaille C))*

*((equal (car C) 'garde) (AskGarde C))*

*((equal (car C) 'couleur) (VerifCouleur C))*

*((equal (car C) 'geographie) (VerifGeographie C))*

*(T)*

*))*

La fonction Vérifier reçoit chaque sous règle des règles et retourne True si chacune des sous règles répond aux critères. Dans ce cas, le vin sera ajouté à la base de résultat. Sinon on passe à la règle suivante

* *Fonction Ask :*

*Le code des fonctions Ask est en annexe*

Toutes les fonctions sont construites sur le même schéma. Si le critère n’a pas encore été renseigné, alors la fonction pose la question relative à elle même sinon elle vérifie si le critère est égal à la sous règle. Si ce n’est pas le cas, elle va renvoyer NULL et faire arrêter vérifier et donc passer à la règle suivante.

* *Fonctions Vérification des couleurs et des géographies :*

*Le code des fonctions VerificationCouleur et Verification géographie est en annexe*

Ces fonctions sont basées sur le même principe que les fonctions ask sauf que comme une couleur ou une géographie peut être multiples, elle permet d’itérer sur plusieurs attributs.

### Fonction d’affichage

* *Affichage :*

*(defun Affichage(result)*

*(loop for NameVinRes in result do*

*(loop for vin in list\_wine do*

*(if (equal NameVinRes (?nom vin))*

*(progn*

*(format t "Nom du Vin: ~a" (?nom vin))*

*(format t "~%Caracteristiques:~%Prix:~a"(?prix vin))*

*(format t " Annee:~a"(?annee vin))*

*(format t " Note:~a"(?note vin))*

*(format t " Couleur:~a"(?couleur vin))*

*(format t " Petillant:~a"(?petillant vin))*

*(format t " Geographie:~a"(?geographie vin))*

*(format t " Appellation:~a"(?appellation vin))*

*(format t " Bio:~a"(?bio vin))*

*(format t " Medaille:~a"(?medaille vin))*

*(format t " Garde:~a"(?garde vin))*

*(format t " Cepage:~a~%"(?cepage vin))*

*(format t "Plat en accord: ~a~%~%" (?plat vin))*

*)))))*

La fonction d’affichage reçoit en paramètre la base de résultat comprenant le nom des vins ayant répondu aux règles. Elle utilise une double boucle qui va comparer chaque vin de « résultat » à notre base de connaissance afin de pouvoir en afficher les attributs. L’affichage utilise la fonction format au lieu de print. Cette dernière permet d’afficher du texte et des valeurs en fonctions d’un accesseur.

### Programme à exécuter

* Main.lisp :

*(load "/Users/SIMON/Desktop/IA01\_TP03/Rendu/BaseVin.lisp")*

*(load "/Users/SIMON/Desktop/IA01\_TP03/Rendu/Fonctions.lisp")*

*(TrouveVin)*

Les fonctions load utilisent le chemin absolu afin de trouver les fichiers à charger. Elles offrent un gain de temps considérable pour charger toutes les fonctions et bases de notre programme. Cependant, elles ont soulevé un problème lors de leur première exécution. En effet l’encodage utf-8 n’étant pas spécifié tous les accents et caractères spéciaux n’étaient pas correctement affichés. N’ayant pas réussi à spécifier l’encodage des fichiers, nous avons du remplacer chaque accent et caractère spécial par une version « basique ».

Afin de charger main.lisp il faut exécuter la commande suivante :

***(load "/Users/SIMON/Desktop/IA01\_TP03/Rendu/main.lisp")***

Une fois les fonctions de chargement exécutées, on appelle la fonction principale TrouveVin qui lance l’exécution.

## Résultats

Dans cette partie nous allons résoudre notre problème initial su cadeau de noël en détaillant chacune des opérations.

### Les différentes étapes de fonctionnement

* **Étape n°1 :**

*"Precision sur la/les couleur(s) du vin recherche(s) :"*

*"Le vin peut il etre (Repondez par oui ou non) :"*

*"Rouge ?:" oui*

*"Blanc ?:" non*

*"Rose ?:" non*

* **Étape n°2 :**

"Concernant la provenance du Vin:"

"Le vin peut il provenir de (Repondez par oui ou non) :"

"Bourgogne ?:" oui

"Bordeaux ?:" oui

"Champagne ?:" non

* **Étape n°3 :**

Renseignement sur le prix"

"Souhaitez vous un prix precis (tres restrictif) plutot qu'une fourchette ?:(oui/non)" non

"Quel prix Minimum (en Euros) ?:" 10

"Quel prix Maximum (en Euros) ?:" 30

* **Étape n°4 :**

"Renseignement sur l'annee"

"Souhaitez vous une annee precis (tres restrictif) plutot qu'une fourchette ?:(oui/non)" non

"Quelle annee Minimum (YYYY) ?:" 1998

"Quel annee Maximum (YYYY) ?:" 2009

* **Étape n°5 :**

"Voulez vous un vin petillant ?(oui/non):" non

* **Étape n°6 :**

"Vin Bio"

"Voulez vous un vin bio:(oui/non)" non

* **Étape n°7 :**

"Note du Vin"

"Choisissez une note minimale:(0/1/2/3)" 2

* **Étape n°8 :**

"Voulez vous un vin medaille:(oui/non)" non

* **Étape n°9 :**

"Garde du vin"

"Voulez vous un vin de garde?:(non, tapez 0/ oui, tapez 1)" 0

* **Valeur finale de \*BF\* :**

*((GARDE "NON") (MEDAILLE "NON") (NOTE 2) (BIO "NON") (PETILLANT "NON") (ANNEEMAX 2009) (ANNEEMIN 1998) (PRIXMAX 30)*

*(PRIXMIN 10) (GEOGRAPHIE ("BORDELAIS" "BOURGOGNE")(COULEUR ("ROUGE")))*

* **Valeur finale de \*Resultat\* :**

*("DOMAINE\_DES\_VIGNES\_DES\_DEMOISELLES" "CHATEAU\_CITRAN" "CHATEAU\_LES\_TROIS\_CROIX" "CHATEAU\_PETIT\_GRAVET\_AINE" "CLOS\_LOUIE")*

* **Affichage :**

Nom du Vin: DOMAINE\_DES\_VIGNES\_DES\_DEMOISELLES

Caracteristiques:

Prix:15 Annee:1998 Note:3 Couleur:ROUGE Petillant:NON Geographie:BOURGOGNE Appellation:SANTENAY Bio:NON Medaille:NON Garde:NON Cepage:PINOT\_NOIR CHARDONNAY

Plat en accord: VIANDE\_NOIRE VIANDE\_BLANCHE

Nom du Vin: CHATEAU\_CITRAN

Caracteristiques:

Prix:20 Annee:2000 Note:3 Couleur:ROUGE Petillant:NON Geographie:BORDELAIS Appellation:MONTAGNE\_SAINT\_EMILION Bio:NON Medaille:NON Garde:NON Cepage:CABERNET MERLOT

Plat en accord: VIANDE\_ROUGE VIANDE\_BLANCHE VIANDE NOIRE APERITIF DESSERT

Nom du Vin: CHATEAU\_LES\_TROIS\_CROIX

Caracteristiques:

Prix:20 Annee:2005 Note:2 Couleur:ROUGE Petillant:NON Geographie:BORDELAIS Appellation:FRONSAC Bio:NON Medaille:NON Garde:NON Cepage:CABERNET

Plat en accord: VIANDE\_ROUGE VIANDE\_NOIRE

Nom du Vin: CHATEAU\_PETIT\_GRAVET\_AINE

Caracteristiques:

Prix:25 Annee:2002 Note:3 Couleur:ROUGE Petillant:NON Geographie:BORDELAIS Appellation:SAINT\_EMILION Bio:NON Medaille:NON Garde:NON Cepage:CABERNET MERLOT

Plat en accord: VIANDE\_ROUGE VIANDE\_BLANCHE APERITIF

Nom du Vin: CLOS\_LOUIE

Caracteristiques:

Prix:20 Annee:2004 Note:2 Couleur:ROUGE Petillant:NON Geographie:BORDELAIS Appellation:CASTILLON Bio:NON Medaille:NON Garde:NON Cepage:CABERNET MERLOT

Plat en accord: VIANDE\_BLANCHE

### Analyse des étapes

Au lancement du programme nous chargeons la base de fait avec les critères sur la couleur et la géographie (étape 1 à 2). Puis notre programme va commencer à itérer sur chacune des règles grâce au moteur d’inférence. Des lors, notre base de résultat étant vide, nous allons itérer sur chacune des sous règles de R1. De même, notre base de fait ne contient aucun renseignement sur les étapes 3 à 9. Il va donc interroger l’utilisateur afin de compléter sa base de fait. Une fois celle ci remplie, le moteur va itérer de façon identique pour toutes les règles. Il va vérifier si elles remplissent les critères. Si tel est le cas, alors des vins seront ajoutés dans \*Resultat\*.

Ainsi l’on peut constater la façon dont est remplie la base de fait et la base de règle. Nous obtenons donc en sortie une ou plusieurs solutions (ici le cas) à notre problème. L’utilisateur n’a plus qu’à choisir le meilleur vin pour son père. Ici nous prendrons le Château Citran

# Conclusion

Ce TP nous a permis de comprendre l'importante de la recherche et de la structure des données du système. Par ailleurs nous avons réalisé qu'étant donnée l'exhaustivité des cas du monde réel (malgré le fait que nous ayons encadré le sujet juste pour trois régions) les possibilités de vins sont incommensurablement grandes. Un tel système est par conséquent très coûteux à produire et maintenir. Les systèmes experts demeurent donc un bon assistant mais ne peuvent pas remplacer l'homme.

En résumé, ce TP nous a permis d'essayer de traiter un problème réel, complexe et dans apercevoir les limites.

# Annexe

* *InitBF :*

*(defun InitBF ()*

*(setq \*BF\* NIL) ;Initialisation a NIL de notre base de faits*

*(let ((listeCouleur NIL)) ;Creation d’une variable temporaire pour l’insertion dans la base de fait*

*; Demande de la couleur du vin a l'utilisateur*

*(print "Precision sur la/les couleur(s) du vin recherche(s) :")*

*(print "Le vin peut il etre (Repondez par oui ou non) :")*

*(print "Rouge ?:")*

*(if (eq (read) 'oui)*

*(push "ROUGE" listeCouleur))*

*(print "Blanc ?:")*

*(if (eq (read) 'oui)*

*(push "BLANC" listeCouleur))*

*(print "Rose ?:")*

*(if (eq (read) 'oui)*

*(push "ROSE" listeCouleur))*

*(push (list 'couleur listeCouleur) \*BF\*)) ;Permet l’insertion de multiples couleurs*

*(let ((listeGeographie NIL))*

*(print "Concernant la provenance du Vin:")*

*(print "Le vin peut il provenir de (Repondez par oui ou non) :")*

*(print "Bourgogne ?:")*

*(if (eq (read) 'oui)*

*(push "BOURGOGNE" listeGeographie))*

*(print "Bordeaux ?:")*

*(if (eq (read) 'oui)*

*(push "BORDELAIS" listeGeographie))*

*(print "Champagne ?:")*

*(if (eq (read) 'oui)*

*(push "CHAMPAGNE" listeGeographie))*

*(push (list 'geographie listeGeographie) \*BF\*));Permet l’insertion de multiples geographie*

*)*

* *AskPrice :*

*(defun AskPrice (C)*

*(if (not (or (assoc 'prix \*BF\*) (assoc 'prixMin \*BF\* )))*

*;on verifie si aucun prix n’a été renseigné dans la base de fait*

*(progn*

*(print "Renseignement sur le prix" )*

*(print "Souhaitez vous un prix precis (tres restrictif) plutot qu'une fourchette ?:(oui/non)")*

*(if (eq (read) 'oui)*

*;l’instruction (read) attend la saisie d’une valeur par l’utilisateur*

*;Si oui alors un prix fixe*

*(progn*

*(print "Quel prix (en Euros)?:")*

*(push (list 'prix (read)) \*BF\*)*

*(AskPrice C)*

*)*

*(progn*

*;Si non intervalle de prix*

*(print "Quel prix Minimum (en Euros) ?:")*

*(push (list 'prixMin (read)) \*BF\*)*

*(print "Quel prix Maximum (en Euros) ?:")*

*(push (list 'prixMax (read)) \*BF\*)*

*(AskPrice C)*

*;On interroge de nouveau pour verifier la bonne saisie des prix*

*)*

*)*

*)*

*;Le/Les prix on deja ete renseignes, on verifie avec la condition*

*(progn*

*(let ((PriceOK NIL))*

*(if (assoc 'prix \*BF\*)*

*;Si une sous liste de BF contient le mot clef prix, on recupere sa/ses valeurs*

*(progn (if (= (cadr C) (cadr (assoc 'prix \*BF\*))) (setq PriceOK T))*

*)*

*(progn (if (AND (<= (cadr C) (cadr (assoc 'prixMax \*BF\*))) (>= (cadr C) (cadr (assoc 'prixMin \*BF\*))))*

*(setq PriceOK T))))*

*PriceOK)*

*)*

*)*

*)*

**L’ensemble des fonctions ASK fonctionne sur le même principe. Ainsi nous n’expliquerons pas en détail leur code**

* *AskMillesime :*

*(defun AskMillesime (C)*

*(if (not (or (assoc 'annee \*BF\*) (assoc 'anneeMin \*BF\* )))*

*(progn*

*(print "Renseignement sur l'annee" )*

*(print "Souhaitez vous une annee precis (tres restrictif) plutot qu'une fourchette ?:(oui/non)")*

*(if (eq (read) 'oui)*

*(progn*

*(print "Quelle annee (format YYYY) ?:")*

*(push (list 'annee (read)) \*BF\*)*

*(AskMillesime C)*

*)*

*(progn*

*(print "Quelle annee Minimum (YYYY) ?:")*

*(push (list 'anneeMin (read)) \*BF\*)*

*(print "Quel annee Maximum (YYYY) ?:")*

*(push (list 'anneeMax (read)) \*BF\*)*

*(AskMillesime C)*

*)*

*)*

*)*

*;Le/Les annee(s) on deja ete renseignes, on verifie avec la condition*

*(progn*

*(let ((YearOK NIL))*

*(if (assoc 'annee \*BF\*)*

*(progn (if (= (cadr C) (cadr (assoc 'annee \*BF\*))) (setq YearOK T))*

*)*

*(progn (if (AND (<= (cadr C) (cadr (assoc 'anneeMax \*BF\*))) (>= (cadr C) (cadr (assoc 'anneeMin \*BF\*))))*

*(setq YearOK T))))*

*YearOK)*

*)*

*)*

*)*

* *AskPetillant :*

*(defun AskPetillant (C)*

*(if (not (assoc 'petillant \*BF\*))*

*(progn*

*(print "Voulez vous un vin petillant ?(oui/non):" )*

*(if (eq (read) 'oui)*

*(progn*

*(push (list 'petillant "OUI") \*BF\*)*

*(AskPetillant C)*

*)*

*(progn*

*(push (list 'petillant "NON") \*BF\*)*

*(AskPetillant C)*

*)*

*)*

*)*

*;Petillant deja renseigne*

*(progn*

*(let (( PetillantOK NIL))*

*(if (equal (cadr C) (cadr (assoc 'petillant \*BF\*))) (setq PetillantOK T))*

*PetillantOK)*

*)*

*)*

*)*

* *AskBio :*

*(defun AskBio (C)*

*(if (not (assoc 'bio \*BF\*))*

*(progn*

*(print "Vin Bio" )*

*(print "Voulez vous un vin bio:(oui/non)")*

*(if (eq (read) 'oui)*

*(push (list 'bio "OUI") \*BF\*)*

*(push (list 'bio "NON") \*BF\*)*

*)*

*(Askbio C)*

*)*

*;Bio est deja renseigne*

*(progn*

*(let ((BioOK NIL))*

*(if (equal (cadr C) (cadr (assoc 'bio \*BF\*)))*

*(setq BioOK T)*

*)*

*BioOK))))*

* *AskNote :*

*(defun AskNote (C)*

*(if (not (assoc 'note \*BF\*))*

*(progn*

*(let ((entree 0))*

*(print "Note du Vin" )*

*(print "Choisissez une note minimale:(0/1/2/3)")*

*(setq entree (read))*

*(cond*

*((= entree 0) (push (list 'note entree) \*BF\*))*

*((= entree 1) (push (list 'note entree) \*BF\*))*

*((= entree 2) (push (list 'note entree) \*BF\*))*

*((= entree 3) (push (list 'note entree) \*BF\*))*

*)*

*)*

*(AskNote C)*

*)*

*;La note est deja renseigne*

*(progn*

*(let ((NoteOK NIL))*

*(if (>= (cadr C) (cadr (assoc 'note \*BF\*)))*

*(setq NoteOK T)*

*)*

*NoteOK))))*

* *AskMedaille :*

*(defun AskMedaille (C)*

*(if (not (assoc 'medaille \*BF\*))*

*(progn*

*(print "Voulez vous un vin medaille:(oui/non)")*

*(if (eq (read) 'oui)*

*(push (list 'medaille "OUI") \*BF\*)*

*(push (list 'medaille "NON") \*BF\*)*

*)*

*(AskMedaille C)*

*)*

*;Medaille est deja renseigne*

*(progn*

*(let ((MedailleOK NIL))*

*(if (equal (cadr C) (cadr (assoc 'medaille \*BF\*)))*

*(setq MedailleOK T)*

*)*

*MedailleOK))))*

* *AskGarde :*

*(defun AskGarde (C)*

*(if (not (assoc 'garde \*BF\*))*

*(progn*

*(print "Garde du vin" )*

*(print "Voulez vous un vin de garde?:(non, tapez 0/ oui, tapez 1)")*

*(if (= (read) 0) (push (list 'garde "NON") \*BF\*)*

*(push (list 'garde "OUI")\*BF\*)*

*)*

*(AskGarde C)*

*)*

*;La garde est deja renseigne*

*(progn*

*(let ((GardeOk NIL))*

*(if (equal (cadr C) (cadr (assoc 'garde \*BF\*)))*

*(setq GardeOk T)*

*)*

*GardeOk*

*)*

*)*

*)*

*)*

* *VerifCouleur :*

*(defun VerifCouleur (C)*

*(let ((CouleurOk NIL) (listeCouleur (cadr (assoc 'couleur \*BF\*))) )*

*;On charge listeCouleur avec l’ensemble des couleurs entrées par l’utilisateur*

*(loop for couleur in listeCouleur while(null CouleurOk) do*

*;On itère sur chacune des couleurs*

*(if (equal (cadr C) couleur) (setq CouleurOk T)))*

*CouleurOk*

*)*

*)*

**Le code de VerifGeographie est quasi identique, nous ne détaillons pas son fonctionnement**

* *VerifGeographie :*

(defun VerifGeographie (C)

(let ((GeographieOk NIL) (listeGeo (cadr (assoc 'geographie \*BF\*))) )

(loop for geo in listeGeo while(null GeographieOk) do

(if (equal (cadr C) geo) (setq GeographieOk T)))

GeographieOk

)

)