```
##############################################################################
# MICREXP *            BE Intelligence Artificielle # 3 Annee ITR CENTRALE #
##############################################################################
#                      FONCTIONS DE BASE                                     #
##############################################################################
Nom de la regle Regle.
Liste des premisses de la regle Regle.
Liste des conclusions de la regle Regle.
Affectation a T de la propriete Valeur de la proposition Prop.
Lecture de la propriete Valeur de la proposition Prop.
Suppression de la propriete Valeur de la proposition Prop.

Listes circulaires des proprietes Vraie et Fausse.
Listes circulaires des proprietes Recherchee et Appliquee.

##############################################################################
#                  COMPILATION DE LA BASE DE REGLES                          #
##############################################################################
Initialisation de la base de regles et construction de la base de propositions
 Chargement de la base de regles.
 Test de la presence de la base de regles: Regles.
 Demande de passage en mode trace des moteurs d'inferences.

 Test de la presence de la base de propositions.
  Sinon construction de la base de propositions a partir de la base de regles

 Initialisation des proprietes Vraie, Fausse et Recherchee des propositions,
 et de la propriete Appliquee des regles.

Constructeur de la base de propositions.


Extraction des propositions a partir de la base de regles.
 Affichage en mode trace des regles de la base de regles.
  Construction de la base de propositions avec les conclusions des regles.
   La propriete En_conclusion contient la liste des regles dont la
   proposition intervient dans la conclusion.
  Construction de la base de propositions avec les premisses des regles.


##############################################################################
#                      CHAINAGE ARRIERE                                      #
##############################################################################
Moteur d'inference en chainage arriere a partir d'une proposition P.
 Initialisation des bases. Choix du mode de deduction.
 Memorisation du mode de deduction dans Determ.
 Chainage arriere a partir de la proposition P.
 affichage des resultats.

Coeur du moteur de en chainage arriere.
 Si la proposition a deja ete recherchee alors T ou NIL suivant Vraie(P).
 Si la proposition est terminale (non en conclusion) elle est alors
  non demontrable et on questionne. Sinon on infere cette proposition.

Questionneur de proposition.
 Question.
 Variable locale contenant la reponse a la question.
   (o oui y yes) Proposition a Vraie et valeur de retour  = T.
   (n non no)    Proposition a Fausse et valeur de retour = NIL.
    ?                          valeur de retour = NIL.
 Tentative suplementaire si reponse incorrecte.

Deducteur a partir de la proposition P sur une liste de regles Lregl.
```

```
;##############################################################################
; MICREXP.LL #          MICRO-SYSTEME-EXPERT         # Christian Jousselin #
;##############################################################################
;                      FONCTIONS DE BASE
;                                                    2/12/90
(DE nom_de       (Regle)      (CAR Regle))
(DE premisse_de  (Regle)      (CADDR Regle))
(DE conclusion_de (Regle)     (CADR(CDDDR Regle)))
(DE ->           (Prop Valeur) (PUTPROP Prop T Valeur))
(DE ?            (Prop Valeur) (GET Prop Valeur))
(DE x            (Prop Valeur) (REMPROP Prop Valeur))

(SETQ Vraies     (CIRLIST 'Vraie)     Fausses    (CIRLIST 'Fausse)
    Recherchees (CIRLIST 'Recherchee) Appliquees (CIRLIST 'Appliquee))
;                                                   OK
;##############################################################################
;                  COMPILATION DE LA BASE DE REGLES
;                                                   3/12/90
(DE initialiser ()
 (LOAD "regles") (COND ((NOT (BOUNDP 'Regles))
      (PRINT ">>>Vous ne m'avez pas donne de regles.<<<")(ERROR)))
 (PRINT "Voulez vous que j'affiche mes inferences ?")
 (IF (MEMBER (READ) '(o oui y yes)) (SETQ Trace T) (SETQ Trace NIL))
 (COND ((NOT (BOUNDP 'Propositions))
      (SETQ Propositions NIL) (MAPC 'extraire_de Regles)
      (PRINT ">>>Base de connaissances chargee.<<<")(TERPRI)))
 (MAPC 'x Propositions Fausses)(MAPC 'x Propositions Recherchees)
 (MAPC 'x Propositions Vraies)(MAPC 'x (MAPCAR 'nom_de Regles) Appliquees))

(DE construire_avec (P)
 (IFN (MEMBER P Propositions) (SETQ Propositions (CONS P Propositions))))

(DE extraire_de (R)
 (IF Trace (PRINT R))
 (MAPC '(LAMBDA (P) (construire_avec P)
  (PUTPROP P (CONS R (GET P 'En_conclusion)) 'En_conclusion))
  (conclusion_de R))
 (MAPC 'construire_avec (premisse_de R)))

;                                                   OK
;##############################################################################
;                      CHAINAGE ARRIERE
;                                                   4/12/90
(DF deduire_de (P)
   (initialiser) (PRINT "Mode de deduction deterministe? [o,n] ")
   (IF (MEMBER (READ) '(o oui y yes)) (SETQ Determ T) (SETQ Determ NIL))
   (tester P)
   (conclure))

(DE tester (P)
 (COND ((? P 'Recherchee) (IF (? P 'Vraie) T NIL))
      ((NULL (? P 'En_conclusion)) (-> P 'Recherchee) (questionner_sur P))
      (T (-> P 'Recherchee) (inferer P (? P 'En_conclusion)))))

(DE questionner_sur (P)
 (PRINT "La proposition <"P"> est-elle vraie? [o,n,?]")
 (LET ((Reponse (READ)))
  (COND ((MEMBER Reponse '(o oui y yes)) (-> P 'Vraie)  T)
      ((MEMBER Reponse '(n non no))    (-> P 'Fausse) NIL)
      ((EQ Reponse '?)                 NIL)
      (T (PRINT ">>>Repondez oui non ou ?<<<") (questionner_sur P)))))

(DE inferer (P Lregl)
```

Si la liste est vide la deduction est fausse.
Sinon on essaye les regles une a une.

Essais d'une regle.
Commentaire en mode trace.
Si la regle a deja ete appliquee : Fin.
Sinon test des premisses de cette regle:
Si test OK => les conclusions de cette regles sont Vraies.
Sinon => les conclusions de cette regles sont Fausses si Determ.
Et la regle est Appliquee.

Test des premisses.
Point d'arret de la recursivite : liste vide des premisses.
Si au moins une proposition est Fausse => NIL.

Rendre Vraies les conclusions d'une regle appliquee.
Conclusions a Vraies. Trace de la deduction en mode trace.

Rendre Fausses les conclusions d'une regle appliquee.
Conclusions a Fausses. Trace de la deduction en mode trace.

Traceur de deduction Vraie.

Traceur de deduction Fausse.

##
CHAINAGE AVANT
##
Moteur d'inference en chainage avant a partir d'un ensemble de propositions.
Initialisation et mise a Vraies des propositions de depart de l'induction.
Coeur du moteur en chainage avant.
Affichage des inductions.

Inducteur a partie d'une regle.
Si la regle a deja ete appliquee => NIL.
Test des premisses de cette regle.
Si applicable => Appliquee.
Commentaire en mode trace.

Rendre Vraies les conclusions de cette regle,et => T Sinon => NIL.

##
CONCLUSION
##
Presentation des conclusions sur la base des propositions.

Affichage de la conclusion sur une proposition.

```
  (COND ((NULL Lregl) NIL)
        (T (essayer_regle (CAR Lregl))(COND ((? P 'Vraie) T)
                                    (T (inferer P (CDR Lregl)))))))

(DE essayer_regle (R)
  (COND (Trace (PRINT "J'essaye la regle ")(TERPRI)(PRINT R)(TERPRI)))
  (COND ((? (nom_de R) 'Appliquee) NIL)
        (T (IF (tester_premisse (premisse_de R))
               (rendre_vraie (conclusion_de R))
               (IF Determ (rendre_fausse (conclusion_de R))))
          (-> (nom_de R) 'Appliquee))))

(DE tester_premisse (Lprop)
  (COND ((NULL Lprop) T)
        (T (AND (tester (CAR Lprop)) (tester_premisse (CDR Lprop))))))

(DE rendre_vraie (Lprop)
  (MAPC '-> Lprop Vraies)  (IF Trace (MAPC 'tracer_V Lprop)))

(DE rendre_fausse (Lprop)
  (MAPC '-> Lprop Fausses) (IF Trace (MAPC 'tracer_F Lprop)))

(DE tracer_V (P) (PRINT "=>J en deduis que <"P"> est vraie."))))

(DE tracer_F (P) (PRINT "=>J en deduis que <"P"> est fausse."))))

;                                                  OK
;#############################################################################
;                        CHAINAGE AVANT
;                                                  5/12/90
(DF induire_de (Lprop)
  (initialiser) (MAPC '-> Lprop Vraies)
  (WHILE (ANY 'appliquer Regles))
  (conclure))

(DE appliquer (R)
  (COND ((? (nom_de R) 'Appliquee) NIL)
        ((NOT(MEMBER NIL (MAPCAR '? (premisse_de R) Vraies)))
         (-> (nom_de R) 'Appliquee)
         (COND (Trace
                (TERPRI)(PRINT "J'applique la regle :")(PRINT R)(TERPRI)))
         (rendre_vraie (conclusion_de R)) T)))

;                                                  OK
;#############################################################################
;                          CONCLUSION
;                                                  2/12/90
(DE conclure ()
  (TERPRI 3) (PRINT "=>Voici ce que j'en conclut :")
  (TERPRI)   (MAPC 'afficher Propositions)
  (TERPRI)   "C'est tout ce que je peux vous dire ....")

(DE afficher (P)
  (COND ((? P 'Vraie)  (PRINT "La proposition < "P" > est vraie."))
        ((? P 'Fausse) (PRINT "La proposition < "P" > est fausse."))
        (Trace         (PRINT "Je ne peux rien dire quant a < "P" >"))))

;                                                  OK
;#############################################################################
```