PROLOG

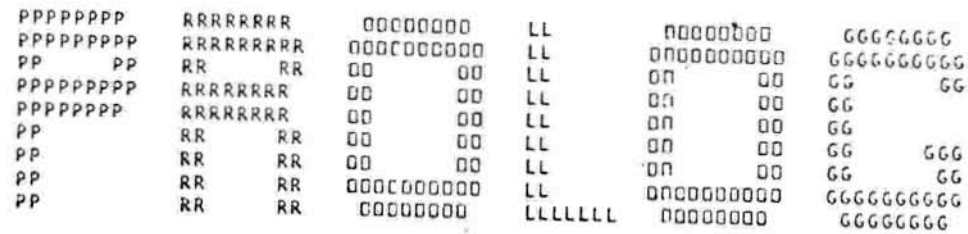
MANUEL DE REFERENCE ET D'UTILISATION

PH. ROUSSEL

GROUPE D'INTELLIGENCE ARTIFICIELLE
UER MARSEILLE-LUMINY

SEPTEMBRE 1975

GROUPE D'INTELLIGENCE ARTIFICIELLE
MARSEILLE-LUMINY



MANUEL D'UTILISATION

PH.ROUSSEL

SEPTEMBRE 1975

TABLE DES MATIERES

INTRODUCTION

LA DEFINITION D'UN LANGAGE COMME PROLOG EST NEE D'UN BESOIN DE DISPOSER D'UN FORMALISME EVOLUE PERMETTANT EN MEME TEMPS LA DESCRIPTION ET LA RESOLUTION DE PROBLEMES SUR CALCULATEUR.CE LANGAGE DE PROGRAMMATION PEUT DONC ETRE EGALEMENT CONSIDERE COMME UN LANGAGE DE FORMALISATION . LA LOGIQUE DES PREDICATS DU PREMIER ORDRE AYANT SERVI DE SUPPORT THEORIQUE A SA REALISATION.

LES OUTILS FONDAMENTAUX DONT DISPOSE LE PROGRAMMEUR A TRAVERS CE LANGAGE SONT:
-DES OBJETS A STRUCTURE ARBORESCENTE
-UN MECANISME EVOLUE QUI PERMET LES SUBSTITUTIONS, LES RECONNAISSANCES PAR CAS("PATTERN-MATCHING"), DES APPELS A DES SOUS-PROGRAMMES
-LE NON-DETERMINISME DE L'EXECUTION DES SOUS-PROGRAMMES, GERE PAR UN MECANISME DE RETOUR EN ARRIERE("BACKTRACKING")
-LA RECURSIVITE DANS LA DEFINITION DES PROGRAMMES

LE CHAPITRE I DU MANUEL EST CONSACRE A LA DESCRIPTION SYNTAXIQUE ET SEMANTIQUE DES OBJETS.

LE CHAPITRE II DECRIT LA STUCTURE ET L'EXECUTION DU CONTROLE.

LE CHAPITRE III TRAITE DES MECANISMES DE BASES IMPLANTES AU NIVEAU DE L'INTERPRETEUR, ET QUI PERMETTENT:
-LE TRAITEMENT DES ENTREES-SORTIES
-LA GESTION DU DICTIONNAIRE DES UNITES SYNTAXIQUES
-LA MODIFICATION DYNAMIQUE DES SOUS-PROGRAMMES ET DU CONTROLE
-LE TRAITEMENT DES ENTIERS ET DES CARACTERES.

DANS LE CHAPITRE IV,LE SUPERVISEUR QUI GERE LES TRAVAUX DE L'UTILISATEUR EST DECRIT, AINSI QUE LES POSSIBILITES QU'IL OFFRE POUR LES ENTREES-SORTIES.

LE CHAPITRE V DONNE QUELQUES RESULTATS THEORIQUES SUR LES NOTIONS DE SEMANTIQUE D'UN PROGRAMME, ET UNE METHODOLOGIE DE BASE POUR L'UTILISATION DE PROLOG DANS LA RESOLUTION DE PROBLEMES.

LA SYNTAXE DES DIFFERENTES UNITES DECRITES DANS LE MANUEL SERA EXPRIMEE EN "B-N-F".
LA REDACTION DE CE MANUEL EST LE RESULTAT DE RECHERCHES EFFECTUEES DANS LE CADRE D'UN CONTRAT C.N.R.S (ATP N= 7299/04).

7

CHAP I DEFINITION ET STUCTURE DES OBJETS

****
* I.1 STRUCTURE DES OBJETS(TERMES) *
****

LES OBJETS MANIPULES PAR UN PROGRAMME PROLOG SONT APPELES TERMES
CES TERMES ONT UNE STRUCTURE D'ARBRE ORIENTE. CHAQUE NOEUD DE L'ARBRE ETANT ETIQUETTE
-SOIT PAR UNE VARIABLE
-SOIT PAR UN SYMBOLE FONCTIONNEL.
LES VARIABLES FIGURENT TOUJOURS EN NOEUD TERMINAL D'UN TERME

1.UNITES
========
LES UNITES SONT LES CONSTITUANTS DE BASE PERMETTANT DE DECRIRE UN TERME.

SYNTAXE:

<UNITE>::=<UNITE RESERVEE>\ <UNITE NON RESERVEE>
<UNITE RESERVEE>::= , \ (\ ) \ " \ °
<UNITE NON RESERVEE>::= <NOMBRE>\ <UNITE A-NUM> \<UNITE NON A-NUM>
<NOMBRE>::=<CHIFFRE>\ <CHIFFRE><NOMBRE>
<CHIFFRE>::= 0\ 1\ 2 \ 2 \ .....\ 9
<UNITE A-NUM>::= <LETTRE> \ <LETTRE><SUITE A-NUM>
<LETTRE>::= A \ B \ ..... \ Z
<SUITE A-NUM>::=<A-NUM> \ <A-NUM><SUITE A-NUM>
<A-NUM>::=<LETTRE> \ <CHIFFRE>
<UNITE NON A-NUM>::= ! \ # \ .....

EXEMPLES:

3225 ABCD A45 ?

NOTA-BENE: LES BLANCS NE SERONT PAS PRIS EN COMPTE, SAUF POUR SEPARER DEUX UNITES A-NUM ET/OU NOMBRES, ET DANS LES CHAINES.

2.TERMES SOUS FORME CANONIQUE
=============================

SYNTAXE:

<TERME CANONIQUE>::= <VARIABLE> \ <TERME FONCTIONNEL CANONIQUE> \
<TERME FONCTIONNEL CANONIQUE>::= <SYMBOLE FONCTIONNEL> \
<SYMBOLE FONCTIONNEL>(<LISTE DE TER-CAN>)
<VARIABLE>::=° <UNITE NON RESERVEE>
<SYMBOLE FONCTIONNEL>::= <UNITE NON RESERVEE>
<LISTE DE TER-CAN>::= <TERME CANONIQUE> \
<TERME CANONIQUE>.<LISTE DE TER-CAN>

8

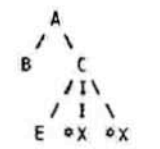
```
PERE(°X.°Y)
AMI(°X.PERE(A.B).°X)
°X
TOTO
RACINE(3.°X)
```

CETTE SYNTAXE DONNE LA FORME STANDARD POUR LA REPRESENTATION DES TERMES.

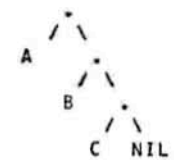
CHAQUE TERME PEUT SE REPRESENTER SANS AMBIGUITE SOUS L'UNE DES DEUX FORMES:
-ECRITURE LINEAIRE FONCTIONNELLE DEFINIF PAR LES REGLES
-ECRITURE SOUS FORME D'ARBRE ORIENTE.

VOICI QUELQUES EXEMPLES DE TERMES SOUS LEUR DEUX FORMES.

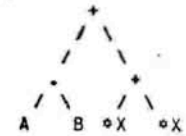
A(B.C(E.°X.°X))



.(A..(B..(C.NIL)))



+(.(A.B).+(°X.°X))



NOTA-BENE:

LES SYMBOLES FONCTIONNELS N'ONT AUCUNE SEMANTIQUE CALCULATOIRE, I-E QU'ILS NE SONT JAMAIS EVALUES PAR LE SYSTEME AVEC UNE SIGNIFICATION OPERATIONNELLE.

3.TERMES SOUS FORME D'EXPRESSIONS
=================================

ALORS QUE LA REPRESENTATION CANONIQUE DONNE LE CODAGE STANDARD DES TERMES. LE SUPERVISEUR OFFRE LA POSSIBILITE D'UTILISER UNE SYNTAXE PLUS RICHE GRACE A L'EMPLOI D'OPERATEURS INFIXES.
L'UTILISATION DE CETTE FORME EST DECRITE DANS IV.2.3

I.2 VARIABLES-SUBSTITUTIONS

LES VARIABLES D'UN TERME, EN ETANT SUBSTITUEES DYNAMIQUEMENT PERMETTENT DES ECHANGES ENTRE LES POINTS D'UN PROGRAMME, ET LA CONSTRUCTION DYNAMIQUE DE NOUVEAUX TERMES.

I.SUBSTITUTION

UNE SUBSTITUTION ELEMENTAIRE EST UN OPERATEUR QUI APPLIQUE A UN TERME PERMET DE SUBSTITUER UNE VARIABLE PAR UN AUTRE TERME PARTOUT OU ELLE APPARAIT.
UNE TELLE SUBSTITUTION SERA REPRESENTEE DANS LES EXPLICATIONS SUR LA SEMANTIQUE PAR UN COUPLE
S=[°X,T]
OU °X EST UNE VARIABLE ET T UN TERME QUELCONQUE NE CONTENANT PAS °X.

SI A EST UN TERME ,A.S DESIGNE LE TERME OBTENU A PARTIR DE A, EN SUBSTITUANT PARTOUT OU ELLE APPARAIT, LA VARIABLE °X.

UNE SUBSTITUTION EST CONSTITUEE A PARTIR DE SUBSTITUTIONS ELEMENTAIRES
S=S1.S2.S2. ....SN
AVEC S1=[°X1,T1] S2=[°X2,T2] ..... SN=[°XN,TN]
OU LES VARIABLES °X1,....,°XN SONT DISTINCTES ET NE FIGURENT DANS AUCUN DES TERMES T1,.....,TN.

CETTE ECRITURE EST LA FORME CANONIQUE D'UNE SUBSTITUTION, ET EST UNIQUE A L'ORDRE PRES DES SUBSTITUTIONS ELEMENTAIRES.

SI A EST UN TERME, LE RESULTAT A.S DE LA SUBSTITUTION APPLIQUEE A CE TERME EST OBTENU EN SUBSTITUANT PARTOUT DANS A LES OCCURENCES DES VARIABLES °X1,....,°XN PAR LES TERMES T1,....,TN.

LE PRODUIT DE DEUX SUBSTITUTIONS S1 ET S2,NOTE S1.S2, EST L'OPERATEUR, QUI APPLIQUE A UN TERME QUELCONQUE A, A POUR RESULTAT (A.S1).S2

CE PRODUIT N'A DE SENS QUE S'IL PEUT S'ECRIRE SOUS FORME CANONIQUE, I-E QUE ,SI AU COURS DES SUBSTITUTIONS, UNE VARIABLE °X N'EST PAS SUBSTITUEE PAR UN TERME CONTENANT °X.

CETTE RESTRICTION EST NECESSAIRE SI L'ON VEUT EVITER DE CREER DES STRUCTURES BOUCLEES.

2.EXEMPLES DE SUBSTITUTIONS

S=[°X,.(A,.[°Y,NIL])].[°Z,.(°Y,NIL)] APPLIQUE AU TERME
T= +(°X,+(°Y,°Z)) DONNERA
T.S=+(.(A,.(°Y,NIL)),+(°Y,.(°Y,NIL)))

S=[°X,.(A,°Y)] APPLIQUE AU TERME
T=.(°X,.(°X,°Y)) DONNERA
T.S=.(.(A,°Y),.(.(°A,.,°Y),°Y))

SOIENT S1=[°X,.(°Z,A)] ET S2=[°Z,.(°Y,NIL)] ET S=S1.S2
S S'ECRIT SOUS FORME CANONIQUE:
S=[°X,.(.(°Y,NIL),A)].[°Z,.(°Y,NIL)]

SOIENT S1=[°X,.(°Z,A)] ET S2=[°Z,F(°X)]
ALORS S1.S2 N'A PAS DE SENS CAR IL CREE LA SUBSTITUTION [°X,.(F(°X),A)]

I.3 UNIFICATION DE TERMES

L'UNIFICATION DE DEUX TERMES EST L'OPERATION DE BASE DU SYSTEME PROLOG
C'EST ELLE QUI PROVOQUE:
-LES SUBSTITUTIONS
-LES TESTS D'EGALITE D'ARBRES ET DE SOUS-ARBRES
-LES RECONNAISSANCES PAR CAS("PATTERN-MATCHING")
-L'ACCES AUX SOUS TERMES D'UN TERME

UNIFIER DEUX TERMES T ET U, CONSISTE A APPLIQUER UNE SUBSTITUTION MINIMALE(AU SENS DU PRODUIT DE SUBSTITUTIONS) S, QUI RENDE LES DEUX TERMES EGAUX(U.S=T.S).

LORSQUE CETTE SUBSTITUTION EXISTE, NOUS DIRONS QUE LES DEUX TERMES SONT UNIFIABLES.
CETTE SUBSTITUTION EST UNIQUE, MODULO UN CHANGEMENT DE NOM DES VARIABLES. ELLE EST MINIMALE DANS LE SENS SUIVANT:
TOUT AUTRE SUBSTITUTION S1 QUI REND T ET U EGAUX PEUT S'ECRIRE SOUS LA FORME S1=S.S2.

1.EXEMPLES

T=.(A,.(B,C))
U=.(°X,°Y)
SONT UNIFIABLES AVEC
S=[°X,A].[°Y,.(B,C)]
QUI DONNE
T.S=U.S=.(A,.(B,C))

T=PLUS(.(A,B),°Y)
U=PLUS(°X,°X)
SONT UNIFIABLES AVEC
S=[°X,.(A,B)].[°Y,.(A,B)]
QUI DONNE
U.S=T.S= PLUS(.(A,B),.(A,B))

T=PLUS(.(A,B),C)
U=PLUS(°X,°X)
NE SONT PAS UNIFIABLES CAR °X NE PEUT ETRE SUBSTITUE PAR .(A,B) ET C.

T=.(°X,F(°X,°Y))
U=.(°Z,F(A,.(°Z,°U)))
SONT UNIFIABLES AVEC
S=[°X,A].[°Z,A].[°Y,.(A,°U)]
QUI DONNE
T.S=U.S=.(A,F(A,.(A,°U)))

NOUS ALLONS PRECISER EXACTEMENT L'ALGORITHME DE L'UNIFICATION.

EXEMPLE DE SOUS-TERMES DISTINCTS

SOIENT DEUX TERMES T ET U DISTINCTS.
LE COUPLE DE SOUS TERMES QUI DISTINGUE T ET U EST DEFINI COMME SUIT:
-SI T ET U SONT DE LA FORME
T=F(T1,.....,TN)
U=F(U1,.....,UN)
C'EST A DIRE AVEC LE MEME SYMBOLE FONCTIONNEL ET LE MEME NOMBRE D'ARGUMENTS.
I ETANT LE PREMIER INDICE TEL QUE TI#UI.
ALORS LE COUPLE QUI DISTINGUE T ET U EST LE MEME QUE POUR TI ET UI.

-SINON LE COUPLE EST T U LUI MEME.

EXEMPLES:

T=PLUS(°X,.(°Y,A),°Z)

U=PLUS(°X,.(°Z,°X),C)

A ET °X DISTINGUENT T ET U.

3.ALGORITHME DE L'UNIFICATION

SOIENT DEUX TERMES T ET U. UNIFIER CES DEUX TERMES CONSISTE A APPLIQUER LA SUITE D'OPERATIONS SUIVANTES:

1)-SI T=U ALLER A 2)
-SINON
SOIENT T1 ET U1 LES DEUX SOUS-TERMES DE T ET U QUI LES DISTINGUENT.
-SI T1 OU U1 EST UNE VARIABLE. PAR EXEMPLE T1=°X. EFFECTUER LA SUBTITUTION S=[°X,U1]. C'EST A DIRE:
T.S-->T
U.S-->U
ALLER EN 1)
-SINON ALLER A 3)

2) LES DEUX TERMES SONT UNIFIES. FIN.

3) LES DEUX TERMES NE SONT PAS UNIFIABLES. FIN.

EXEMPLE:

T=P(°X,.(°Z,A),°X)
U=P(°Y,°Y,.(°U,A))

POUR UNIFIER CES DEUX TERMES , NOUS ALLONS APPLIQUER LES OPERATIONS SUIVANTES:

-APPLIQUER S1=[°X,°Y)] D'OU
T=P(°Y,.(°Z,A),°Y)
U=P(°Y,°Y,.(°U,A))

-APPLIQUER S2=[°Y,.(°Z,A)] D'OU
T=P(.(°Z,A),.(°Z,A),.(°Z,A))
U=P(° °Z,A),.(°Z,A),.(°U,A))

-APPLIQUER S3=[°U,°Z]
T=U=P(°,(°Z,A),.(°Z,A),.(°Z,A))

LA SUBSTITUTION S APPLIQUEE AU COURS DE L'UNIFICATION EST DONC
S=S1.S2.S3 SOIT, SOUS FORME CANONIQUE,
S=[°X,.(°Z,A)].[°Y,.(°Z,A)].[°U,.(°Z,A)]

REMARQUES:

DANS L'ETAPE 1). SI °X ADMET UNE OCCURENCE DANS U1 ,L'UNIFICATION CREERA UNE STRUCTURE BOUCLEE. QUI N'EST PAS DETECTEE PAR LE SYSTEME. DE FUTURES UNIFICATIONS POURRONT ALORS BOUCLER SANS QUE LE PROGRAMMEUR Y PRENNE GARDE.

D'AUTRE PART,LORS DE L'UNIFICATION DE DEUX VARIABLES LIBRES.°X ET °Y ON PEUT APPLIQUER INDIFFEREMMENT S=[°X,°Y] OU S=[°Y,°X]. LE RESULTAT SERA LE MEME DANS LES DEUX CAS HORMIS UN CHANGEMENT DE °X EN °Y ET INVERSEMENT.

CHAP II: STRUCTURE ET GESTION DU CONTROLE

UN PROGRAMME PROLOG EST CONSTITUE DE SOUS-PROGRAMMES. CHACUN D'EUX ETANT
ETANT UN ENSEMBLE ORDONNE DE CLAUSES.

L'EXECUTION D'UN APPEL A UN SOUS PROGRAMME EST NON DETERMINISTE. I-E
QU'IL PEUT S'EXECUTER DE DIFFERENTES FACONS.
CHAQUE CLAUSE D'UN SOUS-PROGRAMME DEFINIT ALORS UNE DES FACONS
POSSIBLES D'EXECUTER LE SOUS-PROGRAMME.

II.1 CLAUSES

1.SYNTAXE DES CLAUSES

```
<CLAUSE>::= <LITTERAL DE TETE><SUITE D'APPELS>\
            <LITTERAL DE TETE>
<LITTERAL DE TETE>:::=<SIGNE><PREDICAT>
<SIGNE>::= + \ -
<PREDICAT>:::=<SYMBOLE DE PREDICAT> \
            <SYMBOLE DE PREDICAT>(<LISTE DE TERMES>)
<SYMBOLE DE PREDICAT>:::=<UNITE NON RESERVEE>
<SUITE D'APPELS>:::=<LITTERAL D'APPEL> \
                    <LITTERAL D'APPEL><SUITE D'APPELS>
<LITTERAL D'APPEL>::= <SIGNE><VARIABLE> \ <SIGNE><PREDICAT>
```

2.EXEMPLES DE CLAUSES

```
+CONC(.(*X,*Y),*U) -CONC(*Y,*Z)
+EXECUTER(.(*X,*L)) -EXECUTERP(*X) -EXECUTER(*L) -*L
+CHEMIN(*X,*Y,.(*X,*Z)) -ARC(*X,*U) -CHEMIN(*U,*Y,*Z)
+CHEMIN(A,B,.(A,NIL)) -ARC(A,B)
+JOB -EXECUTER(*X)
```

3.SOUS-PROGRAMMES

TOUTES LES CLAUSES DONT LE LITTERAL DE TETE EST FORME
-DU MEME SIGNE
-DU MEME SYMBOLE DE PREDICAT(AVEC LE MEME NOMBRE D'
ARGUMENTS)
CONSTITUENT LES MULTIPLES DEFINITIONS D'UN MEME SOUS-PROGRAMME, DONT
LE NOM EST CONSTITUE DU SIGNE ET DU SYMBOLE DE PREDICAT CONSIDERE.

ON PARLERA ALORS DES SOUS PROGRAMMES
+CONC(*X,*Y) , +CHEMIN(*X,*Y,*Z), ...ETC

4.SEMANTIQUE D'UNE CLAUSE

LE LITTERAL DE TETE D'UNE CLAUSE DEFINIT
-LES CONDITIONS DANS LESQUELLES CETTE CLAUSES POURRA ETRE UTILISEE LORS
D'UN APPEL
-LES SUBSTITUTIONS A EFFECTUER LORS D'UN APPEL A CETTE CLAUSE
LES ACTIONS A REALISER LORS DE L'ACTIVATION DE LA CLAUSE. CE
SONT EUX-MEMES DES APPELS A DES SOUS-PROGRAMMES QUI S'EXECUTERONT DE
FACON NON DETERMINISTE.

ON PEUT DONC DIRE SCHEMATIQUEMENT QU'UN SOUS-PROGRAMME DEFINIT
LES DIFFERENTES FACONS DE RESOUDRE UN PROBLEME.
CHAQUE CLAUSE DEFINIT L'UNE DE CES FACONS(AVEC LES CONDITIONS
SOUS LESQUELLES CELA EST POSSIBLE) A PARTIR DE LA REALISATION
D'AUTRES PROBLEMES (LA RECURSIVITE ETANT PERMISE).

II.2 SEMANTIQUE DES APPELS

L'ACTIVATION D'UN LITTERAL D'APPEL A POUR BUT L'EXECUTION DU
SOUS-PROGRAMME AYANT LE NOM DU SIGNE OPPOSE.
AINSI
-CONC(.(A,NIL),NIL,*X) EST L'APPEL AU SOUS-PROGRAMME
+CONC(*X,*Y,*Z)

UNE ACTIVATION DU LITTERAL CONSISTE
-A EFFECTUER LE CHOIX D'UNE CLAUSE DU SOUS-PROGRAMME
-L'ACTIVATION DE L'APPEL SUR CETTE CLAUSE.

1.LOCALITE DES VARIABLES

SYNTAXIQUEMENT ,LES VARIABLES D'UNE CLAUSE LUI SONT LOCALES.
AU COURS D' UN APPEL A CETTE CLAUSE, CES VARIABLES SONT CREES EN MEMOIRE
AVEC DE NOUVEAUX NOMS, POUR EVITER TOUT CONFLIT AVEC LES VARIABLES
DEJA CREES ET POUR PERMETTRE LA RECURSIVITE.

2.ACTIVATION DE L'APPEL A UNE CLAUSE

L'ACTIVATION DE L'APPEL L A UNE CLAUSE C S'EFFECTUE DE LA FA-
CON SUIVANTE:
-LE LITTERAL D'APPEL ET CELUI DE TETE DE LA CLAUSE DOIVENT
AVOIR DES SIGNES OPPOSES
-LES VARIABLES D LA CLAUSE SONT CREES AVEC DE NOUVEAUX NOMS
-LES DEUX PREDICATS CONSTITUANT L ET L' SONT UNIFIES.

SI L'UNIFICATION EST IMPOSSIBLE , L'ACTIVATION SUR C EST UN E-
CHEC.
SINON
-SI LA LISTE D'APPEL DE C EST VIDE(CLAUSE UNITAIRE) L'EXE-
CUTION DE L'APPEL A CETTE CLAUSE EST TERMINEE.
-SI LA LISTE EST L1 L2 .... LN. CHACUN DE CES APPELS SERA
ACTIVE. L'ACTIVATION DE LJ SE PRODUISANT DES QUE LE LITTERAL
PRECEDENT A FINI D'ETRE EXECUTE(AVEC UN CHOIX DE CLAUSE DONNE)

3.NOTION DE LIENS ENTRE LES VARIABLES

LES VARIABLES CREES LORS D'UN APPEL PEUVENT ETRE SUBSTITUEES ULTERIEU-
REMENT PAR D'AUTRES VARIABLES, OU PAR DES TERMES FONCTIONNELS.
NOUS DIRONS QU'A UN INSTANT DONNE, UNE VARIABLE EST "LIBRE" SI ELLE N'A
PAS ENCORE ETE SUBSTITUEE PAR UN TERME FONCTIONNEL. NOUS DIRONS QU'ELLE
EST LIEE DANS LE CAS CONTRAIRE.
DE PLUS DEUX VARIABLES LIBRES UNIFIEES ENTRE ELLES DEVIENDRONT ALORS
RELIEES ENTRE ELLES(TOUT EN RESTANT LIBRES). TOUTE SUBSTITUTION SUR
L'UNE SERA FAITE AUSSI SUR L'AUTRE.

ON PEUT DONC CONSIDERER QUE CES VARIABLES RELIEES ENTRE ELLES SONT
DES ACCES DIFFERENTS POUR LE MEME EMPLACEMENT MEMOIRE.

L'ACTIVATION D'UN TEL LITTERAL N'EST POSSIBLE QUE SI LA VARIABLE,AU MOMENT DE L'APPEL, EST LIE A UN TERME FONCTIONNEL QUI SERA INTERPRETE COMME UN PREDICAT.

EXEMPLE

```
+EXECSUITE(.(*X.*Y))-*X-EXECSUITE(*Y)
+EXECSUITE(NIL)
```

UN APPEL -EXECSUITE(*L), OU *L EST UNE LISTE PEIGNEE D'APPELS. SERA DEMONTRE LORSQUE TOUS LES APPELS DE LA LISTE *L L'AURONT ETE.

AINSI -EXECSUITE(.(P(*X),.(A.NIL))) VA DONNER LIEU SUCCESSIVEMENT A L'EXECUTION DE -P(*X) ET DE -A .

ON POURRA TROUVER EN III.3.5 UN EXEMPLE D'UTILISATION DE LITTERAUX DE CETTE FORME.

II.3 ETAT DU CONTROLE

LE CONTROLE DE L'EXECUTION EST SIMILAIRE (MODULO LES CHOIX) A L'APPEL DE PROCEDURE RECURSIF CLASSIQUE.
LA GESTION DES CHOIX EST EFFECTUEE PAR "BACKTRACKING", I-E QUE DES QU'UN APPEL A UNE CLAUSE EST UN ECHEC, OU LORSQUE TOUS LES APPELS INITIAUX ONT ETE EXECUTES, LE CONTROLE EST REDONNE AU DERNIER APPEL EFFECTUE, ET DANS L'ETAT OU IL ETAIT, POUR CHANGER DE CHOIX.

1.ETAT DU CONTROLE

A CHAQUE INSTANT, L'ETAT DU CONTROLE EST DEFINI PAR UN ENSEMBLE ARBORESCENT DE POINTS DE CONTROLE.
CHACUN DE CES POINTS REPRESENTE UN LITTERAL D'APPEL, ET UNE LISTE DE CHOIX DE CLAUSES QU'IL N'A PAS ENCORE UTILISE.

CES POINTS SE REPARTISSENT EN TROIS ENSEMBLES:

1:
LES POINTS DEMONTRES(LITTERAUX ACTIVES AVEC UN CLAUSE DONNEE, DONT L'EXECUTION EST TERMINEE POUR CE CHOIX).LA LISTE DES CHOIX ASSOCIEE A CHACUN DE CES POINTS EST FORMEE A PARTIR DES CLAUSES DU SOUS-PROGRAMME CONCERNE.

2:
LES POINTS EN DEMONSTRATION(LITTERAUX ACTIVES AVEC UNE CLAUSE, ET DONT L'EXECUTION AVEC CE CHOIX N'EST PAS TERMINEE).
CES POINTS SONT EUX AUSSI MUNIS D'UNE LISTE DE CHOIX.

3:
LE POINT AUQUEL LE CONTROLE DOIT ETRE DONNE, MUNI OU NON D'UNE LISTE DE CHOIX.

4:
LES POINTS A DEMONTRER(APPELS NON ENCORE ACTIVES).

LA STRUCTURE ARBORESCENTE DE L'ENSEMBLE DE CES POINTS EST DEFINIE DE LA FACON SUIVANTE:

-SI UN APPEL L. A ETE ACTIVE SUR UNE CLAUSE C=L' L1 ....LN. LES LITTERAUX L1....LN SONT LES DESCENDANTS IMMEDIATS DE L.
SI UN APPEL L A ETE ACTIVE SUR UNE CLAUSE UNITAIRE, IL EST CONSIDERE COMME DEMONTRE.
-UN POINT DE CONTROLE EST CONSIDERE COMME DEMONTRE QUAND TOUS SES DESCENDANTS LE SONT

L'ETAT INITIAL EST DEFINI PAR UNE LISTE D'APPEL FIXEE PAR UNE COMMANDE D'EXECUTION(CHAP IV.1.2).

2.EXEMPLE D'UN ETAT DU CONTROLE

LISTE D'APPELS INITIALE: -A -B

SOUS-PROGRAMMES:

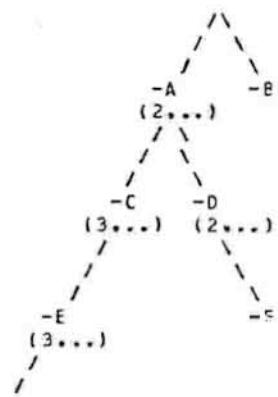
```
+A:
  1. +A-C-D
  2. ...
  ......

+C:
  1....
  2. +C -E
  3....
  .....

+E
  1....
  2. +E
  3....
  .....

+D:
  1. +D -F
  2....
  .....
```

EXEMPLE D'ETAT DU CONTROLE:



-C ET -E SONT DES POINTS DEMONTRES.
-A ET -D SONT EN COURS DE DEMONSTRATION.
-F EST LE POINT AUQUEL SERA DONNE LE CONTROLE.
-B EST A ACTIVER.

CLAUSE UNITAIRE).
LORSQUE -F ET -D AURONT ETE DEMONTRES AVEC TOUS LES CHOIX POSS
SIBLES. LE CONTROLE REVIENDRA PAR "BACKTRACKING" A -D AVEC UNE
LISTE DE CHOIX DIMINUEE D'UNE UNITE. L'APPEL SE FAISANT SUR 2.

II.4 EVOLUTION DE L'ETAT DU CONTROLE

DEUX PROCESSUS, SUIVANT LE CAS, PEUVENT S'APPLIQUER POUR
FAIRE EVOLUER L'ETAT:
 -L'AVANCEMENT
 -LE RETOUR EN ARRIERE.

L'AVANCEMENT(QUI EST L'ACTION INITIALE) CONSISTE A EXECUTER
LES APPELS SUIVANT LE SCHEMA CLASSIQUE, AVEC EN OUTRE DES
CHOIX A EFFECTUER QUAND AUX CLAUSES A UTILISER.
LE RETOUR EN ARRIERE SE PRODUIT DES QU'UNE EXECUTION(AVEC
CERTAINS CHOIX) DE LA LISTE D'APPEL INITIALE EST TERMINEE
OU DES QUE L'ON RENCONTRE UN ECHEC DANS L'AVANCEMENT(FIN DES CHOIX
D'UN POINT DU CONTROLE). LE CONTROLE EST REDONNE ALORS A L'APPEL QUI
PRECEDAIT, POUR FIXER UN NOUVEAU CHOIX ET REPARTIR DANS L'AVANCEMENT.

I.AVANCEMENT DANS L'EXECUTION

1:
 SI AUCUN POINT NE RESTE A ACTIVER(C'EST ALORS QUE LA
LISTE INITIALE EST EXECUTEE AVEC UN ENSEMBLE DE CHOIX),
            REVENIR EN ARRIERE.

2:
SINON, CONSIDERER LE POINT A ACTIVER(DERNIER INTRODUIT
DES LITTERAUX D'APPELS NON ENCORE ACTIVES).

SI SA LISTE DE CHOIX N'EST PAS PRECISEE, LUI ASSIGNER LA LISTE ORDONNEE
(DE HAUT EN BAS)DES CLAUSES DU SOUS-PROGRAMME AYANT COMME NOM, CELUI
AYANT MEME SYMBOLE DE PREDICAT QUE CE LITTERAL,ET LE SIGNE OPPOSE.
OPPOSE.
(CE CAS CORRESPOND A LA PREMIERE ACTIVATION DU POINT)

3:
 SI LA LISTE DES CHOIX EST VIDE ,CONSIDERER CE LITTERAL COMME NON
ACTIVE ET REVENIR EN ARRIERE.

4:
 SINON,CONSIDERER LA PREMIERE CLAUSE C DE CETTE LISTE. LA SUPPRIMER
DES CHOIX DE L, ET EFFECTUER L'APPEL DE L SUR CETTE CLAUSE(CF II.2).

5:
SI L'APPEL EST IMPOSSIBLE, ALLER EN 3 (POUR L'ACTIVATION SUR LA CLAU-
SE SUIVANTE DANS LA LISTE DES CHOIX).

6:
SINON, EFFECTUER LES SUBSTITUTIONS DE L'UNIFICATION SUR L'ENSEMBLE
DES POINTS DU CONTROLE.

 -SI LA LISTE D'APPELS DE C EST L1 L2.....LN, INTRODUIRE CES LIT-
TERAUX COMME POINTS A ACTIVER(L1 ETANT LE PREMIER, LN LE DERNIER). CES
POINTS SONT ALORS CONSIDERES COMME DESCENDANTS DE L.

 -SI LA LISTE DES APPELS EST VIDE(CLAUSE UNITAIRE), CONSIDERER L COMME
DEMONTRE. TOUT ANCETRE DE L AYANT SES DESCENDANTS DEMONTRES EST LUI
AUSSI CONSIDERE COMME DEMONTRE.

7:
 AVANCER

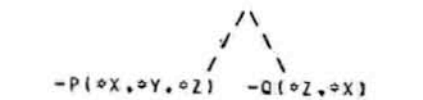
17

EXECUTION DES VARIABLES AU COURS DE L'AVANCEMENT

LORS DE L'AVANCEMENT, A CHACUN DES APPELS, LES VARIABLES
INTRODUITE A CET APPEL OU PRECEDEMMENT PEUVENT ETRE SUBSTITU-
EES PAR D'AUTRES VARIABLES OU PAR DES TERMES FONCTIONNELS.
IL Y A DONC ENTRE LES DIFFERENTS POINTS DU CONTROLE DES LIENS
ENTRE LES VARIABLES.
EN PARTICULIER, CONTRAIREMENT A D'AUTRES LANGAGES USUELS, ON
PEUT AVOIR LES LIENS SUIVANTS:
 -DEUX VARIABLES D'UNE MEME CLAUSE, ORIGINELLEMENT DISTINCTES,
ONT PU ETRE SUBSTITUEES L'UNE PAR L'AUTRE.(I-E DEVENIR RELIEES ENTRE
ELLES), TOUT EN RESTANT LIBRES. TOUTE SUBSTITUTION FAITE SUR L'UNE, SERA
AUSSI FAITE SUR L'AUTRE.
 -TOUTE VARIABLE, UNE FOIS SUBSTITUEE PAR UN TERME FONCTIONNEL
NE PEUT PLUS ETRE SUBSTITUEE PAR UN TERME AU COURS DE L'AVANCEMENT.
 -UNE VARIABLE PEUT ETRE SUBSTITUEE PAR UN TERME FONCTIONNEL
CONTENANT DES VARIABLES LIBRES AU MOMENT DE LA SUBSTITUTION.
PLUS TARD, CES VARIABLES POURRONT A LEUR TOUR ETRE SUBSTITUE
ES.
 -A LA SORTIE DE L'EXECUTION D'UN APPEL(DEMONSTRATION D'UN LITTERAL)
LES VARIABLES FIGURANT DANS L'APPEL, PEUVENT ETRE LIEES A DES TERMES
CONTENANT DES VARIABLES LIBRES CREES DANS LA DEMONSTRATION.

3.EXEMPLE D'AVANCEMENT

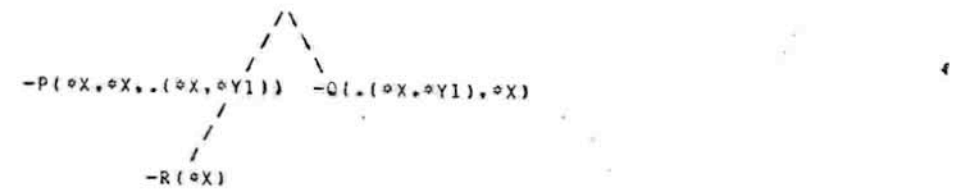
--SOIT LA LISTE D'APPEL SUIVANTE:



ET LES CLAUSES:

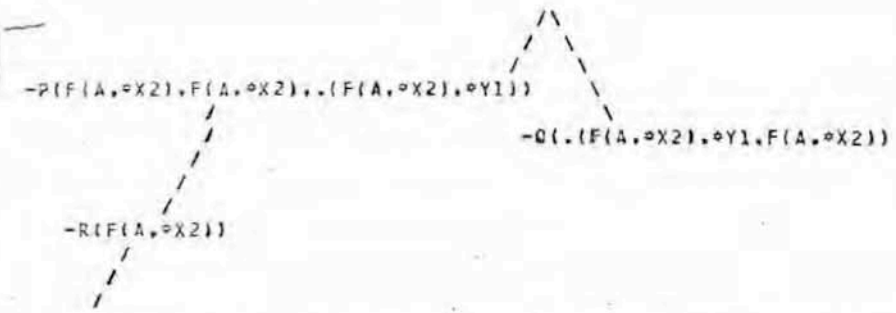
1. +P(*X,*X,.(*X,*Y))-R(*X)
2. +R(F(A,*X))

--APPEL DU LITTERAL -P( ) SUR 1:



OU *X1 ET *Y1 SONT LES VARIABLES NOUVELLES INTRODUITES.
NOTONS QUE *X1 A ETE SUBSTITUE PAR *X, MAIS QUE L'INVERSE AURAIT
PU ETRE FAIT. EN FAIT *X ET *X1 ONT ETE RELIEES ENTRE ELLES.

18



OU °X2 EST INTRODUITE PAR L'APPEL. LES APPELS -P ET -R SONT ALORS DEMONTRES.

--APPEL DE -Q( ).

ETC.....

4.RETOUR EN ARRIERE
==========

LE RETOUR EN ARRIERE SE PRODUIT DES QUE LA LISTE DES CHOIX D'UN POINT DU CONTROLE A ACTIVER EST VIDE.

1:
SI CE LITTERAL EST LE PREMIER DE LA LISTE D'APPELS INITIALE, LE CONTROLE EST REDONNE AU SYSTEME. CECI CORRESPOND A LA FIN D'UN TRAVAIL(CHAP IV.1.2).

2:
SINON. CONSIDERER LE POINT L1 AYANT PRECEDE L AU COURS DE L'AVANCEMENT. REVENIR A L'ETAT QUI PRECEDAIT L'ACTIVATION DE CE POINT L1. LE CONSIDERER COMME POINT A ACTIVER. ET AVANCER.

NOTA-BENE

AU COURS DU RETOUR EN ARRIERE. LES VARIABLES QUI AVAIENT ETE SUBSTITUEES DANS L'AVANCEMENT SONT RESTITUEES DANS LEUR ETAT INITIAL.

5.EXEMPLE DE SCHEMA COMPLET D'EXECUTION
==========

SOIT LA LISTE D'APPEL: -P -Q

LES CLAUSES:

1: +P-A
2: +P-B

1: +Q-C
2: +Q-D

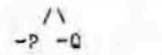
1: +A

1: +B

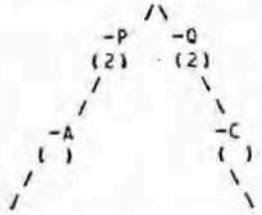
1: +C

1: +D

IL YA DEUS FACONS DE DEMONTRER P ET DEUX POUR -Q. DONC QUATRE POUR LA LISTE -P-Q. PRODUIT DES DEUX ENSEMBLES

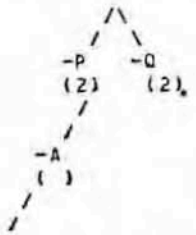


AVANCEMENTS. -P SUR 1.-A SUR 1. -Q SUR 1 ET -C SUR 1:

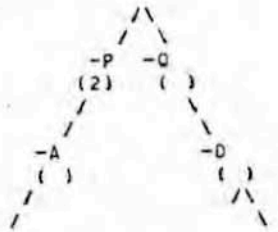


D'OU UNE PREMIERE DEMONSTRATION DE LA LISTE INITIALE.

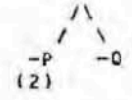
RETOURS SUR -C.-Q:



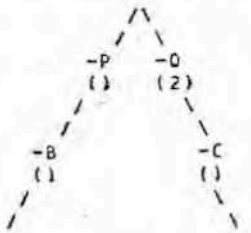
AVANCEMENT DE -Q SUR 2. ET -D SUR 1 (2E DEMONSTRATION)



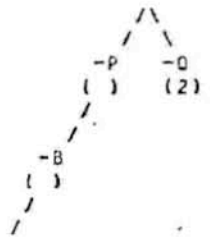
RETOURS SUR -D.-Q.-A.-P:



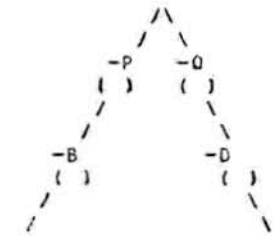
AVANCEMENT DE -P SUR 2. -B SUR 1. -Q SUR 1. -C SUR 1(3E DEMONSTRATION):



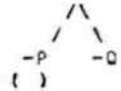
RETOURS SUR -C,-Q:



AVANCEMENT DE -Q SUR 2, ET -C SUR 1(4E DEMONSTRATION):



RETOURS SUR -D,-Q,-B,-P:



FIN DES DEMONSTRATIONS ET RETOUR AU SUPERVISEUR.

°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°
° II.5 EXEMPLE DE SOUS-PROGRAMME °
°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°

SOIT A ECRIRE LE PROGRAMME QUI ETANT LONNES DEUX TERME AYANT UNE STRUCTURE DE LISTE PEIGNEE LES CONCATENE.

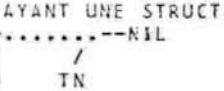
SOIT X1=.(U1,.(U2,....,.(UN,NIL)...)) ET
X2=.(V1,.(V2,....,.(VM,NIL)...))

LE SOUS-PROGRAMME CALCULE LE TERME X3=.(U1,.(U2,.....(V1,..,.(VM,NIL)..)

LE SOUS-PROGRAMME REALISE VA DEFINIR EN FAIT UNE RELATION TERNAIRE SUR L'ENSEMBLE DES TERMES, A L'AIDE D'UN PREDICAT CONC(°X1,°X2,°X3).

LA SEMANTIQUE DE CE PREDICAT SERA ALORS LA SUIVANTE:

+CONC(°X1,°X2,°X3) EST VRAI SI
-°X1 EST UN TERME AYANT UNE STRUCTURE DE LISTE PEIGNEE DE LA FORME:



-°X3 EST LE TERME OBTENU A PARTIR DE °X1 EN REMPLACANT NIL PAR °X2.

1: +CONC(NIL,°X,°X)
2: +CONC(.(°X,°Y),°Z,.(°X,°T)) -CONC(°Y,°Z,°T)

CONSIDERONS ALORS L'APPEL -CONC(°X1,°X2,°X3) .

NOUS ALLONS VOIR COMMENT, ET CONTRAIREMET A D'AUTRES LANGAGES USUELS, LE MEME SOUS-PROGRAMME POURRA ETRE UTILISE AVEC DES VARIABLES D'APPELS DANS UN ETAT QUELCONQUE(LIBRE ,LIEES).
SUIVANT L'ETAT DE CES VARIABLES, L'EXECUTION DE L'APPEL VA DONNER LIEU A UNE EXECUTION DETERMINISTE OU NON, SUIVANT QUE LE PROBLEME A UNE OU PLUSIEURS SOLUTIONS.

1.CAS OU °X1,°X2,°X3 SONT LIES A DES TERMES SANS VARIABLES:
==========

L'APPEL A +CONC(°X1,°X2,°X3) VERIFIE ALORS:
-QUE °X1 EST LIE A UN TERME AYANT UNE STRUCTURE DE LISTE PEIGNEE (DANS LE CAS CONTRAIRE UN RETOUR EN ARRIERE SERA PROVOQUE AVANT LA FIN DE LA DEMONSTRATION DE L'APPEL).
-QUE °X3 EST LIE A UN TERME OBTENU PAR "CONCATENATION" DE °X1 ET °X3.

UNE SEULE DEMONSTRATION DE L'APPEL EST POSSIBLE. DONC APRES AVOIR ETE DEMONTRE, TOUT RETOUR EN ARRIERE SUR CETTE DEMONSTRATION DONNERA LIEU A UN RETOUR EN ARRIERE SUR L'APPEL QUI A PRECEDE -CONC DANS L'AVANCEMENT.

AINSI, SI °X1 EST LIE AU TERME .(A,B)
ET °X2 EST LIE AU TERME .(C,NIL)
ET °X3 EST LIE AU TERME .(A,.(B,NIL))
L'APPEL NE POURRA ETRE DEMONTRE ET SERA SUIVI IMMEDIATEMENT D'UN RETOUR EN ARRIERE.

SI °X1 EST LIE A .(A,NIL)
ET °X2 EST LIE A .(B,.(C,NIL))
ET °X3 EST LIE A .(A,.(B,.(C,NIL)))

LA DEMONSTRATION POURRA ETRE EFFECTUEE,ET LE CONTROLE SERA DONNE ALORS AU PROCHAIN APPEL A ACTIVER.
TOUT RETOUR EN ARRIERE VENANT DE CET APPEL SERA SUIVI D'UN RETOUR SUR LE LITTERAL ACTIVE AVANT -CONC(°X1,°X2,°X3).
APPEL, SERA SUIVI D'UN RETOUR EN ARRIERE SUR LE LITTERAL ACTIVE IMMEDIATEMENT AVANT -CONC(°X1,°X2,°X3).

2.CAS OU°X1 ET °X2 SONT LIEES, ET °X3 LIBRE
==========

L'APPEL VERIFIE QUE °X1 A UNE STRUCTURE DE LISTE PEIGNEE, ET LIE °X3 A UN TERME OBTENU PAR CONCATENATION DE °X1 ET °X3.
LA ENCORE CETTE DEMONSTRATION EST DETERMINISTE, ET TOUT RETOUR EN ARRIERE SUR ELLE SERA SUIVI D'UN RETOUR EN ARRIERRE SUR UN LITTERAL ACTIVE AVANT ELLE.

AINSI, SI °X1 EST LIE A: .(A,NIL)
ET °X2 EST LIE A: .(A,.(F(E),NIL))
ALORS °X3 SERA LIE A: .(A,.(A,.(F(E),NIL)))

23

L'APPEL VA ETRE .LA ENCORE . DEMONTRE AVEC UN SEUL ENSEMBLE DE CHOIX POSSIBLES. EN VERIFIANT
-QUE °X3 EST LIE A UN TERME DE TYPE LISTE PEIGNEF. DONT LA "FIN" EST °X2. DE PLUS °X1 SERA LIE DANS CETTE DEMONSTRATION A UN TERME QUI EST °X3 "OTE DE" °X2.

AINSI. SI °X2 EST LIE A .(A,.(C,NIL))
ET °X3 EST LIE A .(°V,.(F(C),.(A,.(A,.(C,NIL)))))
ALORS °X1 SERA LIE A .(°V,.(F(C),.(A,NIL)))

4.CAS OU °X1 ET °X2 SONT LIBRES. °X3 LIE
==========

LE PROBLEME ADMET ALORS PLUSIEURS SOLUTIONS. I-E COUPLES DE TERMES QUI CONCATENES DONNENT °X3.
LE DEMONSTRATION DE L'APPEL POURRA DONC SE FAIRE DE PLUSIEURS FACONS. CHAQUE DEMONSTRATION SERA SUIVIE D'UN AVANCEMENT A L'APPEL QUI SUIT -CONC(°X1,°X2,°X3). ET UN RETOUR EN ARRIERE VENANT DE CET APPEL DONNERA LIEU A UNE DEUXIEME SOLUTION.AINSI DE SUITE JUSQU'A EPUISEMENT.
LE MEME SOUS-PROGRAMME VA DONC S'EXECUTER DE FACON NON DETERMINISTE.

NOTONS QUE POUR UN POINT DE CONTROLE .LES CHOIX SONT EFFECTUES DANS L'ORDRE DES CLAUSES. ET QU'UN RETOUR EN ARRIERE PROVOQUE LE CHANGEMENT DE CHOIX DU DERNIER APPEL ACTIVE(I-E QUE LE RETOUR EN ARRIERE DONNE LE CONTROLE AU LITTERAUX DANS L'ORDRE INVERSE DE L'AVANCEMENT).

SOIT °X3 LIE A .(A,.(B,.(C,NIL)))

-LA PREMIERE DEMONSTRATION DE L'APPEL VA LIER
°X1 A NIL
°X2 A .(A,.(B,.(C,NIL)))

-UN RETOUR EN ARRIERE VENANT D'UN LITTERAL SUIVANT CET APPEL SERA SUIVI D'UNE DEUXIEME DEMONSTRATION OU
°X1 SERA LIE A .(A,NIL)
°X2 SERA LIE A .(B,.(C,NIL))

LA TROISIEME LIERA
°X1 A .(A,.(B,NIL))
°X2 A .(C,NIL)

LA QUATRIEME LIERA
°X1 A .(A,(B,.(C,.NIL)))
°X2 A NIL

UN RETOUR ULTERIEUR SERA ALORS SUIVI D'UN RETOUR VERS UN LITTERAL ACTIVE ANTERIEUREMENT A -CONC(°X1,°X2,°X3). C'EST A DIRE QUE TOUS LES CHOIX POSSIBLES POUR -CONC(°X1,°X2,°X3) ET LES LITTERAUX LE SUIVANT AURONT ETE EFFECTUES.

CET EXEMPLE MONTRE COMMENT LE NON DETERMINISME DE PROLOG. ET LES LIENS DYNAMIQUES QUE L'ON PEUT CREER SUR LES VARIABLES. PEUVENT ETRE UTILISES AVEC UNE PROGRAMMATION "PROPRE" .OU LE BUT ASSIGNE PAR LE PROGRAMMEUR EST AUSSI PROCHE QUE POSSIBLE DU PROGRAMME LUI-MEME.

24

CHAP III: LITTERAUX EVALUABLES

AFIN DE PERMETTRE UNE PROGRAMMATION PRATICABLE A L'AIDE DES CLAUSES. UN CERTAIN NOMBRE DE MECANISME SONT IMPLANTES AU NIVEAU DE L'INTERPRETEUR ET PERMETTENT AU PROGRAMMEUR DE
-COMMUNIQUER AVEC L'EXTERIEUR(ENTREE-SORTIES)
-CREER ET ACCEDER DYNAMIQUEMENT A LA SYNTAXE DES UNITES
-D'ACCEDER ET DE MODIFIER L'ETAT DU CONTROLE
-DE TESTER LES LIENS ENTRE LES VARIABLES
-DE MODIFIER LES SOUS-PROGRAMMES
-D'UTILISER L'ARITHMETIQUE

III.1 SCHEMA GENERAL DE L'EXECUTION DES LITTERAUX EVALUABLES

CHACUN DES SOUS PROGRAMMES CREES AU NIVEAU DU SYSTEME PEUVENT ETRE APPELES PAR DES LITTERAUX. DITS "EVALUABLES".

1.SYNTAXE
=========

<LITTERAL EVALUABLE>:::=<SIGNE><PREDICAT EVALUABLE>
<PREDICAT EVALUABLE>:::=<ENTREE-SORTIE>\
<ACCES AUX UNITES>\
<ACCES A L'ETAT DU CONTROLE> \
<MODIFICATION DES SOUS-PROGRAMMES> \
<TRAITEMENT DES CARACTERES ET NOMBRES> \
<ACCES A L ETAT DES TERMES>

2.ACTIVATION D'UN LITTERAL EVALUABLE
==========

LORS DE L'ACTIVATION D'UN LITTERAL EVALUABLE. LE SYSTEME AU LIEU DE CONSIDERER UN SOUS-PROGRAMME DEFINIT PAR L'UTILISATEUR. EXECUTE LA SUITE D'ACTION SUIVANTE:

LE PREDICAT CONSTITUANT LE LITTERAL. EST EVALUE PAR LE SYSTEME AVEC TROIS RESULTATS POSSIBLES:
VRAI FAUX ERREUR
DES UNIFICATIONS POUVANT EVENTUELLEMENT AVOIR LIEU AU COURS DE CETTE EVALUATION.

L'EXECUTION DU LITTERAL DE LA FORME -P (RESP. +P). OU P EST UN PREDICAT EVALUABLE.PEUT ETRE CONSIDERE ALORS

-SOIT COMME L'UTILISATION D'UNE CLAUSE UNITAIRE. DANS LE CAS OU L'EVALUATION EST VRAI(RESP. FAUX). LA DEMONSTRATION DE CET APPEL PEUT BIEN SUR CREER DES LIENS ENTRE LES VARIABLES.MAIS IL EST TOUJOURS DETERMINISTE. C'EST A DIRE QUE TOUT RETOUR SUR CET APPEL NE SERA JAMAIS SUIVI D' AUTRES DEMONSTRATIONS .

-SOIT COMME UN APPEL IMPOSSIBLE(ET DONC IMMEDIATEMENT SUIVI D'UN RETOUR) EN ARRIERE) DANS LE CAS DE FAUX(RESP. VRAI).

-SOIT COMME L'APPEL D'UNE CLAUSE DU TYPE
+P -ERREUR(<SYMBOLE DE PREDICAT>)
(RESP. -P -ERREUR(<SYMBOLE DE PREDICAT>)
DANS LE CAS D'UN RESULTAT DE TYPE ERREUR. <SYMBOLE DE PREDICAT> EST ALORS L'UNITE QUI EST LE SYMBOLE DE PREDICAT DE P .

=========

-AJOUT(*X) SERA INTERPRETE EN CAS D'EVALUATION ERREUR. COMME UN APPEL A LA CLAUSE +AJOUT(*X) -ERREUR(AJOUT).

LE PROGRAMMEUR PEUT ALORS GERER LUI MEME LES ERREURS DUES A DE MAUVAISES CONDITIONS D'UTILISATION D'UN PREDICAT EVALUABLE.

L'EVALUATION ERREUR EST CELLE QUI RESULTE D'UNE STRUCTURE INCORRECTE DES ARGUMENTS DU PREDICAT AU MOMENT DE L'APPEL.

4.NOTA-BENE
===========
NOUS PRECISERONS POUR CHAQUE PREDICAT, LA "SYNTAXE" DES ARGUMENTS AU MOMENT DE L'APPEL. TOUTE VARIABLE LIEE ETANT ASSIMILEE AU TERME AUQUEL ELLE EST LIEE AU MOMENT DE L'APPEL.

25

III.2 PREDICATS D'ENTREE-SORTIE

1.SYNTAXE

<ENTREE-SORTIE>::= <CHANGEMENT DE PERIPHERIQUE D'ENTREE>\
<LECTURE DE CAR> \
<LECTURE DE CAR NON BLANC> \
<ECRITURE AUTOMATIQUE DU BUFFER D'ENTREE> \
<ECRITURE DU BUFFER D ENTREE> \
<ECRITURE DE CAR> \
<SAUT DE LIGNE>

LES LECTURES SE FONT SUR UN PERIPHERIQUE D'ENTREE QUI PEUT ETRE
-SOIT L'UNITE LOGIQUE 5 (HABITUELLEMENT LA CONSOLE DE SERVICE OU LE LECTEUR DE CARTES)
-SOIT L'UNITE 2 (FICHIER DISQUE).

LES SORTIES S'EFFECTUENT SUR L'UNITE LOGIQUE 6 .

DES FICHIERS OU DES UNITES PHYSIQUES DOIVENT ETRE ASSIGNES AUX UNITES LOGIQUES PREALABLEMENT A TOUTE EXECUTION.
CECI EST DETAILLE DANS IV.3 .

2.PERIPHERIQUE D'ENTREE

<CHANGEMENT DE PERIPHERIQUE D'ENTREE>::= TTY

LE RESULTAT DE L'EVALUATION EST TOUJOURS VRAI.
L'ACTIVATION A POUR EFFET DE BASCULER L'ASSIGNATION DU PERIPHERIQUE D'ENTREE DE 2 A 5 OU INVERSEMENT.
AU DEPART DE CHAQUE EXECUTION, L'ASSIGNATION EST A L'UNITE 5(CONSOLE).

3.LECTURES

<LECTURE DE CAR>::= LU(<VARIABLE>) \ LU(<TERME CARACTERE>)
<LECTURE DE CAR NON BLANC>::= LUB(<VARIABLE>)
LUB(<TERME CARACTERE>)

RIQUE D'ENTREE.
CELLE DE LUB PROVOQUE LA LECTURE DU PREMIER CARACTERE NON BLANC.

LES LECTURES SE FONT SEQUENTIELLEMENT. AVEC UN BUFFER DE 72 CARACTERES LES CARACTERES ETANT CONSIDERES COMME DES UNITES.

26

LE RESULTAT DE L'EVALUATION EST
-VRAI SI L'UNIFICATION DU CARACTERE LU(CONSIDERE COMME UNITE) ET DE L'ARGUMENT DE LU OU LUB EST POSSIBLE.
-FAUX SINON
-ERREUR SI LA LECTURE EST IMPOSSIBLE(FIN DE FICHIER)

4.ECRITURE DE CARACTERES

<ECRITURE DE CAR>::= ECRIT(<TERME CARACTERE>)
<TERME CARACTERE>::= <SYMBOLE FONCTIONNEL CARACTERE>
<SYMBOLE FONCTIONNEL CARACTERE>::= <TOUT CARACTERE EBDIC>

LES SORTIES S'EFFECTUENT SEQUENTIELLEMENT SUR L'UNITE 6 AVEC UN BUFFER DE 72 CARACTEERES.

LE RESULTAT DE L'EVALUATION DONNE
-VRAI SI L'ARGUMENT DU PREDICAT EST UN TERME DE TYPE CARACTERE
-ERREUR SINON

SOIT L'APPEL -ECRIT(*X) :
-SI *X EST LIBRE IL Y AURA APPEL DE LA CLAUSE +ECRIT(*X(-ERREUR(ECRIT)
-SI *X EST LIEE A TERME FONCTIONNEL A, IL Y AURA IMPRESSION DU CARACTERE A SUR L'UNITE DE SORTIE.PUIS EXECUTION DU LITTERAL QUI SUIT -ECRIT(*X).

5.ECRITURE DU BUFFER D'ENTREE

<ECRITURE AUTOMATIQUE DU BUFFER D'ENTREE>::= BOOLISTE
<ECRITURE DU BUFFER D'ENTREE>::= IMPRIME

SELON QUE L'ETAT D'UN REGISTRE SPECIAL EST A VRAI OU FAUX, CHAQUE FOIS QUE LE BUFFER D'ENTREE EST PLEIN, L'IMPRESSION DE CE BUFFER EST EFFECTUEE OU NON SYSTEMATIQUEMENT.
AU DEPART DE L'EXECUTION LE REGISTRE EST A FAUX(PAS D'IMPRESSIONS).

L'ACTIVATION DE BOOLISTE DONNE TOUJOURS VRAI COMME RESULTAT, ET CHANGE L'ETAT DU REGISTRE.

L'ACTIVATION DE IMPRIME EST TOUJOURS VRAI ET PROVOQUE L'IMPRESSION DU BUFFER.

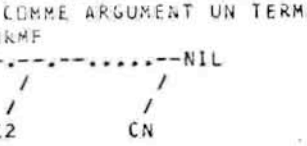
6.SAUT DE LIGNE

<SAUT DE LIGNE>::= LIGNE

CE PREDICAT, TOUJOURS EVALUE A VRAI, PROVOQUE UN SAUT DE LIGNE DANS L'IMPRESSION.

7.EXEMPLE D'UTILISATION DES ENTREES-SORTIES

LE SOUS-PROGRAMME SUIVANT DEFINIT UN PREDICAT ,QUI LORSQU'IL ST ACTIVE, AVEC COMME ARGUMENT UN TERME AYANT UNE STRUCTURE DE LISTE PEIGNEE DE LA FORME



OU C1.C2.....CN SON T DES CARACTERES, PROVOQUE L'IMPRESSION DE

C1 C2 CN !

```
+SORTIE(NIL)-ECRIT(!)-LIGNE
+SORTIE(.(°X,°Y))-ECRIT(°X)-SORTIE(°Y)

+FRREUR(ECRIT)-ECRIT(?)
```

III.3 ACCES AUX UNITES

1.SYNTAXE

<ACCES AUX UNITES>::=UNIV(<VARIABLE>,<TERME FONCT.DECOMPOSE) \
UNIV(<TERME FONCTIONNEL>, <TERME>)

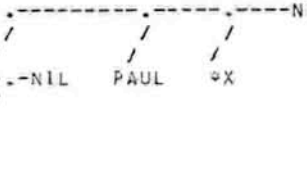
<TERME FONC.DECOMPOSE>::=.(<SYMB.FONC.DEC.>,<LISTE PEIGNEE TERMES>)
<SYMB.FONC.DEC.>::=.(<TERME CARACTERE>,NIL) \
.(<TERME CARACTERE>,<SYMB.FONC.DEC.>)
<LISTE PEIGNEE TERMES>::= NIL \
.(<TERME>,<LISTE PEIGNEE TERMES>)

SEMANTIQUE D'UN TERME DECOMPOSE

TOUTE UNITE ADMET UNE DECOMPOSITION UNIQUE SOUS FORME D'UNE LISTE PEIGNEE DE CARACTERES.
AINSI
PROLOG ADMET COMME DECOMPOSITION .(P,.(R,.(O..(L,.(O,.(G,NIL))))))

DE MEME, TOUT TERME FONCTIONNEL ADMET UNE DECOMPOSITION UNIQUE, OU LE SYMBOLE FONCTIONNEL DECOMPOSE ET LES ARGUMENTS SONT LISTÉS EN UN PEIGNE.

AINSI
PR(PAUL,°X) ADMET COMME DECOMPOSITION LE TERME
.(.(P,.(R,NIL)), .(PAUL,.(°X, NIL)))
SOIT L'ARBRE:



3.SEMANTIQUE DE UNIV

L'ACTIVATION DE UNIV PERMET:
-SOIT DE CREER UN TERME FONCTIONNEL A PARTIR D'UNE UNITE DECOMPOSEE ET D'UNE LISTE D'ARGUMENTS (EN AJOUTANT EVENTUELLEMENT UN NOUVEL ITEM AU DICTIONNAIRE DES SYMBOLES).
-SOIT DE DECOMPOSER UN TERME.

CAS OU LE PREMIER ARGUMENT EST UNE VARIABLE LIBRE

-SI LA SYNTAXE DU DEUXIEME ARGUMENT EST CORRECTE, LE RESULTAT EST VRAI ET LA VARIABLE EST UNIFIEE(I-E LIEE) AU TERME RECONSTRUIT A PARTIR DU DEUXIEME ARGUMENT DE UNIV.

-SINON LE RESULTAT EST ERREUR.

CAS OU LE PREMIER ARGUMENT EST LIE A UN TERME FONCTIONNEL

LE TERME DECOMPOSE DE CE TERME FONCTIONNEL EST CREE PUIS UNIFIE AVEC LE DEUXIEME ARGUMENT DU PREDICAT.
LE RESULTAT DE L'EVALUATION EST VRAI SI L'UNIFICATION EST POSSIBLE, FAUX SINON.

4. PREMIER EXEMPLE D'UTILISATION

LE SOUS-PROGRAMME SUIVANT PERMET D'ECRIRE LA SUITE DE CARACTERES CONSTITUANT L'UNITE D'UN TERME FONCTIONNEL.

```
+ECNOM(°X)-UNIV(°X..(°NOMDEC.°LISTARG))-ECUNITE(°NOMDEC)

+ECUNITE(NIL)
+ECUNITE(.(°CAR.°SUITNOM))-ECRIT(°CAR)-ECUNITE(°SUITNOM)
```

L'APPEL -ECNOM(°X) OU °X EST LIE AU TERME PROLOG(A,B) AU MOMENT DE L'APPEL. VA PROVOQUER L'IMPRESSION DE
PROLOG

LE SOUS-PROGRAMME SUIVANT PERMET,ETANT DONNES DEUX TERMES FONCTIONNELS T1 ET T2 DE LA FORME:

T1=<NOM1> (<P1>,...,<PN>)
T2=<NOM2> (<Q1>,...,<QM>)

DE CREER UN TERME DE LA FORME
T3=<NOM1NOM2>(<P1>,...<PN>,<Q1>,...,<QM>)

C'EST A DIRE DE CONCATENER LES SYMBOLES FONSTIONNELS, ET LES LISTES DE PARAMETRES.

```
+CONCTERME(°X1,°X2,°X3)-UNIV(°X1,.(°NOM1,°LARG1))
   -UNIV(°X2,.(°NOM2,°LARG2))-CONC(°NOM1,°NOM2,°NOM)
   -CONC(°LARG1,°LARG2,°LARG) -UNIV(°X3,.(°NOM,°LARG))

+CONC(°NIL,°X,°X)
+CONC(.(°X,°Y),°Z,.(°X,°T))-CONC(°Y,°Z,°T)
```

L'APPEL DE -CONCTERME(°X1,°X2,°X3) AVEC
-°X1 LIE A FRERE(PAUL,°X)
-°X2 LIE A SOEUR(JEAN,°Y)
-°X3 LIBRE.

VA LIER °X3 AU TERME FRERESOEUR(PAUL,°X,JEAN,°Y)

3.UTILISATION DE UNIV POUR DES ACCES DIRECT AUX CLAUSES

CONSIDERONS LE PROBLEME DE LA GESTION D'UN DICTIONNAIRE DE QUALITES CONCERNANT UN ENSEMBLE D'INDIVIDUS.
UNE PREMIERE FACON DE TRAITER LE PROBLEME EST DE CREER UN ENSEMBLE DE CLAUSES DE LA FORME SUIVANTE:

```
+DIC(JEAN,MASC,HUM)
+DIC(MARIE,FEM,HUM)
+DIC(TABLE,FEM,NEUTRE)
.........
ETC
```

L'ACCES A CE DICTIONNAIRE PEUT SE FAIRE ALORS PAR -DIC(*X,*Y,*Z) ET DONC DE TROIS FACONS POSSIBLES:
- -SOIT PAR LE NOM SI *X EST LIE AU MOMENT DE L'APPEL
- -SOIT PAR LE SEXE SI *Y " " "
- -SOIT PAR LE GENRE SI *Z " " "

L'EXECUTION DE L'APPEL POUVANT DONNER LIEU A UNE OU PLUSIEURS DEMONSTRATIONS SUIVANT LES CAS.
DANS TOUS LES CAS CEPANDANT L'APPEL SERA EFFECTUE SUR CHACUNE DES CLAUSES DU DICTIONNAIRE AVEC SUCCES OU NON POUR CHACUNE D'ENTRE ELLES.

SI CEPANDANT ON DECIDE DE N'ACCEDER AU DICTIONNAIRE QUE A PARTIR DU NOM D'UN INDIVIDU. ET POUR CONNAITRE SES QUALITES. ON PEUT ALORS AVOIR UN ACCES DIRECT DE LA FACON SUIVANTE:

LES INFORMATIONS SERONT CODEES SOUS LA FORME

```
+JEAN(MASC,HUM)
+MARIE(FEM,HUM)
+TABLE(FEM,NEUTRE)
......
```

I-E SOUS FORME DE PLUSIEURS SOUS-PROGRAMMES.

L'APPEL CORRESPONDANT A -DIC(*X,*Y,*Z) SERA REMPLACE PAR
-QUALITE(*X,*Y,*Z)
EN SUPPOSANT QU'AU MOMENT DE L'APPEL *X EST LIE A UN TERME FONCTIONNEL (JEAN,MARIE,....)

LE SOUS-PROGRAMME QUALITE(*X,*Y,*Z) EST LE SUIVANT:

```
+QUALITE(*X,*Y,*Z)-SYMBPRED(*X,.(*Y,.(*Z,NIL)),*PRED) -*PRED

+SYMBPRED(*X,*L,*PRED)-UNIV(*X,*XDEC) -UNIV(*PRED,.(*XDEC,*L))
```

SYMPRED(*X,*L,*PRED) A POUR BUT DE CONSTRUIRE LE TERME *PRED AYANT *X COMME SYMBOLE FONCTIONNEL ET *L COMME LISTE D'ARGUMENTS.

-QUALITE(JEAN,*SEXE,*GENRE) VA ALORS PROVOQUER L'APPEL DE -JEAN(*SEXE,*GENRE), AVEC UN ACCES DIRECT SUR LE SOUS-PROGRAMME +JEAN.

III.4 ACCES A L'ETAT DU CONTROLE

1.SYNTAXE

```
<ACCES A L'ETAT DU CONTROLE>::= <ACCES AUX ANCETRES> \
                                <MODIFICATION DES CHOIX> \
                                <ACCES AUX LITT. NON DEMONTRES>
                                <SAUVEGARDE DE L'ETAT>
```

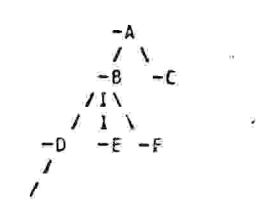
2.NOTION D'ANCETRES

TOUT POINT DU CONTROLE ADMET COMME ANCETRE IMMEDIAT LE LITTERAL QUI A ACTIVE LA CLAUSE OU IL FIGURAIT.
UN ANCETRE D'UN LITTERAL EST L'ANCETRE IMMEDIAT. OU L'ANCETRE IMMEDIAT D'UN ANCETRE.

LES ANCETRES DU LITTERAL A ACTIVER SONT DONC TOUS LES LITTERAUX EN COURS DE DEMONSTRATION.

EXEMPLE:

SOIT L'ARBRE DE CONTROLE SUIVANT:



OU -E EST LE LITTERAL A ACTIVER.

ALORS -E ADMET COMME ANCETRES. DU PLUS RECENT AU PLUS ANCIEN, -B ET -A.

ICI -A A ACTIVE UNE CLAUSE +A-B-C
-B +E-D
-D +D

3.ACCES AUX ANCETRES

```
<ACCES AUX ANCETRES>::= ANCETRE(<LITTERAL SOUS FORME TERME>)
<LITTERAL SOUS FORME TERME>::= <SIGNE>(<TERME>)
```

EXEMPLE: ANCETRE(-(CONC(A.*X.C)))

L'EVALUATION DE CE PREDICAT A POUR EFFET DE RECHERCHER PARMI TOUS LES ANCETRES DU LITTERAL EVALUABLE, LE PREMIER QUI S'UNIFIE AVEC L'ARGUMENT DU PREDICAT ANCETRE.
L'ORDRE DE RECHERCHE S'EFFECTUE DU PLUS RECENT AU PLUS ANCIEN.

LE RESULTAT EST :
-VRAI SI UN TEL ANCETRE EXISTE
-ERREUR SINON.

<MODIFICATION DES CHOIX>::=<SUPPRESSION DES CHOIX DE LA CLAUSE>
<SUPPRESSION DES CHOIX AVEC ANCETRE>

PREMIER CAS

<SUPPRESSION DES CHOIX DE LA CLAUSE>::= /

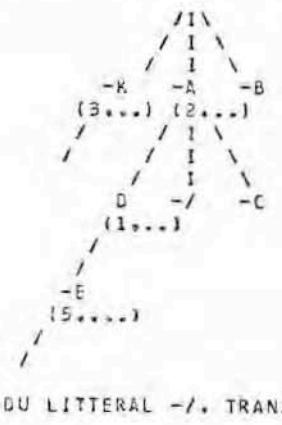
UN PREDICAT DE CETTE FORME, EST TOUJOURS EVALUE A VRAI.

SI LE LITTERAL CONTENANT CE PREDICAT EST SITUE DANS LA CLAUSE

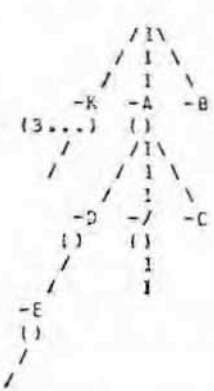
L L1 L2 ... LN <SIGNE>/ LN+1 ..... LM

L'ACTIVATION DU LITTERAL SUPPRIME DANS L'ARBRE DE CONTROLE, POUR TOUS LES POINTS ACCUMULES ENTRE L'APPEL DE LA CLAUSE ET L'ACTIVATION DU LITTERAL EVALUABLE, LES CHOIX RESTANT A EFFECTUER.

AINSI, SI AU MOMENT DE L'APPEL, L'ETAT DU CONTROLE EST LE SUIVANT



L'ACTIVATION DU LITTERAL -/, TRANSFORME L'ETAT EN



LE CONTROLE EST ENSUITE DONNE A -C, ET UN RETOUR EN ARRIERE A PARTIR DE -C, DONNERA LE CONTROLE DIRECTEMENT A -K PUISQUE TOUS LES CHOIX ONT ETE ELIMINES POUR -A ,-D, -E.

DEUXIEME CAS

<SUPPRESSION DES CHOIX AVEC ANCETRE>::= /(<LITTERAL SOUS FORME TERME>)

L'EVALUATION DE CE PREDICAT S'EFFECTUE EN RECHERCHANT LE PLUS RECENT DES ANCETRES QUI S'UNIFIE AVEC L'ARGUMENT DE / .
LE RESULTAT EST ERREUR SI CE LITTERAL ANCETRE EXISTE PAS.
SINON, LE RESULTAT EST VRAI ET LES CHOIX DE TOUS LES POINTS DE CONTROLE ACCCUMULES DEPUIS L'ACTIVATION DE CET ANCETRE(ET Y COMPRIS LUI-MEME) SONT ELIMINES.

UN RETOUR ULTERIEUR EN ARRIERE DONNERA LE CONTROLE AU LITTERAL ACTIVE AVANT CET ANCETRE.

PREMIER EXEMPLE

L'UTILISATION DECRITE ICI DE / PERMET L'EXECUTION D'UN PROCESSUS SIMILAIRE AU "SI...ALORS...SINON"

SOIT PREMIER(°X,°Y) LE PREDICAT DONT LA SEMANTIQUE EST:
-SI °X EST UNE LISTE PEIGNEE DE TERMES ALORS °Y EST LE PREMIER ELEMENT DE LA LISTE QUI VERIFIE P(°X).

+PREMIER(.(°X,°Y),°X)-P(°X)-/
+PREMIER(.(°X,°Y),°Z) -PREMIER(°Y,°Z)

TOUT APPEL DE LA FORME -PREMIER(°X,°Y) A CE SOUS-PROGRAMME, AVEC °X LIE A UN LISTE PEIGNEE ET °Y LIBRE, VA VERIFIER QUE LA LISTE CONTIENT UN ELEMENT VERIFIANT +P(°X) ET UNIFIER °Y CONTRE CET ELEMENT.

EXEMPLE: -PREMIER(.(A..(B..(C,NIL))),°X)

AVEC COMME SOUS-PROGRAMME +P(°X) LE SUIVANT:

+P(C)
+P(B)
+P(D)

CET APPEL VA LIER °X AU TERME B.

DE PLUS TOUT RETOUR EN ARRIERE DONNERA LE CONTROLE AU POINT PRECEDANT CET APPEL PUISQUE LA DEMONSTRATION DE CE LITTERAL N'A ACCUMULE AUCUN CHOIX. UNE SEULE FACON DE DEMONTRER CET APPEL SERA DONC FAITE.

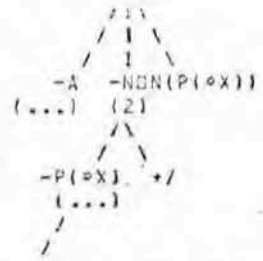
DEUXIEME EXEMPLE

SOIT NON(°X) LE PREDICAT QUI EST IMPOSSIBLE A DEMONTRER SI °LA PROPRIETE °X PEUT L'ETRE ET POSSIBLE SINON.

+NON(°X)-°X+/
+NON(°X)

1:
SOIT L'APPEL -NON(P(°X)) A CE SOUS-PROGRAMME, EN SUPPOSANT QUE -P(°X) PEUT ETRE DEMONTRE D'UNE FACON AU MOINS.
L'ETAT SUIVANT EST CELUI OBTENU APRES ACTIVATION DE -NON(P(°X)) ET APRES DEMONSTRATION DE -P(°X):

33



APRES ACTIVATION IL DE +/ DEVIENDRA:

LE PREDICAT / ETANT EVALUE A VRAI. +/ VA DONC PROVOQUER UN RETOUR
EN ARRIERE IMMEDIAT.
CE RETOUR S'EFFECTUE SUR LE DERNIER APPEL PRECEDANT -NON(P(*X)).

2:
SI -P(*X) NE PEUT ETRE DEMONTRE D'AUCUNE FACON LE RETOUR EN ARRIERE
SUR L'APPEL -NON(P(*X)) VA ACTIVER LA DEUXIEME CLAUSE(TOUJOURS UTILI-
SABLE) PUIS DONNER LIEU A L'AVANCEMENT SANS MODIFIER LE TERME *X.

TROISIEME EXEMPLE

SOIT A CONSTITUER UNE LISTE PEIGNEE DE CARACTERES LUS SUR LE PERIPHERI-
QUE JUSQU'AU CARACTERE ! .
LE PREDICAT LIRE(*X) PERMET DE CONSTITUER UNE TELLE LISTE.

+LIRE(*X)-LU(*C)-LIRELISTE(*C.*X)

+LIRELISTE(!.NIL)-/
+LIRELISTE(*C..(*C.*L))-LU(*C1)-LIRELISTE(*C1.*L)

5.ACCES AUX LITTERAUX NON DEMONTRES
===================================

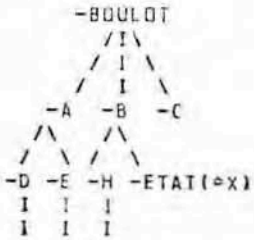
<ACCES AUX LITT. NON DEMONTRES>::= ETAT(<TERME>)

CE PREDICAT PERMET D'ACCEDER A UNE PARTIE DE L'ARBRE DE CONTROLE DU
SYSTEME. CET ETAT EST DEFINI PAR LA SUITE DES ANCETRES DU LITTERAL
EVALUABLE ET DES LITTERAUX NON ENCORE DEMONTRES.
IL EST CONSTITUE PAR UNE LISTE PEIGNEE DONT CHAQUE ELEMENT EST UN NIVEAU
DE L'ARBRE DE CONTROLE(PAR ORDRE DECROISSANT).
CHACUN DES NIVEAUX EST UNE LISTE DONT LES ELEMENTS SONT
LES LITTERAUX A ACTIVER OU EN COURS D'EXECUTION, D'UNE MEME CLAUSE.

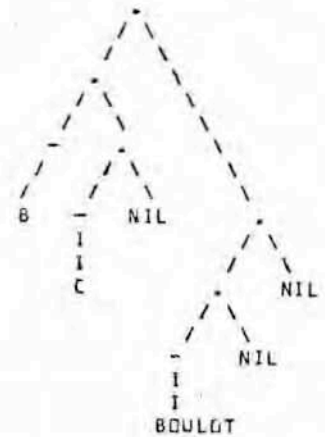
LA LISTE D'APPELS INITIALE EST TOUJOURS -BOULOT .

AINSI.LES LITTERAUX NON DEMONTRES DE L'ETAT SUIVANT:

34



SERONT REPRESENTES PAR L'ARBRE



SOIT, SOUS FORME DE TERME:

.(.(-(B)..(-(C).NIL)) , .(.(-(BOULOT).NIL) .NIL))

L'ETAT AINSI DEFINI PAR UN TERME EST UNIFIE CONTRE L'ARGUMENT DE ETAT.
LE RESULTAT DE L'EVALUATION EST VRAI SI L'UNIFICATION EST POSSIBLE
FAUX SINON.

6.SAUVEGARDE DE L'ETAT
======================

<SAUVEGARDE DE L'ETAT>::= SAUVE

TOUJOURS EVALUE A VRAI. CE PREDICAT A POUR EFFET DE STOPPER L'EXECUTION
ET DE SAUVEGARDER L'ETAT DU SYSTEME SUR UN FICHIER DEFINI A L'EXECUTION
DONT L'ETIQUETTE LOGIQUE EST3.

TOUTE EXECUTION DE PROLOG DOIT ETRE PRECEDEE DE LA DEFINITION DE L'ETI-
QUETTE LOGIQUE 1 , FICHIER SUR LEQUEL SE TROUVE UN ETAT DE DEPART.
CET ETAT PEUT ETRE :
-SOIT UN ETAT STANDARD
-SOIT UN ETAT CREE PAR LE PREDICAT SAUVE DANS UN TRAVAIL ANTERIEUR.

III.5 MODIFICATION DES SOUS-PROGRAMMES

1.SYNTAXE

<MODIFICATION DES SOUS-PROGRAMMES>::=
<AJOUT DE CLAUSE PAR LE HAUT> \
<AJOUT DE CLAUSE PAR LE BAS> \
<SUPPRESSSION DE CLAUSE>

2.AJOUT DE CLAUSE PAR LE HAUT

<AJOUT DE CLAUSE PAR LE HAUT>::= AJOUT(<CLAUSE DECOMPOSEE>)
<CLAUSE DECOMPOSEE>::= .(<LITT. DE TETE> ,
<LISTE DE LITTERAUX-TERMES>)

<LITT. DE TETE >::= <SIGNE> (<TERME FONCTIONNEL>)
<LISTE DE LITTERAUX-TERMES>::= .(<LITTERAL SOUS FORME TERME> .
<LISTE DE LITTERAUX-TERMES>) \
NIL

TOUTE CLAUSE PEUT ETRE DEFINIE BIUNIVOQUEMENT PAR UN TERME AYANT UNE STRUCTURE DE CLAUSE DECOMPOSEE. AINSI

*P(*X)-/-EXEC(*X,*Y)-*Y

SE TRADUIT PAR LE TERME

.(*(P(*X)). .(-(/), .(-(EXEC(*X,*Y)) ,.(-(*Y). NIL))))

L'EVALUATION DE AJOUT, PROVOQUE
-ERREUR SI LE TERME ARGUMENT DE AJOUT N'A PAS UNE STRUCTURE DE CLAUSE DECOMPOSEE AU MOMENT DE L'EVALUATION.
-VRAI SINON. ET L'ADJONCTION DE LA CLAUSE RECOMPOSEE A PARTIR DE L'ARGUMENT DANS LA LISTE DES CLAUSES DU SOUS-PROGRAMME DE MEME NOM..

LA CLAUSE EST AJOUTEE EN TETE DU SOUS-PROGRAMME ET RESTE EN PERMANENCE MEME APRES UN RETOUR EN ARRIERE, JUSQU'A UNE ACTIVATION DU PREDICAT DE SUPPRESSION.

NOTA-BENE1

LA MODIFICATION FAITE PAR UN AJOUT, N'A PAS D'EFFET SUR LES LISTES DES CHOIX DEJA CREES DANS LE CONTROLE. ELLE N'ENTRE EN VIGUEUR QUE POUR LES CREATIONS FUTURES DE LISTES DE CHOIX,LORS DE LA PREMIERE ACTIVATION D UN LITTERAL APPELANT LE SOUS-PROGRAMME CONCERNE.

NOTA-BENE2

LES VARIABLES LIBRES AU MOMENT DE L'ACTIVATION DE AJOUT, ET FIGURANT DANS LE TERME REPRESENTANT LA CLAUSE DECOMPOSEE, SONT RECOPIEES PAR DE NOUVELLES VARIABLES.
DEUX VARIABLES RELIEES ENTRE ELLES AU MOMENT DE L'AJOUT SONT RECOPIEES PAR LA MEME VARIABLE.

SOIT L'APPEL -AJOUT(*X), OU *X AU MOMENT DE L'APPEL EST LIE AU TERME

.(+(VRAI(*X,*X,*Y)),NIL)
OU *X ET *Y SONT DES VARIABLES LIBRES.
LA CLAUSE AJOUTEE EST ALORS
+VRAI(*X,*X,*Y)

EXEMPLE 2:CONSTITUTION D'UN ENSEMBLE DE CLAUSES A PARTIR DE TERMES

LE PREDICAT AJOUDIC SUIVANT PERMET ,ETANT DONNEE UNE LISTE PEIGNEE DE TERMES T1 ,T2, T3,.....TN ,DE CONSTITUER L'ENSEMBLE DES CLAUSES
+DIC(TN)
+DIC(TN-1)
....
+DIC(T1)

LE SOUS-PROGRAMME EST LE SUIVANT:

+AJOUDIC(NIL)
+AJOUDIC(.(*X,*L)) -AJOUT(.(+(DIC(*X)),NIL))-RETOUR
+AJOUDIC(.(*X,*L)) -AJOUDIC(*L)

OU RETOUR EST UN SOUS PROGRAMME INEXISTANT OU MENANT A UN ECHEC.

3.AJOUT DE CLAUSE PAR LE BAS

<AJOUT DE CLAUSE PAR LE BAS>::= AJOUTB(<CLAUSE DECOMPOSEE>)

L'EVALUATION EST LA MEME QUE POUR AJOUT, LA CLAUSE ETANT AJOUTEE EN FIN DU SOUS-PROGRAMME ASSOCIE.
CET AJOUT EST PRIS EN COMPTE POUR LES CHOIX FUTURS, MAIS AUSSI POUR LES CHOIX DEJA CREES.
DE PLUS, SI LE DERNIER APPEL DE AJOUTB CONCERNAIT LE MEME SOUS-PROGRAMME, TOUTE LES CLAUSES DU SOUS-PROGRAMME EXISTANT AU MOMENT DE L'APPEL SONT SUPPRIMEES.

LES LISTES DE CHOIX DEJA CREEES SERONT MODIFIEES PAR UN AJOUT PAR LE BAS.

4.SUPPRESSION DE CLAUSE

<SUPPRESSION DE CLAUSE>::= SUPP(<CLAUSE DECOMPOSEE>)

L'EVALUATION EST LA SUIVANTE:
-ERREUR SI LE TERME ARGUMENT DE SUPP, N'EST PAS UNE CLAUSE DECOMPOSEE
PVRAI SI LA CLAUSE AU SOMMET DU SOUS-PROGRAMME CONCERNE, DECOMPOSEE EN UN TERME, EST UNIFIABLE AVEC L'ARGUMENT DE SUPP
DANS CE CAS LA CLAUSE EST ALORS SUPPRIMEE.
-FAUX SINON.

UNE TELLE SUPPRESSION NE MODIFIE EN RIEN LES LISTES DE CHOIX DEJA CREEES.

SOIT LE SOUS-PROGRAMME SUIVANT:

```
+DIC(A)
+DIC(B)-TRAVAIL(B)
+DIC(C)
```

L'APPEL -SUPP(.(+(DIC(*X)),*Y))
VA SUPPRIMER LA CLAUSE DIC(A). ET LIER *X A TERME A
*Y AU TERME NIL

PUISQUE LA CLAUSE +DIC(A) DECOMPOSEE S'ECRIT
.(+(DIC(A)),NIL)

III.6 TRAITEMENT DES CARACTERES ET DES NOMBRES

1.SYNTAXE

<TRAITEMENT DES CARACTERES ET DES NOMBRES>::=
<TRAITEMENT DES CARACTERES> \
<OPERATION SUR LES NOMBRES> \
<RELATION D'ORDRE>

2.TRAITEMENT DES CARACTERES

<TRAITEMENT DES CARACTERES>::= LETTRE (<TERME CARACTERE>) \
CHIFFRE(<TERME CARACTERE>)

CES PREDICATS SONT EVALUES A
-ERREUR SI L'ARGUMENT N'EST PAS UN TERME DE TYPE CARACTERE
-VRAI SI LE CARACTERE EST UNE LETTRE(RESP. UN CHIFFRE)
-FAUX SI LE CARACTERE N'EST PAS UNE LETTRE(RESP. UN CHIFRE)

3.OPERATION SUR LES NOMBRES

<OPERATION SUR LES NOMBRES>::=<SOMME>\<DIFFERENCE>\<PRODUIT>\
<QUOTIENT>\<RESTE>

<SOMME>::=PLUS(<NOMBRE>,<NOMBRE>,<TERME>)
<DIFFERENCE>::=MOINS(<NOMBRE>,<NOMBRE>,<TERME>)
<PRODUIT>::=MULT(<NOMBRE>,<NOMBRE>,<TERME>)
<QUOTIENT>::=DIV(<NOMBRE>,<NOMBRE>,<TERME>)
<RESTE>::=RESTE(<NOMBRE,NOMBRE>,<TERME>)

CES PREDICATS SONT EVALUES A ERREUR SI LES DEUX PREMIERS ARGUMENTS
N'ONT PAS UNE STRUCTURE D'ENTIERS.

A L'AIDE DES DEUX PREMIERS ARGUMENTS X1 ET X2, UN TROISIEME NOMBRE
X3 EST CREE DE LA FACON SUIVANTE:

-X3=X1+X2 POUR PLUS
-X3=X1-X2 POUR MOINS
-X3=X1.X2 POUR MULT
-X3= QUOTIENT ENTIER DE X1 ET X2 POUR DIV
-X3=RESTE DE LA DIVISION ENTIERE DE X1 PAR X2 POUR RESTE.

LE RESULTAT DE L'EVALUATION EST VRAI SI X3 EST UNIFIABLE AVEC LE
TROISIEME ARGUMENT. FAUX SINON.

EXEMPLE

SOIT LE PREDICAT EVAL(*X,*Y) DONT LA SEMANTIQUE EST:
*Y EST LE RESULTAT DE L'EVALUATION DE L'EXPRESSION *X FORMEE A
PARTIR DES SYMBOLES +(POUR LA SOMME) ET .(POUR LE PRODUIT)

```
+EVAL(+(*X,*Y),*Z)-/-EVAL(*X,*X1)-EVAL(*Y,*Y1)-PLUS(*X1,*Y1,*Z)
+EVAL(.(*X,*Y),*Z)-/-EVAL(*X,*X1)-EVAL(*Y,*Y1)-MULT(*X1,*Y1,*Z)
+EVAL(*X,*X)
```

L'APPEL DE -EVAL(.(+(12,3),8),*X) VA LIER *X AU TERME ENTIER 120 .

4.RELATION D'ORDRE SUR LES NOMBRES ET LES CARACTERES

<RELATION D'ORDRE>::=INF(<NOMBRE>,<NOMBRE>) \
INF(<TERME CARACTERE>,<TERME CARACTERE>)

CE PREDICAT EST EVALUE A
-ERREUR SI LES DEUX ARGUMENTS NE SONT PAS TOUS DEUX DES ENTIERS
OU DES CARACTERES
-VRAI SI LA RELATION EST VERIFIEE(POUR LES CARACTERES, L'ORDRE EST
ALPHABETIQUE).
-FAUX SINON

III.7 ACCES A L'ETAT DES TERMES

1.SYNTAXE

<ACCES A L'ETAT DES TERMES>::=<EGALITE FORMELLE>\
<TEST DE LIBERTE DES VARIABLES> \

2.EGALITE FORMELLE DE TERMES

<EGALITE FORMELLE>::= EGALF(<TERME>,<TERME>)

CE PREDICAT PERMET DE SAVOIR SI ,AU MOMENT DE L'APPEL, LES DEUX
ARGUMENTS SONT DEUX TERMES EGAUX. L'EGALITE ETANT DEFINIS COMME SUIT:

DEUX VARIABLES LIBRES SONT EGALES SSI ELLES SONT RELIEES ENTRE ELLES.
DEUX TERMES FONCTIONNELS SONT EGAUX SSI LEUR SYMBOLE EST LE MEME ET SI
LEUR ARGUMENTS SONT EGAUX.
UNE VARIABLE LIBRE N'EST JAMAIS EGALES A UN TERME FONCTIONNEL.

LE RESULTAT DE L'EVALUATION EST VRAI SSI LES DEUX TERMES SONT EGAUX.

3.TEST DE LIBERTE D'UNE VARIABLE

<TEST DE LIBERTE D'UNE VARIABLE>::= VAR(<TERME>)

CE PREDICAT EST EVALUE A VRAI SI LE TERME, AU MOMENT DE L'APPEL EST
RELIE A UNE VARIABLE LIBRE, FAUX SINON.

CHAP IV UTILISATION DU SUPERVISEUR

ALORS QUE L'INTERPRETEUR PROLOG N'ACCEPTE COMME DONNEES DE BASE QUE DES CLAUSES DECRITES DANS UNE SYNTAXE RUDIMENTAIRE, UN SUPERVISEUR PREND EN COMPTE L'ANALYSE ET LA GESTION DES TRAVAUX DE L'UTILISATEUR.
CE SUPERVISEUR, CONSTITUE PAR DES SOUS-PROGRAMMES DE CLAUSES, COMPREND
-UN ANALYSEUR SYNTAXIQUE PERMETTANT EN ENTREE ET EN SORTIE, UNE SYNTAXE PLUS RICHE QUE L'ECRITURE FONCTIONNELLE.
-UN ENSEMBLE DE SOUS-PROGRAMMES DE LECTURE ECRITURE, PREPROGRAMMES
-UN SYSTEME DE GESTION DES TRAVAUX
-UN SYSTEME DE GRAMMAIRE DE METAMORPHOSES

IV.1 GESTION DES TRAVAUX

LE SUPERVISEUR, QUI A INITIALEMENT LE CONTROLE LORS D'UNE EXECUTION A PARTIR DE L'ETAT STANDARD, A POUR FONCTION DE GERER LES TRAVAUX DE L'UTILISATEUR.
IL LIT SUR LE PERIPHERIQUE D'ENTREE UNE COMMANDE DE TRAVAIL, L'INTERPRETE ET EVENTUELLEMENT DONNE LE CONTROLE A CE TRAVAIL.
UNE FOIS EXECUTE, LE CONTROLE REVIENT AU SUPERVISEUR, CECI AINSI DE SUITE JUSQU'A L'ARRET TOTAL DE L'EXECUTION PAR LA COMMANDE STOP OU L'APPEL DU PREDICAT SAUVE.

1.SYNTAXE

```
<TRAVAIL>:::=<EXECUTION IMMEDIATE> \
             <CLAUSE A AJOUTER> \
             <COMMENTAIRE> \
             <REGLE DE GRAMMAIRE>
```

2.CLAUSE A EXECUTION IMMEDIATE

```
<EXECUTION IMMEDIATE>:::= <SUITE D'APPELS> !
```

LA SUITE DES LITTERAUX DE LA LISTE, CONSTITUE LA LISTE D'APPELS INITIALE DES APPELS A DEMONTRER.
UNE FOIS TOUTES LES DEMONSTRATIONS EFFECTUEES POUR CETTE LISTE, LE CONTROLE REVIENT AU SUPERVISEUR QUI LIT ET INTERPRETE UN NOUVEAU TRAVAIL.
UNE FOIS EXECUTEE, LA CLAUSE DECRIVANT CE TRAVAIL N'EST PAS MEMORISEE.

CHAQUE DEMONSTRATION DE LA LISTE TOTALE DES LITTERAUX CONSTITUANT UN TRAVAIL, EST SUIVI D'UN SAUT DE LIGNE SUR L'UNITE DE SORTIE.

EXEMPLE

```
-LU(*X)-P(*X,*Y)-ECRIT(*Y)!
```

CE TRAVAIL CONSISTE A
-LIRE UN CARACTERE,
-DEMONTRER P(*X,*Y)
-ECRIRE *Y POUR CHAQUE DEMONSTRATION POSSIBLE DE P(*X,*Y)

3.CLAUSE A AJOUTER

```
<CLAUSE A AJOUTER>:::= <CLAUSE> .
```

LE SUPERVISEUR LIT LA CLAUSE, ET L'AJOUTE PAR LE BAS AU SOUS PROGRAMME DE MEME NOM.
LES CLAUSES DU SOUS-PROGRAMME CONCERNE SONT PREALABLEMENT SUPPRIMMEES SI LE DERNIER AJOUT EFFECTUE CONCERNAIT UN AUTRE SOUS-PROGRAMME.

EXEMPLE

VOICI COMME EXEMPLE, LA LECTURE D'UN SOUS-PROGRAMME CONSTITUANT UN DICTIONNAIRE, PUIS L'ECRITURE DES ELEMENTS CONSTITUANT CE DICTIONNAIRE

```
LECTURE:   +P(A).
"          +P(B).
"          +P(C).
"          -P(*X)-ECRIT(*X)!
ECRITURE:  A
"          B
"          C
LECTURE:   ..........
```

4.COMMENTAIRE

```
<COMMENTAIRE>:::= * <SUITE DE CARACTERES AUTRES QUE .>  .
```

UN TEL COMMENTAIRE EST SIMPLEMENT LU SANS AUCUN EFFET.

5. REGLE DE GRAMMAIRE

```
<REGLE DE GRAMMAIRE> ::= <MEMBRE GAUCHE> == <MEMBRE DROIT> .
<MEMBRE GAUCHE> ::= <NON TERMINAL> / <MEMBRE GAUCHE> <TERMINAL>
<MEMBRE DROIT> ::= <VIDE> / <SYMBOLE COMPLEXE> <MEMBRE DROIT> /
                   <CONDITION> <MEMBRE DROIT>
<CONDITION> ::= <LITTERAL D'APPEL>
<SYMBOLE COMPLEXE> ::= <TERMINAL> / <NON TERMINAL>
<TERMINAL> ::= #<PREDICAT> / #<VARIABLE>
<NON TERMINAL> ::= : <PREDICAT> / : NT ( <LISTE DE TERMES> )

RESTRICTION : LE MEMBRE DROIT NE DOIT PAS ETRE DE LA FORME
              ... <CONDITION> <TERMINAL> ...
```

LES REGLES DE GRAMMAIRE SONT DES REGLES DE REECRITURE DE SOUS-CHAINES DE SYMBOLES COMPLEXES EN CHAINES DE SYMBOLES COMPLEXES. CHAQUE SYMBOLE COMPLEXE A UNE STRUCTURE DE TERME. DES CONDITIONS PEUVENT APPARAITRE DANS LE MEMBRE DROIT D'UNE REGLE, ELLES DOIVENT ETRE VERIFIEES POUR L'APPLIQUER. LE SUPERVISEUR LIT CHAQUE REGLE, LA TRANSFORME EN UNE CLAUSE ET L'AJOUTE PAR LE BAS AU RESTE DES CLAUSES DEJA LUES. CETTE TRANSFORMATION SE FAIT AINSI:

-CHAQUE NON-TERMINAL EST TRANSFORME EN UN LITTERAL, CE LITTERAL EST FORME D'UN SIGNE SUIVI D'UN PREDICAT COMMENCANT PAR ":", ET AYANT DEUX ARGUMENTS SUPPLEMENTAIRES. CE SIGNE EST POSITIF POUR LE NON-TERMINAL DU MEMBRE GAUCHE ET NEGATIF POUR LES AUTRES.

INTRODUITE SOUS FORME DE TERMES DANS LES ARGUMENTS SUPPLEMENTAIRES DES LITTERAUX CREES PAR LES NON-TERMINAUX.

-LE SIGNE == EST SUPPRIME.

-CHAQUE CONDITION RESTE TELLE QUELLE.

-LE NON-TERMINAL :NT(X1,X2,....,XN) NE DOIT APPARAITRE QUE DANS LE MEMBRE DROIT. IL EST TRANSFORME EN UNE SUITE DE DEUX LITTERAUX FAISANT INTERVENIR LE PREDICAT EVALUABLE UNIV . CES DEUX LITTERAUX SIMULENT LE COMPORTEMENT DU NON-TERMINAL FICTIF :Y1(X2,....,XN) OU Y1 EST LA SUITE DE CARACTERES CONSTITUANT LA CHAINE DE CARACTERES QUE X1 DOIT REPRESENTER LORSQU'IL EST EVALUE.

LES REGLES DE GRAMMAIRES PERMETTENT DE DEFINIR DES "GRAMMAIRES DE METAMORPHOSES". NOUS RECOMMANDONS AU LECTEUR DE SE REFERER A L'ARTICLE CITE A CE SUJET DANS LA BIBLIOGRAPHIE. LA COMMANDE D'ANALYSE-SYNTHESE SYN (DECRITE EN IV.2.7) PERMET D'UTILISER CES GRAMMAIRES POUR ANALYSER OU SYNTHETISER DES TEXTES.

EXEMPLE DE REGLES DE GRAMMAIRE

```
:HANOI(*N) == :METTRE(*N,A,B,C) -/ .

:METTRE(0,*A,*B,*C) ==.
:METTRE(S(*N),*A,*B,*C) ==
   :METTRE(*N,*A,*C,*B) :VERS(*A,*C) :METTRE(*N,*C,*B,*A).

:VERS(*A,*B) == #DE #*A #VERS #*B #!.
```

```
**********************
* IV.2  COMMANDES *
**********************
```

1.SYNTAXE

```
<COMMANDE>::=<COMMANDE OPERATEUR> \ <COMMANDE D'ARRET> \
             <COMMANDE DE LECTURE> \ <COMMANDE D'ECRITURE> \
             <COMMANDE D'ANALYSE-SYNTHESE>
```

CES COMMANDES SONT DES LITTERAUX D'APPELS A DES SOUS-PROGRAMMES PRE-PROGRAMMES DISPONIBLES A L'UTILISATEUR

2.COMMANDE OPERATEUR

```
<COMMANDE OPERATEUR>::= <DEFINITION D'UN OPERATEUR>
                        <SUPPRESSION DES DEFINITIONS D'OPERATEURS>

<DEFINITION D'UN OPERATEUR>::=-AJOP(<NOM D'OPERATEUR>,<PRIORITE>,
                                   <TYPE>)
<NOM D'OPERATEUR>::=<SYM.FONC.DEC.>
<PRIORITE>::=<NOMBRE>
<TYPE>::=<UNAIRE G-D> \ <UNAIRE D-G> \ <BINAIRE G-D> I <BINAIRE D-G>
         <MIXTE G-D> \ <MIXTE D-G>
<UNAIRE G-D>::="X$"
<UNAIRE D-G>::="$X"
<BINAIRE G-D>::="(X$X)$X"
<BINAIRE D-G>::="X$(X$X)"
<MIXTE G-D>::="(D$X)$X"
<MIXTE D-G>::="X$(A$D)"
<SUPPRESSION DES DEFINITIONS D'OPERATEURS>::= -SOP
```

LES COMMANDES D'OPERATEURS PERMETTENT DE DEFINIR DE FACON REMANENTE DES UNITES COMME OPERATEUR INFIXES, UNAIRES, BINAIRES OU MIXTES. UNE FOIS DEFINIE COMME OPERATEUR PAR LA COMMANDE -AJOP, UNE UNITE PEUT ETRE UTILISEE POUR CONSTRUIRE DES EXPRESSIONS, FORME SYNTAXIQUEMENT EVOLUEE DES TERMES.

CHAQUE OPERATEUR DOIT AVOIR
-UN SENS DE PARENTHESAGE(D-G OU G-D)
-UNE PRIORITE(ENTIER >0)
-LE NOMBRE DE SES ARGUMENTS(UNAIRE, BINAIRE OU MIXTE)

LE SUPERVISEUR PEUT ALORS TRADUIRE UNE EXPRESSION EN UN TERME CANONIQUE (EN LECTURE) OU INVERSEMENT UN TERME CANONIQUE EN UNE EXPRESSION(ECRITURE).
LA FORME CANONIQUE EST CELLE DANS LAQUELLE SONT ENREGISTREE LES INFORMATION EN MEMOIRE.

REMARQUE SUR LA COMMANDE AJOP

UNE FOIS DEFINIE COMME OPERATEUR ,UNE UNITE NE PEUT ETRE EMPLOYEE COMME SYMBOLE FONCTIONNEL CLASSIQUE.

DEUX OPERATEURS DE MEMES PRIORITES, NE PEUVENT AVOIR DES SENS DE PARENTHESAGES DIFFERENTS.

LA DEFINITION D'UN OPERATEUR PAR -AJOP EST REMANENTE, JUSQU'A L'UTILISATION DE LA COMMANDE -SOP QUI SUPPRIMME TOUTES LES DEFINITIONS EXISTANTES.

L'UNITE SPECIFIEE DANS LA DEFINITION DOIT ETRE DONNEE SOUS FORME DECOMPOSEE, C'EST A DIRE COMME LISTE PEIGNEE DE CARACTERES.
ON POURRA POUR CELA UTILISER LES CHAINES(IV.2.3).

3.FORME EVOLUEE DES TERMES:LES EXPRESSIONS

SYNTAXE DES TERMES

```
<TERME>::=<TERME PRIMAIRE> \ <EXPRESSION>
<TERME PRIMAIRE>::= <TERME FONCTIONNEL> \ <VARIABLE> \ <CHAINE> \
                    ( <TERME> )
<TERME FONCTIONNEL>::= <SYMBOLE FONCTIONNEL> \
                       <SYMBOLE FONCTIONNEL>!<LISTE DE TERMES>)
<LISTE DE TERMES>::= <TERME> \ <TERME>.<LISTE DE TERMES>
<SYMBOLE FONCTIONNEL>::= <UNITE NON RESERVEE>
<CHAINE>::= "<SUITE DE CARACTERES AUTRE QUE " >"
<VARIABLE>::= *<UNITE NON RESERVEE>
<EXPRESSION>::= <TERME><OPERATEUR><TERME> \ <TERME><OPERATEUR> \
                <OPERATEUR><TERME>
<OPERATEUR>::= <UNITE NON RESERVEE>
```

SEMANTIQUE DES EXPRESSIONS

UN TERME DECRIT A L'AIDE D'OPERATEURS DOIT RESPECTER LES REGLES CI-DESSOUS. CES REGLES PERMETTENT DE TRADUIRE DE FACON NON AMBIGUE UN TERME SOUS FORME D'EXPRESSION EN UN TERME CANONIQUE.

SOIENT
F UN OPERATEUR DE PRIORITE N
A UNE EXPRESSION DE PRIORITE SUPERIEURE OU EGALE A N, DONT LA TRADUCTION EST A'
B UNE EXPRESSION DE PRIORITE SUPERIEURE A N, DONT LA TRADUCTION EST B'

-SI F EST BINAIRE OU MIXTE DE SENS GD (RESP.DG)
ALORS AFB (RESP. BFA) EST UNE EXPRESSION DE PRIORITE N, TRADUITE PAR F(A',B') (RESP. F(B',A'))

-SI F EST MIXTE ,DE SENS GD (RESP. DG)
ALORS FB (RESP. FA) EST UNE EXPRESSION DE PRIORITE N, TRADUITE PAR F(B') (RESP. F(A'))

-SI F EST UNAIRE DE SENS GD (RESP. DG)
ALORS AF (RESP. FB) EST UNE EXPRESSION DE PRIORITE N, TRADUITE PAR F(A') (RESP. F(B'))

REGLE 2
TOUT TERME PRIMAIRE EST CONSIDERE COMME EXPRESSION DE PRIORITE MAXIMUM.

EXEMPLES DE TERMES ECRITS A L'AIDE D'OPERATEURS

CONSIDERONS LE TRAVAIL SUIVANT A EXECUTER:

-AJOP("SUM",1,"↑X") -AJOP("+",2,"(D↑X)↑X)") -AJOP("-",2,"(D↑X)↑X)")
-AJOP(".",3,"X↑(X↑X)") :

EN S'EXECUTANT, IL VA PROVOQUER LES DEFINITIONS D'OPERATEURS SUIVANTS:

SUM	1	DG	UNAIRE
+	2	GD	MIXTE
-	2	GD	MIXTE
.	3	DG	BINAIRE

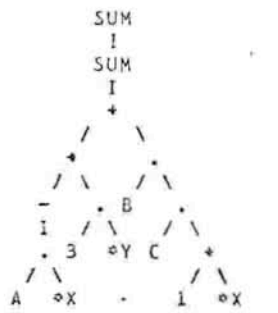
PAR LECTURE(SOIT PAR LA COMMANDE -ENT(°X). SOIT PAR LA LECTURE D'UN TRAVAIL), LE TERME SUIVANT:

SUM SUM -A.°X + 3.°Y + B.C.(1+°X)

SERA TRADUIT EN

SUM(SUM(+(+(-(.(A,°X)),.(3,°Y)),.(B,.(C,+(1,°X))))))

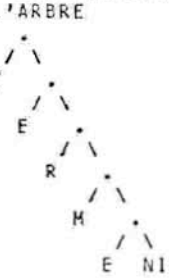
SOIT L'ARBRE



SEMANTIQUE DES CHAINES

TOUTE CHAINE SERA TRADUITE EN UN TERME CONSTITUE A PARTIR DES DEUX UNITES . ET NIL, ET DES CARACTERES DE LA CHAINES CONSIDERES COMME UNITES.
SI LA CHAINE EST "A1 A2.....AN", ELLE SERA TRADUITE PAR
.(A1,.(A2, ...(AN,NIL)....))

EXEMPLE:
"TERME" EST TRADUIT EN
.(T,.(E,.(R,.(M,.(E,NIL)))))
SOIT L'ARBRE



LES CHAINES PEUVENT ETRE UTILISEES POUR LIRE DES UNITES RESERVEES QUE L'ON OBTIENT ALORS SOUS FORME DECOMPOSEE.
EN OUTRE ELLES EVITENT D'AVOIR A PROGRAMMER LES DECOMPOSITIONS DE MOTS EN CARACTERES DANS LES LECTURES.

VOICI UN EXEMPLE DE PREDICAT PERMETTANT DE CONSTITUER UNE LISTE DE CARACTERES A PARTIR DE CARACTERES LUS SUR LE PERIPHERIQUE D'ENTREE, JUSQU'AU CARACTERE VIRGULE:

```
+LIRELISTE(°L) -LU(°C) -SUITE(°C,°X)
+SUITE(°C,NIL) -VIRG(.(°C,NIL)) -/
+SUITE(°C,.(°C,°L)) -LU(°C1) -SUITE(°C1,°L)
+VIRG(",")
```

REMARQUE IMPORTANTE

LA TRADUCTION D'UNE CHAINE EN UNE LISTE DE TERMES CARACTERES EST TOUJOURS EFFECTUEE PAR LE SUPERVISEUR AU MOMENT DE LA LECTURE.
UNE DECOMPOSITION DYNAMIQUE D'UNE UNITE EN UNE LISTE DE CARACTERES NE PEUT ETRE EFFECTUEE QUE PAR LE PREDICAT EVALUABLE UNIV.

4.COMMANDE DE LECTURE

<COMMANDE DE LECTURE>::= <LECTURE D'UN TERME>
<LECTURE D'UN TERME>::= -ENT(<TERME>)

UN APPEL DE CETTE SORTE A POUR EFFET DE LIRE SUR LE PERIPHERIQUE D'ENTREE(EN RESPECTANT LES DEFINITIONS D'OPERATEURS) UN TERME, ET DE L'UNIFIER AVEC L'ARGUMENT DU PREDICAT ENT.
L'APPEL N'EST DEMONTRE QUE SI L'UNIFICATION EST POSSIBLE.
LE TERME LU SUR LE PERIPHERIQUE DOIT ETRE SUIVI DE . OU) POUR MARQUER LA FIN DE LA LECTURE.

LES VARIABLES DU TERME(DECRITES PAR °X,°E,...) SONT RECREES EN DE NOUVELLES VARIABLES. DEUX MEMES NOMS ETANT RECREES EN DES VARIABLES LIBRES LIEES ENTRE ELLES.

```
<COMMANDE D'ECRITURE>::=<ECRITURE D'UN TERME> \
                        <ECRITURE D'UNE CHAINE> \
                        <ECRITURE D'UNE CLAUSE

<ECRITURE D'UN TERME>::= -SORT(<TERME>)
<ECRITURE D'UNE CHAINE>::= -SORM(<LISTE DE CARACTERES>)
<ECRITURE D'UNE CLAUSE>::= -SORC(<CLAUSE DECOMPOSEE>)
<LISTE DE CARACTERES>::= NIL \
                        .(<TERME CARACTERE>.<LISTE DE CARACTERES>)
```

-SORT PERMET D'ECRIRE SUR LE PERIPHERIQUE DE SORTIE . UN TERME QUELCONQUE, EN RESPECTANT LES CONVENTIONS D'OPERATEURS, ET EN REPRESENTANT LES VARIABLES LIBRES DU TERME PAR *X1,*X2,...ETC. DEUX VARIABLES LIBRES MAIS RELIEES ENTRE ELLES ETANT ECRITE A L'AIDE DU MEME NOM.

-SORM NE PEUT ETRE DEMONTRE QUE SI L'ARGUMENT EST UNE LISTE PEIGNEE DE CARACTERES DE LA FORME

```
   .--.--.--.......--NIL
   /  /  /  /       /
  C1 C2 C3 C4      CN
```

IL A POUR EFFET D'IMPRIMER LA LISTE SOUS FORME D'UNE SUITE DE CARACTERES DU TYPE

```
   C1 C2 C3          CN
```

-SORC NE PEUT ETRE DEMONTRE QUE SI L'ARGUMENT A UNE STRUCTURE DE CLAUSE DECOMPOSEE DE LA FORME

```
   .---.---.........---NIL
  /    /   /         /
 L1   L2  L3       LN
```

QU'IL ECRIT SOUS LA FORME

```
+P1 -P2 .........-PN
```

6. COMMANDE D'ARRET

```
<COMMANDE D'ARRET>::= -STOP
```

CETTE COMMANDE A POUR EFFET D'ARRETER L'EXECUTION DU TRAVAIL EN COURS ET DE TERMINER LA SESSION.

7.COMMANDE D'ANALYSE-SYNTHESE

```
<COMMANDE D'ANALYSE-SYNTHESE> ::= SYN ( <CHAINE DE SYMBOLES> .
                                        <CHAINE DE SYMBOLES> )
<CHAINE DE SYMBOLES> ::= . ( <TERME> . <CHAINE DE SYMBOLES> ) / NIL /
                         <VARIABLE>
```

SUIVANT QUE LE PREMIER OU LE DEUXIEME PARAMETRE EST CONNU, CETTE COMMANDE PERMET D'EFFECTUER UNE SYNTHESE OU UNE ANALYSE CONFORME A DES REGLES DE GRAMMAIRE PRECEDEMMENT ENREGISTREES. PLUS EXACTEMENT, POUR TOUT X TOUT Y, SYN(X,Y) EST VRAI SSI LA CHAINE DE SYMBOLES COMPLEXES X PEUT, PAR UNE SUITE DE REECRITURES, SE DERIVER DANS LA CHAINE DE SYMBOLES COMPLEXES Y ET QUE Y N'EST FORME QUE DE TERMINAUX. CONTRAIREMENT AUX REGLES DE GRAMMAIRES LES SYMBOLES COMPLEXES NE SONT NI PRECEDES DE ":" NI DE "1", MAIS DES CONVENTIONS IMPLICITES PERMETTENT DE DISTINGUER LES TERMINAUX DES NON-TERMINAUX:

- DANS LE PREMIER PARAMETRE DE SYN. LE PREMIER ELEMENT DE LA CHAINE DE SYMBOLES COMPLEXES EST TOUJOURS CONSIDERE COMME NON-TERMINAL ALORS QUE TOUS LES AUTRES SONT CONSIDERES COMME TERMINAUX.

- DANS LE DEUXIEME PARAMETRE DE SYN, TOUS LES SYMBOLES COMPLEXES SONT CONSIDERES COMME TERMINAUX.

DE PLUS LA RESTRICTION SUIVANTE DOIT ETRE RESPECTEE:

- LE PREMIER PARAMETRE DE SYN, AU MOMENT DE SON EVALUATION, NE DOIT PAS ETRE UNE VARIABLE ET SON PREMIER ELEMENT NON PLUS.

EXEMPLE

CONSIDERONS LES REGLES DONNEES COMME EXEMPLE EN IV.1.5
SI L'ON NOTE LE POINT SOUS FORME INFIXEE AVEC ASSOCIATION DE DROITE A GAUCHE, L'EVALUATION DE:

```
-SYN(HANOI(S(S(O))).NIL.*T)
```

PROVOQUERA L'UNIFICATION DE *T CONTRE

```
DE.A.VERS.C.!.DE.A.VERS.B.!.DE.C.VERS.A.!.NIL
```

ET INVERSEMENT L'EVALUATION DE

```
-SYN(HANOI(*N).NIL.DE.A.VERS.C.!.DE.A.VERS.B.!.DE.C.VERS.B.!.NIL)
```

PROVOQUERA L'UNIFICATION DE *N CONTRE

```
S(S(O))
```

47

IV.3 ASSIGNATION DES UNITES LOGIQUES

PREALABLEMENT A TOUTE SESSION, ET AVANT L'APPEL DE L'INTERPRETEUR PROLOG, DES FICHIERS OU UNITES PHYSIQUES DOIVENT ETRE ASSIGNEES AUX UNITES LOGIQUES 1,2,3,5 ET 6.

VOICI LA LISTE DES TYPES D'ECHANGES ET LES FONCTIONS DE CHACUNE DE CES UNITES.

UNITE 1
=======

ACCES: LECTURE SEQUENTIELL BINAIRE(DISQUE)

CE FICHIER DOIT CONTENIR L'ETAT DE CONTROLE DE DEPARTQUI PEUT ETRE
 -SOIT UN ETAT STANDARD
 -SOIT UN ETAT CREE PAR LE PREDICAT SAUVE DANS UNE SESSION ANTERIEURE.

L'ETAT DE DEPART COMPREND ,OLTRE L'ENSEMBLE DES SOUS-PROGRAMMES CREES PAR LA LECTURE DE CLAUSE OU PAR LES PREDICATS AJOUT ET AJOUTB, L'ARBRE DE CONTROLE ,LE POINT A ACTIVER, ET LA LISTE DE CHOIX DE CHACUN DES POINTS.

UNITE 2
=======

ACCES: LECTURE SEQUENTIELLE

CETTE UNITE EST UNE DES DEUX UTILISEE COMME PERIPHERIQUE D'ENTREE, COMMUTABLE PAR LE PREDICAT TTY. C'EST HABITUELLEMENT UN FICHIER DISQUE OU UN LECTEUR DE CARTE ,BANDE OU RUBAN.

UNITE 3
=======

ACCES: ECRITURE SEQUENTIELLE BINAIRE(DISQUE)

C'EST LE FICHIER SUR LEQUEL SERA SAUVE L'ETAT DU CONTROLE DE L'EXECUTION LORS D'UNE ACTIVATION DU PREDICAT SAUVE.

UNITE 5
=======

ACCES: LECTURE SEQUENTIELLE(CONSOLE).

C'EST LA DEUXIEME UNITE UTILISEE COMME PERIPHERIQUE D'ENTREE.C'EST A ELLE QU'EST AFFECTE LE PERIPHERIQUE AU DEPART DE TOUTE SESSION.

UNITE 6
=======

ACCES: ECRITURE SEQUENTIELLE

C'EST L'UNITE UTILISEE DANS LES SORTIES PAR ECRIT, SORT....ETC.

48

IV.4 EXEMPLE COMPLET D'UNE SESSION

VOICI L'EXEMPLE COMPLET DE SESSIONS D'UTILISATION DE PROLOG.

SESSION 1
=========

AVANT LA SESSION, LES ASSIGNATIONS SUIVANTES SONT EFFECTUEES:
A L'UNITE 1: LE FICHIER BINAIRE DISQUE "'ETAT STANDARD"
..... 3: FICHIER "ETATSAUVE1" (BINAIRE SUR DISQUE)
..... 5: CONSOLE
.....6: CONSOLE

EXECUTION DE LA SESSION:

UNITE UTILISEE	TEXTE
5:	+CONC(NIL,*X,*X) -/ .
5:	+CONC(.(*X,*Y),*Z,.(*X,*T))-CONC(*Y,*Z,*T).
5:	-SAUVE!

FIN DE LA SESSION: LE PROGRAMME *CONC A ETE SAUVE ET TRADUIT SUR LE FICHIER "ETATSAUVE1".

SESSION 2
=========
ASSIGNATIONS PREALABLES:

UNITE 1: "ETATSAUVE1"
..... 2: "PROGRAMME SOURCE" QUI EST UN FICHIER DISQUE CONTENANT LE TEXTE SUIVANT:
 +LISTES("AB","CA").
 -TTY!

..... 5:CONSOLE
..... 6: CONSOLE

EXECUTION DE LA SESSION:

```
5: -TTY!
2: LECTURE DE +LISTES("AB","CA").
2:            -TTY!
5: -AJOP(".",1,"(X↑X)↑X)") !
5: -LISTES(*X,*Y) -SORT(*X,*Y)-LIGNE
   -CONC(*X,*Y,*Z) -SORT(*Z)!
6: (A.B.NIL).C.A.NIL
6: A.B.C.A.NIL
5: -STOP!
```

FIN DE LA SESSION2 .

CHAP V= METHODOLOGIE DE LA PROGRAMMATION EN PROLOG

LES QUELQUES POINTS PRECISES ICI ONT POUR BUT DE MONTRER COMMENT ABORDER LA PROGRAMMATION EN PROLOG, ET DE DONNER QUELQUES TECHNIQUES DE BASES FRUITS DES RECHERCHES AYANT TRAITEES DE L'UTILISATION DU CALCUL DSE PREDICATS COMME LANGAGE DE PROGRAMMATION.
LE LECTEUR TROUVERA EGALEMENT UNE JUSTIFICATION, QUI SE VEUT DIDACTIQUE DE L'UTILISATION DE PROLOG COMME OUTIL DE PROGRAMMATION, OU PLUS EXECTEMENT COMME LANGAGE D'EXPRESSION DE PROBLEMES.

V.1 PROGRAMMES ET ASSERTIONS

LES CHAPITRES PRECEDENTS ONT DONNES DE PROLOG UNE DEFINITION PUREMENT "OPERATIONNELLE", NECESSAIRE A LA COMPREHENSIN DU DEROULEMENT D'UNE EXECUTION, ET DENOTANT UN CHOIX A PRIORI QUAND A LA FACON DE "DEMONTRER LES LITTERAUX.

EN FAIT UN SOUS-PROGRAMME PEUT ETRE CONSIDERE SOUS DEUX ASPECTS:
-SOIT COMME UN ENSEMBLE D'"INSTRUCTUONS" DONT LE DEROULEMENT A ETE DECRIT PRECEDEMMENT(ASPECT OPERATIONNEL)
-SOIT COMME UN ENSEMBLE D'ASSERTIONS QUANTIFIEES UNIVERSELLEMENT.

I.EXEMPLE D'INTERPRETATION D'UN SOUS-PROGRAMME

SOIT:
+CONC(.(°X,°Y),°Z,.(°X,°T)) -CONC(°Y,°Z,°T)

CETTE CLAUSE PEUT ETRE CONSIDEREE COMME L'ASSERTION SUIVANTE:

QUELQUIT °X,°Y,°X, ET °T,
[.(°X,°Y) CONCATENE A °Z EST EGAL A .(°X,°T)] EST VRAI
SI
[°Y CONCATENE A °Z EST EGAL A °T] EST VRAI.

DE MEME:
-CONC(NIL,°X,°X)
PEUT ETRE INTERPRETEE PAR:

QUELQUIT °X,
[NIL CONCATENE A °X EST EGAL A °X] EST VRAI.

ETANT QUANTIFIEES UNIVERSELLEMENT, CES DEUX CLAUSES SUPPOSENT IMPLICITEMENT L'ASSERTION D'UNE INFINITE DE FORMULES OBTENUES EN SUBSTITUANT LES VARIABLES PAR DES TERMES QUELCONQUES.

RECIPROQUEMENT LORSQUE L'ON CHERCHE A RESOUDRE UN PROBLEME ON PEUT SIMPLEMENT ENONCER LE BUT RECHERCHE A L'AIDE D'ENONCES LOGIQUES.ET CONSIDERER CES ASSERTIONS COMME UN PROGRAMME..

EXECUTER CE PROGRAMME CONSISTE ALORS A DEMONTRER, A PARTIR DE CES ASSERTIONS, UN FAIT QUI EN DECOULE(EN GENERAL L'EXISTENCE D'UN ETRE VERIFIANT TELLE PROPRIETE).

CEPENDANT, UN ESEMBLE QUELCONQUE D'ENONCES LOGIQUES, NE POURRA PAS TOUJOURS ETRE "EXECUTE" DE FACON EFFICACE PAR UN DEMONSTRATEUR DONNE. LE PROBLEME DEVRA DONC ETRE FORMALISE EN FONCTION DU "DEMONSTRATEUR" QUI SERA UTILISE'(EN L'OCCURENCE PROLOG) , DE LA TECHNIQUE DE DEMONSTRATION UTILISEE, ET DES LIMITATIONS POSSIBLES PAR RAPPORT AU LANGAGE TOTAL DE LA LOGIQUE DU PREMIER ORDRE.

2.EXEMPLE D'EXPRESSION DE PROBLEME

SOIT A FORMALISER LA DESCRIPTION D'UN GRAPHE DE RELATIONS ENTRE DES VILLES, ET DU CHEMINEMENT POSSIBLE.

DESCRIPTION DU GRAPHE

```
+ROUTE(PARIS,MARSEILLE).
+ROUTE(PARIS,LONDRES).
+ROUTE(LONDRES,ROME).
+ROUTE(MARSEILLE,NANCY).
```

DESCRIPTION DU CHEMINEMENT

```
+CHEMIN(°X,°Y)-ROUTE(°X,°Z)-CHEMIN(°Z,°Y).
+CHEMIN(°X,°Y)-ROUTE(°X,°Y).
```

CE SOUS-PROGRAMME DECRIT LA FERMETURE TRANSITIVE DU GRAPHE "ROUTE" EN UN GRAPHE "CHEMIN".

DESCRIPTION D'UNE QUESTION

```
-CHEMIN(PARIS,MARSEILLE)!
```

CETTE ECRITURE TRADUIT LA QUESTION:"Y-A-T-IL UN CHEMIN DE PARIS A MARSEILLE",.
AUTREMENT DIT CE LITTERAL NE POURRA ETRE DEMONTRE A PARTIR DES ASSERTIONS QUE S'IL EST LOGIQUEMENT DEDUCTIBLE .

DE MEME LA QUESTION:" QUELLES VILLES MENENT A ROME?" POURRA ETRE DECRITE PAR LE LITTERAL D'APPEL SUIVANT

```
-CHEMIN(°X,ROME)!
```

QUI REPRESENTE LE THEOREME SUIVANT A DEMONTRER:

IL EXISTE °X / CHEMIN(°X,ROME)

ON VOIT DONC LA DUALITE ASSERTION-PROGRAMME, QUESTION-APPEL DE SOUS PROGRAMME, ET DEMONSTRATION-EXECUTION D'APPEL.

V.2 LE LANGAGE PROLOG ET LE LANGAGE DU CALCUL DES PREDICATS

1.ASSERTIONS

TOUT TRAVAIL CONSISTANT EN L'ADJONCTION D'UNE CLAUSE DU TYPE

1) +A(*X,*Y)-B(*X,*Y,*Z).....

OU

2) +A(*X,*Y).

PEUT ETRE CONSIDEREE COMME L'ASSERTION. QUANTIFIEE UNIVERSELLEMET

QQSOIT *X,*Y,.. A(*X,*Y,*Z) EST IMPLIQUE PAR B(*X,*Y,*Z) ET .....

OU

QQSOIT *X, *Y,.. A(*X,*Y)

OPERATIONNELLEMENT. CETTE CLAUSE PEUT ETRE CONSIDEREE COMME SUIT:

TYPE 1:

T1 ET T2 ETANT DEUX TERMES QUELCONQUES. UN FACON POSSIBLE DE DEMONTRER QUE A(T1,T2) EST VRAI. EST DE TROUVER T3,... ET DE DEMONTRER QUE B(T1,T2,T3) ET ..... SONT VRAIS.

TYPE 2:

T1 ET T2 ETANT DES TERMES QUELCONQUES. IL EST TOUJOURS POSSIBLES DE DEMONTRER QUE A(T1,T2) EST VRAI.

2.DEMONSTRATIONS

TOUT TRAVAIL DU TYPE

-A(F(*X))-B(*X,*Y).....!

PEUT ETRE CONSIDERE COMME LE THEOREME SUIVANT A DEMONTRER:

IL EXISTE *X,*Y,.. TELS QUE A(F(*X)) ET B(*X,*Y) ET ..... EST VRAI.

OPERATIONNELLEMENT. CETTE QUESTION CONSTITUE UNE SUITE D'APPELS A EXECUTER. DE TOUTES LES FACONS POSSIBLES. EN FOURNISSANT AINSI AUCUNE, UNE OU PLUSIEURS REPONSES.

UNE INTERPRETATION D'UN PROGRAMME DU INTERVIENNENT LES SYMBOLES DE PREDICATS P.Q.R,....., EST UN ENSEMBLE DE RELATIONS P1.Q1.R1... SUR U

L'INTERPRETATION I SATISFAIT LE PROGRAMME SI ELLE SATISFAIT CHAQUE CLAUSE.

ELLE SATISFAIT LA CLAUSE
+P(F(*X),G(*X,*Y)) PAR EXEMPLE.
SI POUR TOUT ELEMENT T1.T2 DE U, P1(F(T1)).G(T1,T2)) EST VRAI.

ELLE SATISFAIT UNE CLAUSE DU TYPE
+P(F(*X),*X)-Q(*X,*Y)-....
SI, POUR TOUT DOUBLET T1 T2 DE U X U, P(T1,T2) EST VRAI SI Q(T1,T2) L'EST.

EXEMPLE:

+LISTE(.(*X,*Y))-ATOME(*X)-LISTE(*Y).
+LISTE(NIL).

+ATOME(A).
+ATOME(B).

PE PROGRAMME EST SATISFAIT PAR L'INTERPRETATION I SUIVANTE:

ELEMENT VERIFIANT LISTE: NIL..(A,NIL)..(B.NIL) .(E,NIL) .(A..(A.NIL))..

ELEMENT VERIFIANT ATOME: A B C D E.....

CETTE INTERPRETATION,SATISFAISANT LE PROGRAMME, EST CEPANDANT TROP "LARGE".

INTERPRETATION MINIMUM

ON MONTRE ALORS QUE SI, POUR CHAQUE PREDICAT P A N-ARGUMENTS, ON CONSIDERE L'INTERSECTION DE TOUS LES SOUS-ENSEMBLES DE N-UPLETS DE U DEFINIS PAR LES INTERPRETATIONS SATISFAISANT LE PROGRAMME, ON OBTIENT ALORS UNE INTERPRETATION QUI SATISFAIT ELLE AUSSI LE PROGRAMME. ON L'APPELLE INTERPRETATION MINIMUM, ET C'EST ELLE QUI CONSTITUE LA SEMANTIQUE DU PROGRAMME.

POUR CHAQUE SYMBOLE DE PREDICAT P, LLA RELATION PMIN DEFINIT SUR L'UNIVERS DE HERBRAND PAR L'INTERPRETATION MINIMUM POUR CE SYMBOLE, EST LA RELATION LA PLUS FAIBLE .AU SENS DE L'IMPLICATION LOGIQUE,SATISFAISANT AUX AXIOMES.

3.SEMANTIQUE D'UN PROGRAMME PROLOG

INDEPENDEMMENT DE LA FACON DONT LE CONTROLE ET L'EXECUTION D'UN PROGRAMME SERONT EFFECTUES, ON PEUT DEFINIR CE QU'IL EST CENSE FAIRE, C'EST A DIRE DEFINIR SA SEMANTIQUE, EN CONSIDERANT UNIQUEMENT SON ECRITURE (CECI EN OMMETTANT CEPENDANT LE ROLE DES PREDICATS EVALUABLES DONT LA DEFINITION EST DONNEES SOUS UN ASPECT PUREMENT OPERATIONNEL).

L'UNIVERS U DANS LEQUEL SERONT INTERPRETES LES CLAUSES D'UN PROGRAMME EST L'ENSEMBLE INFINI DES TERMES FONCTIONNELS QUE L'ON PEUT DEFINIR AVEC LES SYMBOLES FONCTIONNELS.
EX: A,B,C,.....F(A),F(B),...F(F(A)),...,G(A,F(B))....

POUR CHAQUE SYMBOLE DE PREDICAT P A N-ARGUMENTS, UN PROGRAMME EST CENSE DEFINIR UNE RELATION N-AIRE SUR U, DONC UN SOUS-ENSEMBLE P1 DE U X U X ... X U (L'ENSEMBLE DES N-UPLETS VERIFIANT P).

DANS L'EXEMPLE PRECEDENT NOUS AURIONS OBTENU COMME SEMANTIQUE
 -POUR ATOME: A ET B
 -POUR LISTE: NIL .(A.NIL) .(B.NIL) .(A..(A.NIL)) .(A..(B.NIL)) .....

53

SEMANTIQUE DENOTATIONNELLE ET OPERATIONNELLE

LA SEMANTIQUE AINSI DEFINIE EST PUREMENT "DENOTATIONNELLE", PUISQUE POINT N'EST BESOIN DE CONNAITRE LA FACON DONT ON "DEROULE" LE PROGRAMME POUR LA DEFINIR.

LA SEMANTIQUE "OPERATIONNELLE", POUR UN PREDICAT P(°X,°Y,...) EST L'ENSEMBLE DES N-UPLETS T1,T2, TN POUR LESQUELS UN APPEL
 -P(T1,T2,...)
PEUT ETRE DEMONTRE.

IL Y A ALORS IDENTITE(MODULO LES BOUCLES ET L'EMPLOIDES PREDICATS EVALUABLES) ENTRE LES DEUX SEMANTIQUES.
L'ENSEMBLE DES N-UPLETS POUR LESQUELS L'APPEL PEUT ETRE DEMONTRE EST CELUI, MINIMUM, QUI SATISFAIT LE PREDICAT.

4.IMPLICATION SUR LA METHODOLOGIE

CETTE IDENTITE ENTRE LES DEUX SEMANTIQUES PERMET AU PROGRAMMEUR D'EXPLIQUER, DE COMMENTER ET DE JUSTIFIER AISEMENT SON PROBLEME. DE PLUS L'ECRITURE DU PROGRAMME(PASSAGE DE L'EXPOSITION DU PROBLEME A SA DESCRIPTION DANS LE FORMALISME CHOISI) SERA ASSEZ VOISINE DE L'EXPRESSION MEME DU PROBLEME.
EN OUTRE UNE REFLEXION SUR LA FACON DE DECRIRE LE MONDE TRAITE SERA NECESSAIRE, PERMETTANT ULTERIEUREMENT DE MEILLEURES ET DE PLUS CLAIRES EXPLICATIONS QUE DANS UN LANGAGE PUREMENT OPERATIONNEL.

54

APPENDICE I

REFERENCES ET DEFINITIONS DES UNITES SYNTAXIQUES

UNITE	DEFINITION
<ACCES A L'ETAT DES TERMES>	III.7.1
<ACCES A L'ETAT DU CONTROLE>	III.4.1
<ACCES AUX ANCETRES>	III.4.3
<ACCES AUX LITT.NON DEMONTRES>	III.4.1
<ACCES AUX UNITES>	III.3.1
<AJOUT DE CLAUSE PAR LE BAS>	III.5.3
<AJOUT DE CLAUSE PAR LE HAUT>	III.5.2
<A-NUM>	I.1.1
<BINAIRE D-G>	IV.2.2
<BINAIRE G-D>	IV.2.2
<CHANGEMENTDE PERIPHERIQUE D ENTREE>	III.2.2
<CHAINE>	IV.2.3
<CHAINE DE SYMBOLES>	IV.2.7
<CHIFFRE>	I.1.1
<CLAUSE>	II.1.1
<CLAUSE A AJOUTER>	IV.1.3
<CLAUSE DECOMPOSEE>	III.5.2
<COMMANDE>	IV.2.1
<COMMANDE DE LECTURE>	IV.2.4
<COMMANDE D'ANALYSE-SYNTHESE>	IV.2.7
<COMMANDE D'ARRET>	IV.2.6
<COMMANDE D'ECRITURE>	IV.2.5
<COMMANDE OPERATEUR>	IV.2.2
<COMMENTAIRE>	IV.1.4
<CONDITION>	IV.1.5
<DEFINITION D'UN OPERATEUR>	IV.2.2
<DIFFERENCE>	III.6.3
<ECRITURE AUTOMATIQUE DU BUFFER D'ENTREE>	III.2.5
<ECRITURE DE CAR>	III.2.4
<ECRITURE DU BUFFER D'ENTREE>	III.2.5
<ECRITURE D'UNE CHAINE>	IV.2.5
<ECRITURE D'UNE CLAUSE>	IV.2.5
<ECRITURE D'UN TERME>	IV.2.5
<EGALITE FORMELLE>	III.7.2
<ENTREE-SORTIE>	III.2.1
<EXECUTION IMMEDIATE>	IV.1.2
<EXPRESSION>	IV.2.3
<LECTURE DE CAR>	III.2.3
<LECTURE DE CAR NON BLANC>	III.2.3
<LECTURE D'UN TERME>	IV.2.4
<LETTRE>	I.1.1
<LISTE DE LITTERAUX-TERMES>	III.5.2
<LISTE DE TERMES>	IV.2.3
<LISTE DE TER-CAN>	I.1.2
<LISTE PEIGNEE TERMES>	III.3.1
<LITTERAL DE TETE>	II.1.1
<LITTERAL D'APPEL>	II.1.1
<LITTERAL EVALUABLE>	III.1.1
<LITTERAL SOUS FORME TERME>	III.4.3

55

56

APPENDICE II

SYNOPSIS DES PREDICATS EVALUABLES ET DES COMMANDES DU SUPERVISEUR

LISTE DES PREDICATS EVALUABLES

```
TTY
LU(<VARIABLE>)     LU(<TERME CARACTERE>)
LUB(<VARIABLE>)    LUB(<TERME CARACTERE>)
ECRIT(<TERME CARACTERE>)
LIGNE

UNIV(<VARIABLE>,<TERME FONCT.DECOMPOSE>)
UNIV(<TERME FONCTIONNEL>,<TERME>)

ANCETRE(<LITTERAL SOUS FORME TERME>)

/
/(<LITTERAL SOUS FORME TERME>)

ETAT(<TERME>)

AJOUT(<CLAUSE DECOMPOSEE>)
AJOUTB(<CLAUSE DECOMPOSEE>)
SUPP(<CLAUSE DECOMPOSEE>)

LETTRE(<TERME CARACTERE>)
CHIFFRE(<TERME CARACTERE>)

PLUS(<NOMBRE>,<NOMBRE>,<TERME>)
MOINS(<NOMBRE>,<NOMBRE>,<TERME>)
MULT(<NOMBRE>,<NOMBRE>,<TERME>)
DIV(<NOMBRE>,NOMBRE>,<TERME>)
RESTE(<NOMBRE>,NOMBRE>,<TERME>)

INF(<TERME CARACTERE>,<TERMECARACTERE>)
INF(<NOMBRE>,<NOMBRE>)

EGALF(<TERME>,<TERME>)
VAR(<TERME>)
```

LISTE DES COMMANDES

```
-AJOP(<NOM D'OPERATEUR>,<PRIORITE>,<TYPE>)
-SOP

-ENT(<TERME>)

-SORT(<TERME>)
-SORC(<CLAUSE DECOMPOSEE>)
-SORM(<LISTE DE CARACTERES>)

-SYN(<CHAINE DE SYMBOLES>,<CHAINE DE SYMBOLES>)
```

57

APPENDICE III

BIBLIOGRAPHIE

A.COLMERAUER.
"LES GRAMMAIRES DE METAMORPHOSE".
A PARAITRE, MEMO PROVISOIRE, GROUPE DE RECHERCHE EN INTELLIGENCE ARTIFICIELLE.U.E.R MARSEILLE LUMINY, 1975.

A.COLMERAUER, H.KANOUI, R.PASERO, P.ROUSSEL.
"UN SYSTEME DE COMMUNICATION HOMME-MACHINE EN FRANCAIS",
RAPPORT PRELIMINAIRE, GROUPE RECHERCHE EN INTELLIGENCE ARTIFICIELLE, U.E.R MARSEILLE-LUMINY, 1972.

C.HEWITT.
"DESCRIPTION AND THEORETICAL ANALYSIS OF PLANNER".
MIT-AI-258, 1972.

R.KOWALSKI,
"PREDICATE LOGIC AS PROGRAMMING LANGUAGE",
D.C.L., MEMO 70, UNIVERSITY OF EDINBURGH, 1973.

R.KOWALSKI,
"AN IMPROVED THEOREM-PROVING SYSTEM FOR FIRST ORDER LOGIC".
D.C.L. MEMO 65, UNIVERSITY OF EDINBURGH, 1973

R.KOWALSKI, M.VAN EMDEN,
"THE SEMANTIC OF PREDICATE LOGIC AS PROGRAMMING LANGUAGE",
D.C.L. MEMO 73 , UNIVERSITY OF EDINBURGH, 1974.

N.NILSSON,
"PROBLEM SOLVING IN ARTIFICIAL INTELLIGENCE",
MC.GRAW HILL, N-Y, 1971.

J.A.ROBINSON,
"A MACHINE ORIENTED LOGIC BASED ON RESOLUTION PRINCIPLE",
J.A.C.M, VOL 12, PP23-41, 1965.

P.ROUSSEL,
"DEFINITION ET TRAITEMENT DE L'EGALITE FORMELLE EN DEMONSTRATION AUTOMATIQUE".
THESE 3 CYCLE, UNIVERSITE AIX-MARSEILLE, U.E.R LUMINY, 1972.

E.SANDEWAL,
"CONVERSION OF PREDICATE CALCULUS AXIOMS, VIEWED AS NON-DETERMINISTIC PROGRAMS, TO CORRESPONDING DETERMINISTIC PROGRAMS".
3 RD. I.J.C.A.I. STANFORD, PP 230-234, 1975.

D.WARREN,
"A SYSTEM FOR GENERATING PLANS",
D.C.L. MEMO 76, UNIVERSITY OF EDINBURGH, 1974.