

Micro-Prolog et géométrie élémentaire

Alain Connes

▶ **To cite this version:**

Alain Connes. Micro-Prolog et géométrie élémentaire. Bulletin de l'EPI (Enseignement Public et Informatique), 1986, 44, pp.125-137. edutice-00000978

HAL Id: edutice-00000978
https://edutice.hal.science/edutice-00000978

Submitted on 20 Oct 2005

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MICROPROLOG ET GÉOMÉTRIE ÉLÉMENTAIRE

A. CONNES

Dans cet article, je ne propose pas l'étude des subtilités du langage PROLOG ou MicroPROLOG ni la découverte des mécanismes mis en oeuvre ; pour cela, il suffit de consulter les livres, et documents aujourd'hui publiés (voir bibliographie en annexe).

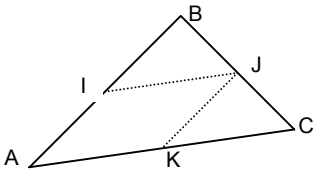
A travers un exemple, je voudrais persuader mes collègues que l'utilisation de ce langage dans une classe (élèves non-spécialisés en informatique permet de renouveler l'approche pédagogique d'un thème, d'un problème classique, en amenant les élèves à formuler règles et données et utilisées, dans la discipline, pour le résoudre.

La discipline considérée est mathématique ! Que les non-mathématiciens ne soient pas effrayés et poursuivent la lecteur de ce qui suit !

PLAN DE L'ARTICLE

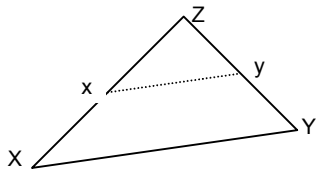
- 1. Le problème
- 2. Syntaxe MicroPROLOG
- 3. Résolution du problème
- 4. Listing du programme PROLOG qui résout le problème posé
- 5. Mise en oeuvre de MicroPROLOG
- 6. Interrogation
- 7. Le départ d'un système-expert
- 8. Un complément la négation
- 9. Une suite possible
- 10. Conclusion provisoire, bibliographie sommaire ANNEXE : iface.doc + iface-mod

1. Le problème



Soit ABC un triangle. Démontrer que si I J K sont les milieux respectifs de AB, BC et CA alors IJKA est un parallélogramme.

2. Syntaxe MicroPROLOG



Considérons l'énoncé suivant bien connu de tous les élèves :

le vecteur IJ est moitié du vecteur A car

- I est milieu du vecteur AB et
- J est milieu du vecteur BC.

Plus généralement, si X Y A x y sont des variables points, nous pouvons énoncer la règle suivante :

la conclusion *"le vecteur (x y) est moitié du vecteur (X Y)"* est vraie si :

l'hypothèse *"x est milieu du vecteur (X Z)"* est vraie et l'hypothèse *"y est milieu du vecteur (Z Y)"* est vraie.

ou encore :

résoudre le problème *"le vecteur (x y) est moitié du vecteur (X Y)"*

c'est résoudre le sous problème *"x est milieu du vecteur (X 7)"*
et le sous problème *"y est milieu du vecteur (Z Y)"*

Si nous utilisons la règle énoncée précédemment ; bien entendu nous pouvons choisir l'application d'autres règles... et PROLOG aussi... à condition de les écrire.

En syntaxe standard MicroPROLOG, remplaçant "est moitié du" par "colineaires-05", une telle règle s'écrit :

```
((colineaires-05 (x y)(X Y)
  (milieu-vecteur x (X Z))
  (milieu-vecteur y (Z Y)))
```

Une règle MicroPROLOG est donc constituée de la partie conclusion (le premier atome, ici "`(colineaires-05 (x y)(X Y))`") et d'une suite d'hypothèses, les atomes suivants. Tous ces atomes, écrits entre parenthèses, sont eux-mêmes délimités par des parenthèses. "`colineaires-05`" est appelé le prédicat. La forme générale d'une règle MicroPROLOG est donc :

```
(conclusion
 hypothese1
 hypothese2
 ...)
```

Il se peut que la suite d'hypothèses soit vide ; la règle prend alors le nom d'assertion : la conclusion est toujours vraie. Ainsi le fait que A soit un point se code : `((point A))`

↑ pas d'hypothèse après la conclusion.

Les éléments d'un programme PROLOG sont des assertions ou des règles, indifféremment appelés des clauses.

3. Résolution du problème

Nous pouvons utiliser droite et parallélisme comme le fait Cuppens dans l'article du bulletin de l'APMEP. Pour ma part, je choisis d'utiliser vecteurs et colinéarité.

Comment montrer le but (I J K A) est en parallélogramme ?

Il suffit de vérifier que les vecteurs (I J) et `(A K)` sont égaux ; en généralisant, si X1 X2 X3 X4 sont des variables points, nous pouvons écrire une première règle :

```
((parallélogramme (X1 X2 X3 X4))
 (vecteurs-egaux (X1 X2) (X4 X3)))
```

Comment montrer `le` but les vecteurs (I J) et (A K) sont égaux ?

Il suffit de vérifier que le vecteur (I J) est la moitié (colineraires-05) du vecteur (A C) et que le vecteur (A K) est la moitié du vecteur (A C) ; en généralisant, X Y Z désignant des vecteurs, nous obtenons une deuxième règle :

```
((vecteurs-egaux X Y)
 (colineaires-05 X Z)
 (colineaires-05 Y Z))
```

Cette règle peut s'interpréter ainsi : les vecteurs X et Y sont égaux si : X est colineaires-05 a un certain vecteur Z
et Y est colineaires-05 a ce même vecteur Z.

Comment montrer le but : le vecteur (I J) est colineaires-05 au vecteur (A C) ?

Il suffit de vérifier que I est milieu du vecteur `(A B)` et que J est milieu du vecteur (B C). En généralisant, nous obtenons la règle :

```
((colineaires-05 (x y)(X Y))
  (milieu-vecteur x (X Z))
  (milieu-vecteur y (Z Y)))
```

Les données du problème sont écrites immédiatement :

```
((point A))  ((point B)) ...
((milieu I (A B))  ((milieu J (B C)) ...
```

Mais attention, nous savons que si I est milieu du vecteur (A B) alors I est aussi le milieu du vecteur (B A). Nous devons l'indiquer ; c'est pourquoi nous proposons deux prédicats distincts "milieu" et "milieu-vecteur" ; nous pouvons dire que I est milieu-vecteur (A B) si I milieu du vecteur (A B) est une assertion connue ou si I milieu du vecteur (B A) est une assertion connue. En généralisant, nous obtenons la règle :

```
((milieu-vecteur X (Y Z))
  (OU
    ((milieu X (Y Z)))
    ((milieu X (Z Y)))))
```

que nous complétons par la règle qui consiste à dire que un point X est toujours le milieu du vecteur (X X) :

```
((milieu-vecteur X (X X))
  (point X))
```

4. Le listing du programme PROLOG qui résout le problème posé

/* les assertions */

```
((point A)) ((point B)) ((point C))
((point I)) ((point J) (point K))
((milieu I (A B))) ((milieu J (C C))) ((milieu K (C A)))
```

/* les règles */

```
((milieu-vecteur X (X X))
  (point X))
((milieu-vecteur X (Y Z))
  (OU ((milieu X (Y Z))) ((milieu X (Z Y))) ))
((vecteurs-egaux X X)
  (vecteur X))
((vecteurs-egaux X Y)
  (colineaires-05 X Z)(coIineaires-05 Y Z))
((colineaires-05 (X1 X2)(Y1 Y2))
  (milieu-vecteur X1 (Y1 Y1 Z)) (milieu-vecteur X2 (Z Y2)))
```

```
((parallélogramme (X1 X2 X3 X4))
 (vecteurs-egaux (X1 X2) (X4 X3)))
```

5. Mise en oeuvre de MicroPROLOG

Nous allons utiliser le module de manipulation de clauses **manipule-mod** contenu dans le fichier **iface.log** (voir annexe). Pour cela, charger le système (CP/M ou MSDOS) puis frapper :

A>prolog ram iface (LOAD à la place de ram si MSDOS)

Le prompt Microprolog est en général le caractère "&" suivi d'un point ".". Il suffit de frapper les clauses, les unes après les autres. Toutefois, nous pouvons écrire les clauses sur plusieurs lignes en validant, par exemple, après chaque atome de la clause ; MicroPROLOG affiche alors en début de ligne, le nombre de parenthèses encore ouvertes, suivi d'un point.

```
&. ((colineai res-05 (x y)(X Y))
1. (milieu-vecteur x (X Z)
1. (milieu-vecteur Y (Z Y »)
&.
```

Attention, si nous listons cette règle, nous découvrons (hélas) que MicroPROLOG a renommé les variables :

```
&. lister colineaires-05
((colineaires-05 (X Y)(Z x))
 (milieu-vecteur X (Z y))
 (milieu-vecteur Y (Y x)))
```

Il faut savoir que MicroPROLOG utilise les variables dont les noms sont, dans l'ordre : `X Y Z x y z X1 Y1 Z1 x1 y1 z1 X2 Y2 Z2...`

6. Interrogation

Nous utilisons le module d'interrogation **iface-mod** contenu dans le fichier **iface.log** et déjà en mémoire. Dans ce qui suit, les commentaires sont placés entre /* et */.

```
/* est-ce-que (l J K A) est un parallélogramme ? */
&.???((parallelogramme (I J K A)))
Oui /* réponse immédiate de MicroPROLOG */
/* comment prouver que (I J K A) est un parallélogramme ?
&.comment(parallelogramme (I J K A))
```

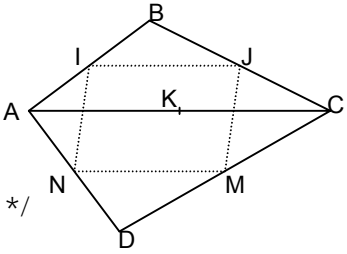
Pour montrer `(parallelogramme (I J K A))` j'utilise

```
        (parallelogramme (I J K A))
        (vecteurs-egaux (I J) (A K))
    Je peux montrer (vecteurs-egaux (I J) (A K))
/* comment prouver que les vecteurs (I J) et (A K) sont égaux? */
&.comment (vecteurs-egaux (I J)(A K))
    Pour- montrer (vecteurs-egaux (I J) (A K)) j'utilise
        (vecteurs-egaux (I J) (A K))
        (colineaires-05 (I J) X)
        (colineaires-05 (A K) X)
    Je peux montrer (colineaires-05 (I J) (A C))
    Je peux montrer (colineaires-05 (A K) (A C))
/* comment prouver que le vecteur (I J) est colinéaire (0,5) */
/* au vecteur (A C) c-a-d IJ = 0,5 * AC ?
        &.comment(colineaires-05 (I J)(A C))
    Pour montrer (colineaires-05 (I J) (A C)) j' utilise
        (colineaires-05 (I J) (A C))
        (milieu-vecteur I (A X))
        (milieu-vecteur J (B C))
    Je peux montrer (milieu-vecteur I (A B))
    Je peux montrer (milieu-vecteur J (B C))
        /* on achève rapidement */
        &.comment(milieu-vecteur I (A B))
    Pour montrer (milieu-vecteur I (A B)) j'utilise
        (milieu-vecteur I (A B))
        (OU ((milieu I (A B))) ((milieu I (B A))))
        (milieu I (A B)) est connu
        &.comment(colineaires-05 (A K)(A C))
    Pour montrer (colineaires-05 (A K) (A C)) j'utilise
        (colineaires-05 (A K) (A C))
        (millieu-vecteur A (A X))
        (milieu-vecteur K (X C))
    Je peux montrer (milieu-vecteur A (A A)) /* hé oui ! */
    Je peux montrer (milieu-vecteur K (A C))
        &.comment(milieu-vecteur A (A A))
    Pour montrer (milieu-vecteur A (A A)) j'utilise
        (milieu-vecteur A (A A))
        (point A)
```

`(point A)` est connu

7. Le départ d'un système expert

Abordons un deuxième exemple : nous ajoutons un point D et deux points M et N milieux respectifs des vecteurs (C D) et (D A). Il s'agit de montrer que IJMN est un parallélogramme.




```
((point D)) ((point M)) ((point N))
((milieu M (C D))) ((milieu N ( *D A)))
/* interrogation */
&.???((parallélogramme (I J M N)))
Oui
/* ça marche!!! Comment est-ce possible ? */
&.comment(parallelogramme (I J M N))
```

Pour montrer (parallélogramme (I J M N))
j'utilise (parallélogramme (I J M N))

`(vecteur-égaux (I J) (N M))`

Je peux montrer (vecteurs-egaux (I J) (N M))

`&.comment(vecteurs-egaux (I J) (N M))`

Pour montrer `(vecteurs-egaux (I J) (N M))` j'utilise

```
(vecteurs-egaux (I J) (N M))
(colineaires-05 (I J) X)
(colineaires-05 (N M) X)
```

Je peux montrer `(colineaires-05 (I J) (A C))`

Je peux montrer `(colineaires-05 (N M) (A C))`

etc.

Nous pouvons distinguer deux types d'informations dans ces exemples :

- les règles effectivement utilisées par les experts mathématicien ; disons rapidement que cet ensemble de règles constitue la **base de connaissance**,
- les faits propres au problème à traiter utilisés et énoncés par l'utilisateur ; cet ensemble de faits constitue la **base de faits**.

PROLOG constitue le **moteur d'inférence** du système en gérant ces deux bases ; enfin les modules du fichier iface.log sont les modules d'interaction avec l'utilisateur : ils permettent de poser les questions, d'une part, pour résoudre les problèmes de l'utilisateur avec l'efficacité

d'un expert (??? quel un résoudre..) et d'autre, part pour acquérir un savoir-faire analogue à celui de l'expert (comment ..) .

Autrement dit, nous venons de construire un petit système-expert !

Bien évidemment, nous pouvons faire évoluer ce système :

- en complétant, en tant qu'expert, la base de connaissance : c'est ce que nous allons faire,
- en modifiant, en tant qu'utilisateur, la base de faits : c'est ce que nous avons déjà fait en ajoutant les points D M et N.

8. Un complément : la négation

Commençons par une parenthèse afin d'expliquer ce que nous entendons par variables locales et variables globales. Considérons pour cela la règle suivante :

```
((vecteurs-egaux X Y)
 (colineaires -05 X Z)
 (colineaires-05 Y Z))
```

Pour cette règle, la variable Z, qui apparaît dans les hypothèses mais pas dans la conclusion, est appelée variable locale alors que les variables X et Y sont appelées variables globales. Ces définitions posées et facilement généralisables, voyons comment exprimer "le vecteur (A B) est non nul" sachant que nous pouvons utiliser la règle :

```
((vecteur-nul (X X))
 (vecteur (X X)
```

avec :

```
((vecteur (X Y))
 (point X) (point Y))
```

Il suffit de nier `((vecteur-nul (A B)))` cela est possible en MicroPROLOG, par exemple sous la forme interrogative suivante :

```
???((NON vecteur-nul (A B)))
```

ou `NON`, mot prédéfini de MicroPROLOG, est écrit impérativement en majuscule ; MicroPROLOG répond `Oui`.

Plus généralement (mais la généralisation que permet Micro-PROLOG est plus importante encore), si `<atome>` est un atome, nous pouvons utiliser la condition d'inversion sous la forme : `(NON <atome>)`

Utilisons cela pour écrire que deux vecteurs sont strictement colinéaires de même sens ; nous dirons qu'ils sont colinéaires plus ; deux vecteurs égaux et non nuls sont colinéaires-plus :

```
((colineaires-plus X Y)
 (vecteurs-egaux X Y)
 (NON vecteur-nul X)
 (NON vecteur-nul Y))
```

deux vecteurs dont l'un est égal à la moitié de l'autre, et non nuls, sont colineaires-plus :

```
((colineaires-plus, X Y)
 (OU
  ((colineaires-05 X Y))
  ((colineaires-05 Y X)))
 (NON vecteur-nul X)
 (NON vecteur-nul Y))
```

Nous pouvons demander quels sont les couples de vecteurs deux à deux colineaires-plus :

```
&.résoudre((colineaires-plus x y))
(colineaires-plus (A B) (A B))
(colineaires-plus (A C) (A C))
(colineaires-plus (A I) (A I)
....
(colineaires-plus (A I) (I B))
(colineaires-plus (A I) (K J))
....
(colineaires-plus (I J) (A K))
*** Interruption *** /* Nous avons frappé ^C*/
```

```
&.comment(colineaire-plus (I J)(A K))
```

Pour montrer `(colineaires-plus (I J) (A K.))` j'utilise

```
(colineaires-plus (I J) (A K))
(vecteurs-egaux (I J) (A K)=
(NON vecteur-nul (I J))
(NON vecteur-nul (A K))
```

Je peux montrer (vecteurs-egaux (I J) (A K))

```
(vecteur-nul (I J)) echoue
(vecteur-nul (A K)) echoue
```

Notez la façon dont MicroPROLOG montre que les inversions :

```
(NON vecteur-nul (I J))
(NON vecteur-nul (A K))
```

sont vraies en montrant l'échec des affirmations correspondantes ; en d'autre terme, avant parcouru toute la base, MicroPROLOG n'a pu vérifier

```
(vecteur-nul (I J))
(vecteur-nul (A K))
```

Autre exemple :

```
&.comment(colineaires-plus (I J)(A C))
```

Pour montrer `(colineaires-plus (I J) (A C))`
j'utilise `(colineaires-plus (I J) (A C))`

```
(OU ((colineaires-05 I J) (A C)))((colineaires-05 (A C) (I J))))
      (NON vecteur-nul (I J))
      (NON vecteur-nul (A C))
```

Je peux montrer `(colineaires-05 (I J) (A C))`

```
(vecteur-nul (I J)) echoue
(vecteur-nul (A C)) echoue
/* enfin ... */
&.comment(colineaires-plus (A B)(B C))
```

Je ne peux montrer `(colineaires-plus (A B) (B C))`

Dans un tel cas, il serait intéressant de savoir pourquoi le système ne peut prouver le but `((colineaires-plus (A B)(B C)))`. C'est l'objet du futur développement de l'interface iface.log !

Nous aurions pu écrire en permutant certains atomes :

```
((colineaires-plus X Y)
 (NON vecteur-nul X)
 (NON vecteur-nul Y)
 (vecteurs-egaux X Y))
((colineaires-plus X Y)
 (NON vecteur-nul X)
 (NON vecteur-nul Y)
 (OU
  ((colineaires-05 X Y))
  ((colineaires-05 Y X))))
```

Sous cette forme, les règles précédentes ne peuvent servir qu'en vérification, pas en calcul.

```
&.???((colineaires-plus (I J)(A C)))
Oui
/* ça marche en vérification mais ..*/
&.resoudre((colineaires-plus x y))
Ok
/* ça ne marche pas en calcul */
```

Énonçons la propriété suivante :

dans une question ou dans une règle, une condition d'inversion (NON <atome>) ne peut âtre utilisée que pour vérifier les valeurs déjà données aux variables globales. Elle ne peut être utilisée pour calculer les valeurs de telles variables globales. Cela signifie que, dans une question ou dans une règle, une condition d'inversion doit être précédée par une condition positive pour chacune de ses variables globales.

Ainsi, reprenant les exemples précédents, nous devons ajouter les conditions positives suivantes (par exemple) :

```
(vecteur X)
(vecteur Y)
```

qui vérifient ou calculent X et Y ; ce qui donne :

```
((colineaires-plus X Y)
 (vecteur X)
 (vecteur Y)
 (NON vecteur-nul X) (NON vecteur-nul Y)
 (vecteurs-egaux X Y) )
((colineaires-plus X Y)
 (vecteur X)
 (vecteur Y)
 (NON vecteur-nul X) (NON vecteur-nul Y)
 (OU
  ((colineaires-05 X Y))
  ((colineaires-05 Y X))))
```

9. Une suite possible (ou un début ?)

Nous pourrions par exemple nous intéresser à la relation de Chasles qui dit que si A, B et C sont 3 points alors la somme des vecteurs (A B) et (B C) est le vecteur (A C) ; plus généralement :

```
((chasles (X Y) (Y Z) (X Z))
```

```
(vecteur (X Y) )(vecteur (Y Z)) (vecteur X Z)
```

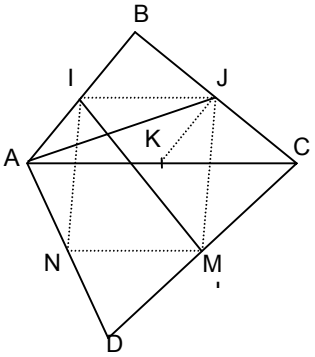
Mais nous pouvons utiliser une notation plus "parlante" :

```
((chasles (X Y) + (Y z) = (X Z))
(vecteur (X Y)) (vecteur (Y Z)) (vecteur (X Z)))
```

Mais, si (z J M N) est en parallélogramme, le relation de Chasles affirme que la somme dés vecteurs (I J) et (I N) est le vecteur (I M) ; en généralisant, nous obtenons :

```
((chasles (X Y) + (X Z) = (X x))
(parallélogramme (X Y x Z))
```

sous forme de soustraction, nous pouvons dire que la différence des vecteurs `X` et `Y` est le vecteur `Z` si la somme des vecteurs `Y` et `Z` est `X` soit :




```
((chasles X - Y = Z)
(chasles Y + Z x Z)
/* interrogation */
&.???((chasles (A I) + (A K) = (A J)))
Oui
&.???((chasles (A J) - (A K) = (A I)))
Oui
&.???((chasles (A J) - (A K) = (K J)))
Oui
&.resoudre((chasles (A B) + (B C) = x))
(chasles (A B)+ (B C) = (A C))
OK
&.resoudre((chasles (I J) + (I N) = x))
chasles (A B) + (B C) = (A C))
Ok
&.resoudre((chasles (I J) + (I N) = x))
(chasles (I J) + (I N) = (I M))
Ok
&.comment(chasles(I J) + (I N) = (I M))
```

Pour montrer `(chasles (I J) + (I N) = (I M))` j'utilise `(chasles (I J) + (I N) = (I M)`

```
(parallélogramme (I J M N))
```

Je peux montrer `(parallélogramme (I J M M))`

```
&.
```

10. Conclusion provisoire

MicroPROLOG permet de résoudre rapidement et convenablement certains problèmes. Il est facilement utilisable par des élèves non spécialisés en informatique... si l'on veut qu'il en soit ainsi !

Car il faut bien reconnaître que programmer en logique n'est pas aussi évident que cela peut paraître à travers cet article.

Cependant, plusieurs expériences ont été faites en classes de 2e, 1a1, en club (théorie des ensembles, dérivation formelle, relations binaires, parenthèsage d'expressions, jeux sur les graphes,.. sans oublier les autres disciplines) : à condition de rester dans certaines limites, MicroPROLOG permet une progression informatique rapide des élèves et un renouvellement des questions posées.

La machine vérifie, "d'une certaine façon", indépendamment de l'enseignant, que le programme écrit par l'élève "marche bien". Elle lui procure une impression de satisfaction vraisemblablement peu courante, et peut lui donner une indépendance vis à vis de l'enseignant non négligeable pour sa formation à l'autonomie.

BIBLIOGRAPHIE SOMMAIRE

PROLOG, InterEditions - F. Giannesini, H. Kanoui, R. Pasero et M. Van *Caneghem, préface de A. Colmerauer,*

PROLOG, bases théoriques et actuels, TSI, vol.2, n°4, 1983 - A. Colmerauer, H. Kanoui, M..van Caneghem

Micro-PROLOG : programmer en logique, Eyrolles - K.L. Clark et F.G. McCabe traduit par M. Rousseau

Programmation en Prolog, Eyrolles - W.F. CLoksin, C.S. Mellish documentation de l'IREM-CREFI de Marseille

Premier pas en MICRO-PROLOG, Essais 84-85 en Collèges et Lycées - CRDP Marseille - J.J. Achard, C. Dufossé

A lire également :

L'INTELLIGENCE ARTIFICIELLE, *La Recherche*, n° spécial 170, octobre 1985.

Pour la SCIENCE, numéro spécial informatique pour la science, n° 85, novembre 1984.

Systèmes Experts et Enseignement - *EPI* - n° spécial Décembre 84

ANNEXE

```
iface-mod (??? quel un résoudre comment) (oui non)
((??? X)
   (SI (? X) ((AL Oui )) ((AL Non))))
((quel (X|Y))
  (POURTT Y ((AL X)))
   (AL Ok))
((un X|Y))
   (? Y)
   (AL X)
   (A "Encore/oui/non? ")
  (L Z)
  (NON EQ Z oui))
((un X)
   (AL Ok))
((resoudre X)
  POURTT X ((AL X)))
  (AL Ok))
((est-explique(NON X))/
  (AL X echoue))
((est-explique (OU X Y))/
  (Si (? X)
    ((sont-expliques X)
    ((sont-expliques Y))))
((est-explique (X Y))
   (CL((X Y)))/
  (AL (X Y) est connu))
((est-explique X)
  (AL Je peux montrer X))
((sont-expliques ()))
((sont-expliques (X | Y))
  ((est-expliques X)
  ((sont-explique Y))
((est-demontre X Y)
  (CL (X|Y) Y Y)
  (AL X)
  (hypotheses Z))
  ((hypotheses ()))
  ((hypotheses (X|Y))
    (A " " X) (AL)
    (hypotheses Y))
((est-prouve X Y Z)
```

```
  (CL (X|Z) 1 Y)
  (? Z))
((comment (X Y))
   (SYS X)
  (X Y)
  /
  (AL (X|Y)est évident))
((Comment X)
   (CL(X))
   /
   (AL X est connu))
((comment X)
   (est-prouve X Y Z)
   /
  (AL Pour montrer X j'utilise)
  (est-demontre X Y) (AL)
  (sont-expliques Z))
((comment X)
(AL Je ne peux montrer X))
FEMOD
```