POUR EN SAVOIR PLUS SUR LES MODULES (MICRO-PROLOG)

Jean-Jacques ACHARD

I - STRUCTURE

Un module est défini par :

- un nom
- une liste export
- une liste import
- un programme
- FEMOD

Le nom du module : chaîne de caractères, exemple simple-mod.

Liste export : liste des prédicats exportables, définis à l'intérieur du module, mais utilisables de l'extérieur.

Liste import : prédicats ou constantes utilisés à l'intérieur du module, mais définis dans un autre module.

Le programme : écrit en micro-Prolog

FEMOD : marque la fin du module

Exemple 1 : instal-mod, module écrit pour SIL'Z II, MICRAL 8022 et LX 529 qui définit les relations de gestion d'écran. Il suffit de modifier les codes, pour l'adapter à chaque appareil.

Version SIL'Z

```
instal-mod
(efec iv pos cloche)
(.)
((efec .)
    (CAR X 26)
    (A X))
((iv)
```

```
    (CAR X 31)
    (A X))
((pos X Y)
    (CAR Z 27)
    (CAR x 61)
    ((SOM 32 X y)
    (CAR z y)
    (SOM 32 Y X1)
    (CAR Y1 X1)
    (CHAINE (Z x z YI) Z1)
    (A Z1)
((cloche)
    (CAR X 7)
    (A X))
FEMOD
```

Les 4 prédicats exportés seront importés par d'autres modules en particulier par les modules &, simp-mod.

Exemple 2 : simp-mod

```
simp-mod
(aj ac li ot su ed non app conc liste-des ?? ??? quel )
(dict et si tout pred . f 0 efec )
```

Le programme de ce module traite des relations qui gèrent la syntaxe de simple et la base de données de l'utilisateur.

Les autres relations sont définies dans l'autres modules.

Il faut déclarer les constantes : et si tout pred . f 0, en import pour qu'elles puissent être "reconnues" à l'intérieur de simp-mod. Exemple, pour : su tout, le programme demande de confirmer par 0. Si 0 est absent dans la liste import, la suppression de toute les clauses n'est jamais effectuée. La relation efec est utilisée par li, donc il faut l'importer sinon elle sera considérée comme non définie.

remarque : ce simple est écrit avec 4 modules

```
simp-mod
com-mod
instal-mod
erreur-mod
```

Ces 4 modules ont été rangés dans un même fichier, pour être en mémoire centrale tous les 4 en même temps.

II - COMMENT CRÉER UN MODULE

1. Écrire le programme sous micro-Prolog. On peut se passer de simple, mais en utilisant une relation pour l'édition des clauses

`((ED X)`	`X` : prédicat
`(L Y)`	`Y` : n° de la clause
`(C ((X Z) x) 1 Y)`	Recherche de la clause de rang `Y`
`(LTAMP (((X Z) x)) y)`	y : clause corrigée
`(AJCL y Y)`	y est ajoutée au rang `Y+1`
`(OTCL X Y))`	L'ancienne clause est supprimée

Les relations prédéfinies de micro-Prolog sont toutes en majuscules. La syntaxe est :

<clause> ::= ((tête de clause> <queue de clause>))
<tête de clause> ::= <prédicat> < suite d'arguments>))
<queue de clause> ::= <<condition> <queue de clause/ vide
<condition> ::= (<prédicat> <suite d'arguments>) /ARRET/FAUX//
<prédicat> ::= constante
<suite d'arguments> ::= <argument> <suite d'arguments> / vide
<argument ::= constante/nombre/variable/(<liste d'arguments)
<liste d'arguments> ::=(<suite d'arguments>)/ ()

Exemple :

sous simple, on écrit : `aj(efec(.) si CAR(X 26) et A(X))`

sous micro-Prolog, il suffit de taper : `((efec .)(CAR X 26) (A X))`

et la clause est enregistrée dans la base de données.

On utilisera les relations prédéfinies :

`LIST TT` ou `LIST <prédicat>` ; `SUPP TT` ou `SUPP <prédicat>`.

`OTCL` en écrivant : `?((OTCL <prédicat> <numéro> ))`

2. Ranger le programme dans un fichier, exemple sous le nom de PROG.

3. Méthode 1

Utiliser un éditeur de texte type éditeur de PASCAL ou WORDSTAR (pour fichier programme)

- on appelle le fichier PROG.LOG
- on insère le nom du module, les listes export, import et FEMOD.
- après sauvegarde, le fichier contient alors un module reconnu par micro-Prolog.

On pourra de même effectuer toutes les corrections. L'écriture de tout le module peut se faire sous éditeur (mais attention à la syntaxe !)

4. Méthode 2 :

Sous Micro-Prolog, écrire un programme de création de module avec CRMOD :

```
((cr mod)
    (AL Nom du module)
    (L X)
    (AL Nom du programme à ramener)
    (L Y)
    (export Z)
    (import x)
    (CRMOD X Z x)
    (RAM Y)
    (FEMOD)
    (AL Nom du fichier contenant le module X) (L z)
    SAUV z X))
((export (........ ) ))      liste à préparer
((import (........ ) ))      "       "
```

Il suffit pour lancer l'exécution de taper `cr mod` RC et d'indiquer le nom du module créé, le nom du programme à ramener, le nom du fichier contenant le module.

III - COMMENT CORRIGER UN MODULE

1. Utiliser un éditeur (voir II)

2. Sous micro-Prolog, charger le module puis entrer par :

OUMOD (nom du module) (attention : ne pas confondre avec le nom fichier). Exemple :

```
&.RAM SIMPLE
&.OUMOD simple-mod
simple-mod.    ---)
```
devient le nouveau prompt

On peut alors effectuer toutes les opérations, comme :

`simple-mod. RAM EDIT` ramener le fichier contenant ED

Une fois les corrections terminées, supprimer les relations comme ED.

Sortir du module par FEMOD :

`simple-mod. FEMOD x` `x` : caractère quelconque

Sauver le module corrigé :

```
&. ?((SAUV simple simple-mod))
```

IV - RELATIONS PRÉDÉFINIES

`RAM` (nom du fichier contenant le/les module(s))

`SAUV` (nom du fichier contenant le module) (nom du module)

`OUMOD` (nom du module)

`FEMOD` (caractère quelconque)

`CRMOD` (nom du module) (liste export) (liste import)

`MODC x` : pour connaître le module en service.

`(DICT X Y Z x)`

`X` : nom du module en service

`Y` : liste export

`Z` : liste import

`x` : dictionnaire du module en service

On peut lister `DICT` comme n'importe quelle relation par : `LIST DICT`.

Jean Jacques ACHARD
Ceyreste 13600
le 20.11.85