Langage reconnu

On reconnaît toutes les formules bien formées de la logique du premier ordre, avec les conventions d'écriture suivantes :

prédicats	Identificateurs en minuscules
variables	Lettres minuscules typées dans le lexique
constantes	Lettres minuscules typées dans le lexique
$\wedge, \vee, \rightarrow, \neg$	`&`, `|`, `->`, `~`
$\forall, \exists$	`A`, `E`

Pour simplifier les manipulation, on réservera les parenthèses à la délimitation des opérandes d'un connecteur binaire : les prédicats seront simplement suivis de leurs arguments.
Ex. «$\forall x\,(\text{homme}(x) \rightarrow \neg\text{femme}(x)$» s'écrit `Ax ( homme x -> ~ femme x )`

Interprétation Lorsqu'une telle formule est suivie d'un point d'interrogation, cette formule est interprétée par rapport au modèle courant.

Affectation Lorsqu'une formule (nécessairement atomique et sans variable) est suivie d'un point, elle est considérée comme une affectation du modèle. Pour faciliter la manipulation du modèle, on admettra aussi des expressions de la forme « `homme a = 0.` ».

Actions sémantiques

Les actions associées consisteront en la construction d'un arbre syntaxique représentant la formule. Ensuite, cet arbre sera utilisé soit pour l'affectation, soit pour l'interprétation.

Modèle

On fait l'hypothèse qu'on a au plus 5 individus, 5 prédicats unaires, et 5 prédicats binaires. Ces trois types d'objets sont chacun représentés par un entier compris entre 0 et 4. Un modèle pourra donc être représenté par deux tableaux, indicés par les éléments :

```
int predicats[5][5];
int relations[5][5][5];
```

N.B. Il n'est pas nécessaire que le langage comprenne 5 constantes de chaque catégorie.

Construction de l'arbre

L'arbre syntaxique est construit de façon classique, avec des cellules qui pourront avoir la forme suivante :

`n` :

type : `int`	
valeur : `int`	
fg : `n *`	fd : `n *`

Pour les types, on peut proposer la liste suivante : `VAR, CTE, P1, P2, C1, C2, Q`. On suggère de mettre la variable en fils gauche et la formule en fils droit pour les quantificateurs.
L'implémentation peut être réalisée au moyen d'une fonction `cree_noeud(type,val,fg,fd)` qui alloue une cellule et la remplit avec ses arguments.

Evaluation

L'évaluation des formules sans quantificateur (et sans variable) ne pose pas de problème particulier. On peut définir une fonction récursive qui renvoie 0 ou 1 selon le type de la racine et les valeurs de la fonction pour ses descendants. Cependant, pour évaluer toute sous-formule, il faut faire intervenir la fonction d'assignment (cf. polycopié « évaluation d'une formule »). On va donc définir une fonction d'évaluation qui prend en paramètre un argument supplémentaire, la fonction g.
L'implémentation est possible grâce aux hypothèses précédentes (nombre d'individus et nombre de variables borné dans le modèle). Ainsi l'ensemble des fonctions g possibles, qui associent une variable parmi 5 à un individu parmi 5, contient exactement 5^5 valeurs. On peut donc définir un type "assignment" (par exemple, un entier entre 0 et 3124 ($5^5 - 1$) interprété en base 5, ou un tableau de 5 entiers — pose des problèmes de mémoire —).