Projet de C++ n°1

Cours de C++

—*Licence d'informatique*—

Décembre 1999

Projet FOLIE

L'objet du projet FOLIE (FOrmules de Logique: Interprétation et Évaluation) est de manipuler des formules de logique.

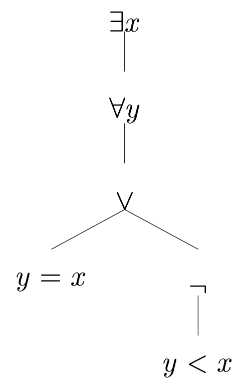
---

1 Le projet

Nous ne manipulerons dans le projet que des ensembles finis.

Le but du projet est un programme orienté objet permettant la manipulation de formules de logique.

Une formule de logique est composée de variables, de relations, de connecteurs et de quantificateurs (pour plus d'explications voir plus loin). Une formule de logique peut être représentée par son arbre syntaxique:



ou par son écriture usuelle

$$\exists x \forall y (y = x \vee \neg(y < x))$$

ou par son écriture préfixe

$$\exists x \forall y \vee = yx \neg < yx$$

Le programme devra permettre:

- la saisie de formules,
- la mise en forme prénexe des formules, c'est-à-dire la mise en tête des quantificateurs,
- la mise en forme normale disjonctive des formules, c'est à dire en forme prénexe de telle manière que la formule sans ses quantificateurs est une disjonction ($\vee$) de conjonction ($\wedge$) de formules atomiques,
- le calcul des variables libres d'une formule,
- de tester si une formule est un énoncé ou non,
- de quantifier universellement sur demande les variables libres d'une formule,
- d'évaluer un énoncé: tester si un énoncé est vrai quand les variables sont interprétées comme des entiers entre 0 et un entier positif `N` défini par l'utilisateur,
- d'afficher une formule.

Les formules et commandes sont saisies soit au clavier, soit en utilisant une interface graphique. En particulier, on pourra éviter de faire l'analyse lexicale et syntaxique d'un énoncé en utilisant une interface graphique qui permet de construire directement l'arbre qui représente la formule en cliquant sur des boutons ("Ajouter quantificateur", "Ajouter connecteur", "Ajouter relation", ...). La qualité de l'interactivité du programme avec l'utilisateur sera prise en compte dans la notation du projet.

Vous devrez:

- Fournir un dossier donnant notamment une description motivée de la hiérarchie des classes choisie pour implanter les formules de logique. Les choix devront être explicités et argumentés de façon détaillée dans le rapport. On mettra l'accent sur la délégation et les responsabilités de chaque objet.
- Implanter les différentes opérations citées, plus d'autres, bien sûr...
- Prévoir quelques jeux d'essai pertinents illustrant les fonctionnalités de base demandées au programme et mettant en valeur les particularités liées à vos choix.

Le but principal de ce projet est de concevoir une application orientée objet et la hiérarchie des classes qu'elle met en oeuvre. Vous porterez donc un intérêt particulier à cette phase de conception et de réalisation de sorte que l'application soit modulaire et facilement adaptable.

Tous les objets manipulés auront leur classe propre. Par exemple, la classe permettant de construire un objet $\exists x \phi$ (où ϕ est une formule de logique) n'est pas la même que celle permettant de construire un objet $\neg \phi'$. En revanche, elles ont un sur-type (classe) commun, qui est "formule de logique". On s'attachera à remonter (factoriser) les propriétés (fonctions et données) communes à différents sous-types le plus haut possible dans la hiérarchie des classes.

2 Un peu de logique

2.1 Spécifications

Dans la suite nous noterons **V** et **F** les valeur booléennes vrai et faux.

Les *variables* sont des objets ayant un nom et pouvant avoir une valeur numérique entière positive ou nulle inférieure à un entier `N` fixé.

Les *relations* que nous emploierons sont les comparateurs habituels $<$ et $=$ sur les entiers naturels. On se limitera à ces deux là.

Les *connecteurs* sont le $\neg$ (non), $\vee$ (ou), $\wedge$ (et) dont les tables de vérités sont:

$\vee$	**F**	**V**
F	**F**	**V**
V	**V**	**V**

$\wedge$	**F**	**V**
F	**F**	**F**
V	**F**	**V**

$\neg$	**F**	**V**
	V	**F**

Les quantificateurs sont $\forall$ (pour tout) et $\exists$ (il existe), avec la signification usuelle.

Nous dirons qu'une formule de logique est vraie si elle est vraie pour des valeurs entières comprises entre 0 et `N`, où `N` est un entier positif fixé. Par exemple, la formule $\exists x \forall y (y = x) \vee \neg(y < x)$ est vraie, car $(y = x) \vee \neg(y < x)$ signifie que y est supérieur ou égal à x, et, pour $x = 0$, la formule revient à dire que y est positif ou nul, ce qui est bien le cas pour y compris entre 0 et un entier `N` fixé.

2.2 Terminologie

2.2.1 Divers

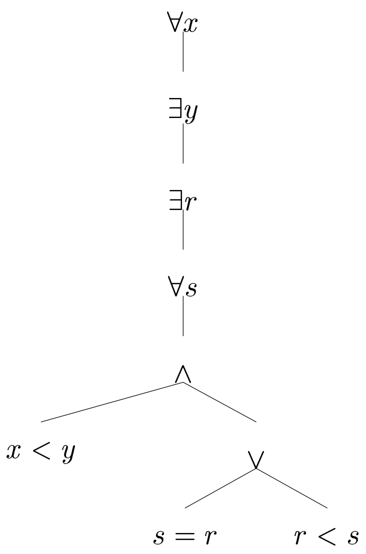
Une *formule atomique* est de la forme $x = y$ ou $x < y$, avec x et y deux variables.

Une variable est *libre* si aucun quantificateur n'y fait référence.

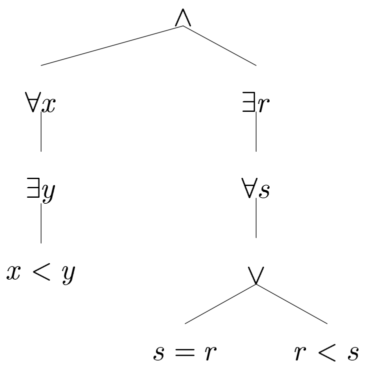
Une *formule close* ou *énoncé* est une formule dont aucune variable n'est libre.

2.2.2 Forme prénexe

Une formule est en *forme prénexe* si tous les quantificateurs sont en tête. Par exemple, la formule



est en forme prénexe, et la formule



ne l'est pas. Cependant la première est une forme prénexe de la seconde.

La mise en forme prénexe d'une formule se réalise récursivement sur son arbre syntaxique. On fera attention à portée des quantificateurs lors de la mise en forme prénexe des formules. Par exemple, la formule

$$\forall x \exists z \forall y((x < z) \wedge (y = z)) \tag{1}$$

n'est pas une forme prénexe de

$$(\forall x(x < z)) \wedge (\exists z \forall y(y = z)) \tag{2}$$

puisque dans la seconde z est libre et pas dans la première. On peut aussi donner l'exemple de

$$\forall x \exists x((x < z) \wedge (x = z)) \tag{3}$$

qui n'est pas une forme prénexe de

$$(\forall x(x < z)) \wedge (\exists x(x = z)) \tag{4}$$