*[26 avril 2011]* L'objectif de la séance est de réaliser un programme qui fait l'analyse syntaxique de formules logiques et évalue ces formules (ou certaines d'entre elles) après les avoir mises sous forme normale.

### Logique des propositions

1. Avec un vocabulaire composé des lettres majuscules {`A`, ..., `Z`}, des parenthèses et des connecteurs logiques { `&`, `|`, `>` et `~`}, réaliser un parser `ply`, avec la grammaire la plus simple possible, qui reconnaît toutes les formules de la logique des propositions **complètement** parenthésées.
2. En supposant une valeur arbitraire pour chaque variable propositionnelle (p. ex. `A`, `C`, `E`, ... valent 1 et `B`, `D`, ... valent 0), faire en sorte que la valeur de chaque formule reconnue soit calculée (en utilisant un attribut de la grammaire).
3. Sans utiliser le mécanisme de priorités de `ply`, proposer une nouvelle grammaire qui reconnaisse les formules de la logique des proposition **incomplètement** parenthésées, avec les priorités habituelles.
4. Faire en sorte de produire pendant l'analyse une représentation de la formule logique (par exemple, la formule `A > (B | C)` pourrait correspondre à l'arbre
   `(imp, (var, A), (disj, (var, B), (var, C)))`)
   et prévoir une méthode qui affiche de façon lisible une telle structure interne.
5. Réaliser une mise sous forme normale de la formule : on choisira comme forme normale ce qu'on appelle la *forme normale conjonctive*, c'est-à-dire une conjonction de *clauses*, une clause étant une disjonction de *littéraux*, un littéral étant une variable propositionnelle ou sa négation.

   **Algorithme de normalisation**

   (a) Remplacer les formules de la forme $(\varphi \to \psi)$ par $(\neg\varphi \vee \psi)$
   (après avoir remplacé si nécessaire les formules de la forme $(\varphi \leftrightarrow \psi)$ par $((\varphi \to \psi) \wedge (\psi \to \varphi))$)

   (b) Appliquer autant de fois que possible les règles de réécriture dérivées des lois de De Morgan : $\neg(\varphi \wedge \psi) \rightsquigarrow (\neg\varphi \vee \neg\psi)$ et $\neg(\varphi \vee \psi) \rightsquigarrow (\neg\varphi \wedge \neg\psi)$

   (c) Annuler les doubles négations

   (d) Appliquer autant que possible les règles de réécriture dérivées des lois de distributivité :
   $(\varphi \vee (\psi \wedge \zeta)) \rightsquigarrow ((\varphi \vee \psi) \wedge (\varphi \vee \zeta))$ et $((\varphi \wedge \psi) \vee \zeta) \rightsquigarrow ((\varphi \vee \zeta) \wedge (\psi \vee \zeta))$
6. [Bonus] Procéder à une simplification (qui peut être faite en cours de normalisation) de la forme normale.

### Logique des prédicats

1. En supposant un vocabulaire fixe et relativement restreint (par exemple, variables : `x`, `y`, `z`, `t`, constantes : `a`, `b`, `c`, prédicats `P`, `Q`, `R` ; `A` pour le quantificateur universel, `E` pour le quantificateur existentiel...), proposer un parser `ply` qui reconnaît les formules de la logique des prédicats.
2. Faire en sorte que le parser créée une représentation arborescente de la formule logique.
3. Proposer une normalisation de la formule en forme *prenexe* (formule composée d'un *préfixe* comportant un nombre fini de quantificateurs, suivi d'un *noyau*, sous-formule ne comprenant aucun quantificateur). Algorithme détaillé pages suivantes.

avons vu que la transformation d'une formule de la logique des propositions en une de ces formes normales permettait de simplifier les algorithmes de preuve de consistance ou de validité (§ 1.1.11-14). Ce sont des raisons analogues qui conduisent à introduire les formes normales du calcul des prédicats.

Les problèmes liés à la quantification et à la portée des quantificateurs sont parfois gênants. En particulier, la présence simultanée d'occurrences libres et liées d'une même variable dans une formule peut conduire à des difficultés. La situation est nettement plus simple dans le cas des formes prénexes. Une *forme prénexe* est une formule se composant d'une matrice précédée d'un *préfixe*, c'est-à-dire d'une suite finie de quantifications. Ces quantifications concernent des variables distinctes, et leur ordre est généralement crucial.[5] Une telle formule est donc de la forme

$$Q_1 x_1 \ldots Q_n x_n \, M \,,$$

où le symbole $Q_i$ désigne soit $\forall$, soit $\exists$, pour $i = 1, \ldots, n$, et où $M$ est une formule (la matrice) ne contenant pas de quantification.

On peut, sans réelle restriction, exiger que seules des variables ayant une occurrence (nécessairement libre) dans la matrice puissent être quantifiées. En effet, d'après les règles d'interprétation, si une formule $A$ ne comporte pas d'occurrence de la variable $x$, alors les formules $A$, $\exists\, x\, A$ et $\forall\, x\, A$ ont même valeur de vérité, pour toute interprétation.

L'intérêt des formes prénexes est lié au théorème suivant.
**Théorème 1.9.** *Pour toute formule logique il existe une forme prénexe qui lui est logiquement équivalente.*

**Preuve.** L'algorithme d'obtention de la forme prénexe est fort simple. Les étapes sont les suivantes.

- Eliminer les connecteurs d'équivalence et d'implication, au moyen des règles de réécriture vues au paragraphe 1.1.10.
- Renommer des variables liées (si nécessaire), de manière à ce qu'aucune variable n'ait simultanément des occurrences libres et liées. Cette condition est requise non seulement pour la formule traitée mais aussi pour toutes ses sous-formules.

[5] On peut aussi admettre, en cas de quantifications répétées sur une même variable, que seule la quantification la plus à droite est pertinente : cela est conforme à la règle générale d'interprétation des formules quantifiées.

- Supprimer les quantifications dont la portée ne contient pas d'occurrence de la variable quantifiée; ces quantifications sont en effet inutiles.
- Transférer toutes les occurrences de la négation immédiatement devant les atomes, en utilisant les règles de réécriture suivantes :

$$\begin{aligned}
&\neg\forall x\, A \longrightarrow \exists x \neg A\,,\\
&\neg\exists x\, A \longrightarrow \forall x \neg A\,,\\
&\neg(A \wedge B) \longrightarrow (\neg A \vee \neg B)\,,\\
&\neg(A \vee B) \longrightarrow (\neg A \wedge \neg B)\,,\\
&\neg\neg A \longrightarrow A\,.
\end{aligned}$$

- Transférer les quantifications en tête de la formule. On utilise à cet effet un ensemble de règles de réécriture. Nous donnons ici les règles relatives à la conjonction. Celles relatives à la disjonction s'en déduisent par dualité.

$$\begin{aligned}
&(\forall x A \wedge \forall x B) \longrightarrow \forall x(A \wedge B)\,,\\
&(\forall x A \wedge B) \longrightarrow \forall x(A \wedge B) \qquad \text{si } B \text{ ne contient pas } x\,,\\
&(A \wedge \forall x B) \longrightarrow \forall x(A \wedge B) \qquad \text{si } A \text{ ne contient pas } x\,,\\
&(\exists x A \wedge B) \longrightarrow \exists x(A \wedge B) \qquad \text{si } B \text{ ne contient pas } x\,,\\
&(A \wedge \exists x B) \longrightarrow \exists x(A \wedge B) \qquad \text{si } A \text{ ne contient pas } x\,.
\end{aligned}$$

  Pour que cet ensemble de règles soit complet, il est nécessaire d'ajouter la possibilité de renommer certaines variables liées; par exemple, la formule $\exists x\, P(x) \wedge \forall x\, Q(x)$ sera d'abord transformée en $\exists x\, P(x) \wedge \forall y\, Q(y)$ avant l'application des règles de réécriture. Notons aussi qu'à tout moment, les propriétés de commutativité, d'associativité et d'idempotence des connecteurs $\wedge$ et $\vee$ (§ 1.1.9) peuvent être mises à profit pour simplifier les formules.

Le théorème est ainsi démontré, de manière constructive. □

En guise d'application, nous cherchons une forme prénexe équivalente à la formule

$$\forall x[P(x) \wedge \forall y \exists x(\neg Q(x,y) \supset \forall z\, R(a,x,y))]\,.$$

Les étapes successives de cette recherche sont énumérées ci-dessous.