TP : normalisation de formules logiques

L'objectif de la séance est de réaliser un programme qui fait l'analyse syntaxique de formules logiques (propositionnelles), les évalue, et les simplifie après les avoir mises sous forme normale.

Les 3 premières questions sont dans une logique d'*interprétation*, puisqu'il s'agit de faire des calculs à la volée ; les suivantes sont dans une logique de *compilation*, dans le sens où il va s'agir de produire un arbre (l'arbre syntaxique de la formule) sur lequel un traitement (la normalisation) peut intervenir, *on-line* ou *off-line*. La question 3 est indépendante.

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.

   Rappel : syntaxe des formules propositionnelles

   (a) Tous les symboles de propositions sont des formules de L_p.
   (b) Si φ est une formule de L_p, alors $\neg\varphi$ est une formule de L_p.
   (c) Si φ et ψ sont des formules de L_p, alors $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$, et $(\varphi \leftrightarrow \psi)$ sont des formules de L_p.
   (d) Rien d'autre n'est une formule.

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 propositions **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.

Date limite de rendu du TP : dimanche 13 décembre.
On demande 6 programmes python distincts, exécutables (à un changement de chemin près), abondamment commentés, contenant chacun un petit jeu d'essai.