Cette *vingt-neuvième* colle vous fera travailler sur les définitions et premiers résultats du cours de logique.

Formules logiques vues comme des arbres

On considère la formule logique $P = (x \vee y) \wedge (z \vee \neg x)$

1. Représenter l'arbre correspondant à P.
2. Donner l'ensemble des variables propositionnelles apparaissant dans la formule P ? (noté $\mathcal{V}(P)$ dans le cours)
3. Quelle est la hauteur $h(P)$ et la taille $t(P)$ de cette formule ?
4. Donner toutes les sous-formules de P.

Substitutions

Toujours sur la formule P, on considère les substitutions suivantes :

— a) $P_1 = P[Q_1/x]$, avec $Q_1 = \neg y$.
— b) $P_2 = P[Q_2/x]$, avec $Q_2 = w$ (une autre variable propositionnelle).
— c) $P_3 = P[Q_3/y][Q_4/z]$, avec $Q_3 = \neg x$ et $Q_4 = x$.
— d) $P_4 = P[Q_4/z][Q_3/y]$.

5. Pour chaque substitution, donner la formule obtenue.
6. Lesquelles de ces P_1, P_2, P_3 et P_4 sont égales ?

Valuations

7. Si possible, donner deux valuations ϕ_1 et ϕ_2, telles que leur évaluation de la formule P soit respectivement à V (vrai) et à F (faux).
8. Donner la table de vérité de la formule P. Combien a-t-elle de lignes ?
9. En déduire si P est une tautologie, une formule satisfiable, ou bien une formule insatisfiable.
10. Peut-on en déduire une formule simplifiée P' (de taille et/ou hauteur plus petite que P) qui soit sémantiquement équivalente à P ? (ce que l'on a noté $P \equiv P'$ dans le cours, c'est-à-dire $P \models P'$ et réciproquement $P' \models P$).

Implémentation rapide (en C)

Vous utiliserez `https://www.onlinegdb.com/` ou `https://replit.com/languages/c` pour coder en C les questions suivantes :

11. Écrire une fonction `bool formule_P(bool x, bool y, bool z)` qui réalise l'évaluation de la formule P selon le contexte ϕ donné par les valeurs de vérités des trois variables x, y et z.

12. En déduire une fonction `void affiche_table_verite_P(void)` qui affiche au format suivant la table de vérité de la formule P :

```
| x | y | z | P |
-----------------
| 0 | 0 | 0 | ? |
...
| 1 | 1 | 1 | ? |
```

13. Est-ce bien cohérent avec les résultats donnés en question 8. ?
14. Écrire une fonction `int nb_valuations_sat_P(void)`, qui renvoie un entier comptant le nombre de valuations (sur les variables apparaissant dans P, ie. dans $\mathcal{V}(P)$) qui satisfont la formule P.
15. En déduire une fonction `int nature_formule_P(void)`, qui renvoie un entier valant `0` si la formule P est insatisfiable, `2` si elle est tautologique, et `1` si elle est satisfiable mais non tautologique.

Encore un peu...

Sur les formules vues comme des arbres

16. Donner une formule Q_n qui dépend d'un paramètre entier n, telle que Q_n n'ait qu'une seule variable, mais soit de taille $\Theta(n)$ et de hauteur $\Theta(n)$.
17. Faire de même, mais en donnant une formule R_n de taille $\Theta(n)$ mais de hauteur $\Theta(\log_2(n))$.
18. Peut-on faire l'inverse, avec une formule S_n de taille $\Theta(\log_2(n))$ mais de hauteur $\Theta(n)$?

Sur les conséquences et équivalences logiques

19. Donner un exemple de votre choix de formules P_{19} et Q_{19} telles que $P_{19} \models Q_{19}$.
20. De même avec une formule Q_{20} et un ensemble Γ_{20} d'au moins deux formules, tels que $\Gamma_{20} \models Q_{20}$.
21. De même avec deux formules P_{21} et Q_{21} telles que $P_{21} \equiv Q_{21}$. On les prendra bien sûr différentes.

Bonus : implémentation en OCaml

22. Faire de même mais en OCaml.

