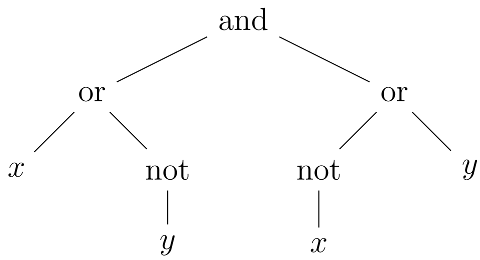
Formules logiques

Florian Bourse

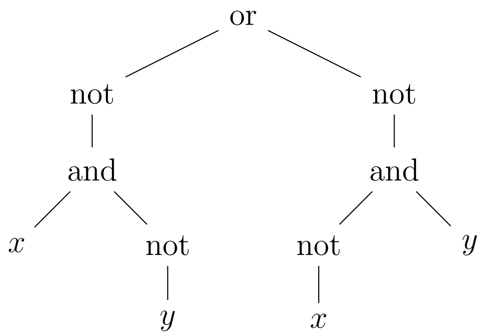
1 Syntaxe des formules logiques

Question 1. Écrire comme mots les formules logiques représentées par les arbres suivants :

Arbre 1



Arbre 2



Question 2. On choisit de représenter les formules en OCaml comme suit :

Une définition des formules logiques en OCaml

```ocaml
type formula =
  | V
  | F
  | Var of int
  | Not of formula
  | And of formula * formula
  | Or of formula * formula
  | Imp of formula * formula
  | Eq of formula * formula;;
```

Représenter les arbres syntaxiques des formules logiques suivantes, puis les écrire comme mots.

Exemples de formules

```ocaml
let f1 = Imp(Or(Var 1, Var 2),Or(Var 3, Var 2));;
let f2 = Or(Imp(Var 1, Var 2),Not(Var 1));;
let f3 = And(Or(Or(Var 1,Var 2),Not(Var 3)),Var 4);;
let f4 = Or(Var 1,And(Var 2,Or(Not(Var 3),Var 4)));;
```

2 Sémantique de vérité du calcul propositionnel

Question 3. x et y désignent deux variables propositionnelles. On pose $\varphi = (x \vee y)$ et $\psi = (x \to y)$, ainsi que $\omega = (\varphi \to \psi)$.

1. Quelles valuations donnent la même valeur aux formules φ et ψ ?
2. Quel est l'ensemble des modèles de ω, noté $\mathrm{mod}(\omega)$?
3. Une formule est dite *contingente* si elle est satisfiable sans être tautologique. La formule ω est-elle contingente ou tautologique ?

Question 4. x et y désignent deux variables propositionnelles. Parmi les trois formules suivantes, l'une est une tautologie, une autre est contingente et une dernière est insatisfiable. Les identifier.

$$\varphi_1 = (x \to y) \to y$$

$$\varphi_2 = x \to (y \to x)$$

$$\varphi_3 = (x \wedge y) \leftrightarrow (x \to \neg y)$$

Question 5. φ et ψ désignent deux formules logiques.

1. Montrer les deux équivalences suivantes.
   (a) $(\varphi \to \psi) \equiv (\neg\varphi \vee \psi)$
   (b) $(\varphi \leftrightarrow \psi) \equiv ((\varphi \to \psi) \wedge (\psi \to \varphi))$
2. En déduire deux formules sémantiquement équivalentes aux formules suivantes dans lesquelles seuls les connecteurs $\neg$, $\wedge$ et $\vee$ apparaissent.
   (a) $(\varphi \leftrightarrow \psi)$
   (b) $(((\varphi \to \psi) \wedge \varphi) \to \psi)$

Question 6. Étant donné trois formules logiques φ, ψ, ω, on définit l'opérateur ternaire Δ par : $\Delta(\varphi, \psi, \omega) = (\varphi \wedge \psi) \vee (\neq \varphi \wedge \omega)$.

1. Écrire sous la forme la plus simple les formules $\Delta(\top, \psi, \omega)$ et $\Delta(\bot, \psi, \omega)$.
2. Exprimer $(\varphi \wedge \psi)$ et $(\varphi \vee \psi)$ par une formule ne comportant qu'un seul Δ.
3. Exprimer $(\neg\varphi)$ en fonction de Δ, de $\top$ et de $\bot$.

Question 7. φ et ψ désignent deux propositions logiques.

1. (a) Prouver l'équivalence $(\varphi \to \psi) \equiv (\neg\psi \to \neg\varphi)$ *(contraposition)*
   (b) Prouver l'équivalence $(\varphi \to (\psi \to \omega)) \equiv ((\varphi \wedge \psi) \to \omega)$ *(exportation)*
   (c) Prouver que $(((\varphi \to \psi) \to \varphi) \to \varphi)$ est une tautologie *(loi de Pierce)*
   (d) Si x, y, z sont trois variables propositionnelles, que dire de la formule : $((x \to y) \wedge (y \to z))$?
2. (a) Prouver que $(\varphi \leftrightarrow \psi) \equiv ((\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi))$.
   (b) Prouver que $\varphi \equiv \psi$ si et seulement si $(\varphi \leftrightarrow \psi)$ est une tautologie.

3 Sudoku

Question 8. On suppose donnée une grille de jeu avec un certain nombre de cases déjà remplies. Afin de compléter la grille, on va définir une formule logique qui, si elle est satisfiable, permet de construire une solution. Pour modéliser le problème en termes logiques, on adopte les notations suivantes.

— Chaque ligne du tableau est repérée par un entier $r \in [\![0, 8]\!]$.

— Chaque colonne du tableau est repérée par un entier $c \in [\![0, 8]\!]$.

— Chaque case contient un entier $k \in [\![1, 9]\!]$.

On définit alors une variable propositionnelle $x_{r,c,k}$ dont la valeur de vérité est V si la case (r, c) du tableau solution contient la valeur k ; F sinon.

1. (a) Sachant que chaque case doit contenir au moins un entier non nul, écrire une CNF C_1 qui exprime cette contrainte.
   (b) Sachant que chaque case doit contenir au plus un entier non nul, écrire une CNF C_2 qui exprime cette contrainte.
   (c) Sachant que chaque ligne doit contenir exactement une seule occurence de chaque entier, écrire une CNF C_3 qui exprime cette contrainte.
   (d) Sachant que chaque colonne doit contenir exactement une seule occurence de chaque entier, écrire une CNF C_4 qui exprime cette contrainte.
   (e) Sachant que chaque sous-groupe 3×3 doit contenir exactement une seule occurence de chaque entier, écrire une CNF C_5 qui exprime cette contrainte.
   (f) On note I l'ensemble des triplets (r, c, k) des cases (r, c) et des entiers k initialement présents dans la grille. Écrire une CNF C_0 qui traduit la présence de ces chiffres dans la solution du problème.
   (g) En déduire une formule φ associée au problème du Sudoku.
2. (a) Combien de variables propositionnelles le problème comporte-t-il ?
   (b) Si on omet la clause C_0, combien φ comporte-t-elle de clauses disjonctives ?
3. En généralisant le problème du Sudoku à une grille $n \times n$ où $n = d^2$, $d \in \mathbb{N}^*$, montrer que le nombre de clauses disjonctives dans φ est un $O(n^4)$.
4. Si la formule est satisfiable, comment la connaissance des valeurs de vérité de chaque variable permet-elle la construction de la solution ?

4 Logique du premier ordre

Question 9. Traduire en formule logiques du premier ordre les phrases suivantes. Introduire tous les prédicats nécessaires.

1. Dans une école, il existe des ordinateurs non connectés au réseau local.
2. Dans les écoles, tous les ordinateurs sont connectés à un réseau local.
3. Dans chaque école, au moins un ordinateur est connecté à la fois au réseau local et à internet.

Question 10. On cherche à déterminer si un tableau a contient un doublon, c'est-à-dire un élément apparaissant deux fois. On se donne la fonction C suivante :

```c
bool duplicate(int a[], int n) {
        for (int i = 0; i < n; i = i + 1) {
                for (int j = i + 1; j < n; j = j + 1) {
                        if (a[i] == a[j]) { return true; }
                }
        }
        return false;
}
```

1. Voici quatre formules à propos d'un tableau a de taille n.
   (a) $\forall i \in [0, n[.\forall j \in [0, n[.i \neq j \rightarrow t[i] = t[j]$
   (b) $\forall i \in [0, n[.\exists j \in [0, n[.i \neq j \wedge t[i] = t[j]$
   (c) $\exists i \in [0, n[.\forall j \in [0, n[.i \neq j \rightarrow t[i] = t[j]$
   (d) $\exists i \in [0, n[.\exists j \in [0, n[.i \neq j \wedge t[i] = t[j]$

   Pour chacune de ces formules, donner une traduction en langue naturelle et un exemple de tableau validant la formule. Quelle formule exprime effectivement le plus fidèlement la présence d'un doublon ?
2. Donner des invariants pour les deux boucles de la fonction `duplicate`, écrits comme des formules de logique du premier ordre.

Question 11. On considère le problème de la recherche d'un motif m dans un texte t, et on se donne la fonction C suivante. La fonction renvoie l'indice de début d'une occurence de m dans t s'il en existe une, et -1 sinon.

```c
int search(char *m, char *t) {
        int lm = strlen(m), lt = strlen(t);
        for (int i = 0; i <= lt -lm; i++) {
                int j = 0;
                while (j < lm) {
                        if (m[j] != t[i+j]) break;
                        j++;
                }
                if (j == lm) return i;
        }
        return -1;
}
```

1. Donner des formules de logique du premier ordre exprimant la spécification du problème. Dans ces formules, on dénotera le résultat renvoyé par r.
2. Donner des formules de logique du premier ordre exprimant les invariants de deux boucles.