Interrogation du Lundi 30 Mars

Exercice 1 (Déduction naturelle)

On rappelle les règles d'inférence en déduction naturelle suivantes :

— Axiome :

$$\frac{}{\Gamma, F \vdash F}(\text{ax})$$

— Affaiblissement :

$$\frac{\Gamma \vdash F}{\Gamma, G \vdash F}(\text{aff})$$

— Implication :

$$\frac{\Gamma, F \vdash G}{\Gamma \vdash F \to G}(\to_i)\,, \qquad \frac{\Gamma \vdash F \quad \Gamma \vdash F \to G}{\Gamma \vdash G}(\to_e)\,.$$

— Conjonction :

$$\frac{\Gamma \vdash F \quad \Gamma \vdash G}{\Gamma \vdash F \wedge G}(\wedge_i)\,, \qquad \frac{\Gamma \vdash F \wedge G}{\Gamma \vdash F}(\wedge_e^1)\,, \qquad \frac{\Gamma \vdash F \wedge G}{\Gamma \vdash G}(\wedge_e^2)\,.$$

— Disjonction :

$$\frac{\Gamma \vdash F}{\Gamma \vdash F \vee G}(\vee_i^1)\,, \qquad \frac{\Gamma \vdash G}{\Gamma \vdash F \vee G}(\vee_i^2)\,, \qquad \frac{\Gamma \vdash F \vee G \quad \Gamma, F \vdash H \quad \Gamma, G \vdash H}{\Gamma \vdash H}(\vee_e)\,.$$

— Négation :

$$\frac{\Gamma, F \vdash \bot}{\Gamma \vdash \neg F}(\neg_i)\,, \qquad \frac{\Gamma \vdash F \quad \Gamma \vdash \neg F}{\Gamma \vdash \bot}(\neg_e)\,, \qquad \frac{\Gamma, \neg F \vdash \bot}{\Gamma \vdash F}(\neg_c)\,.$$

Soient p, q, r des formules propositionnelles. Prouver les formules suivantes en utilisant les règles d'inférence ci-dessus (on indiquera à chaque étape la règle utilisée) :

1. $\vdash p \to \neg\neg p$.
2. $\vdash (p \to q) \to (\neg q \to \neg p)$.
3. $p \to q, q \to r \vdash p \to r$.
4. $\vdash (p \wedge q) \to (p \wedge (\neg p \vee q))$.

Exercice 2 (Backtracking)

Pour éviter de faire la table de vérité d'une proposition logique, on peut utiliser la méthode du backtracking pour résoudre le problème SAT d'une proposition.

1. On considère, sous sa forme normale conjonctive, la proposition logique suivante :

$$F(a, b, c, d) = (b \vee c) \wedge (\neg a \vee \neg b \vee c) \wedge (\neg c \vee \neg d) \wedge (\neg a \vee \neg c \vee d) \wedge (a \vee d).$$

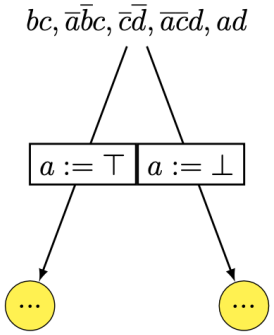
Pour simplifier la notation, nous choisirons de noter la formule de la façon suivante :

$$F(a, b, c, d) = bc, \overline{a}\overline{b}c, \overline{c}\overline{d}, \overline{a}\overline{c}d, ad.$$

(a) On suppose que a prend la valeur vrai ($\top$), déterminer la forme normale conjonctive de $F(\top, b, c, d)$.

(b) On suppose que a prend la valeur faux ($\bot$), déterminer la forme normale conjonctive de $F(\bot, b, c, d)$.

(c) Compléter l'arbre suivant :



2. Montrer, en utilisant un arbre, l'insatisfiabilité de la formule suivante :

$$G(a, b, c) = abc, a\overline{b}, b\overline{c}, \overline{a}\overline{b}\overline{c}, \overline{a}c.$$

3. Pour résoudre le problème en OCaml, on définira une formule à l'aide de listes de listes. Une clause est donc représentée par une liste avec autant d'éléments que de variable. Pour chaque littéral positif, on associera la valeur 1, et -1 pour un littéral négatif. Si la variable n'apparait pas dans la clause, on associera la valeur 0.

   Par exemple, pour la formule F, on aura alors :

```
let exemple_F =
[[0;1;1;0];[-1;-1;1;0];[0;0;-1;-1];[-1;0;-1;1];[1;0;0;1]];;
```

   (a) Écrire la fonction `positive : int list list -> int list list` qui prend en argument une proposition et renvoie la proposition pour laquelle la première variable prend la valeur vraie.

   Par exemple,

```
positive exemple_F;;
-: int list list = [[1;1;0];[-1;1;0];[0;-1;-1];[0;-1;1]]
```

   (b) Écrire la fonction `negative : int list list -> int list list` qui prend en argument une proposition et renvoie la proposition pour laquelle la première variable prend la valeur faux.

   Par exemple,

```
negative exemple_F;;
-: int list list = [[1;1;0];[0;-1;-1];[0;0;1]]
```

   (c) Écrire une fonction `solution : int list list -> bool` qui prend une liste de listes, représentant un état terminal dans l'arbre, et renvoie un booléen. On a deux cas :
   — Soit la dernière liste est vide, donc toutes les clauses ont été éliminées. La proposition est donc vraie.
   — Soit la liste contient les clauses constituées d'un seul littéral. Il faut alors qu'il soit toujours du même signe (si les clauses contiennent que des zéros, alors la proposition est fausse...).

   (d) En déduire une fonction `recherche_sat : int list list -> bool` qui utilise la méthode du backtracking pour résoudre le problème SAT.

Exercice 3 (Théorie des automates et des langages rationnels)

Dans tout cet exercice, la lettre ε désigne le mot vide, Σ désigne un alphabet et Σ^* l'ensemble des mot finis sur Σ.

Définition. Un **automate déterministe** A est un quintuplet $A = (Q, \Sigma, q_0, F, \delta)$, avec :
- Q un ensemble d'états ;
- Σ un alphabet ;
- q_0 l'état initial ;
- $F \subset Q$ un ensemble d'états finaux ;
- $\delta : Q \times \Sigma \to Q$ une application de transition.

Définition. Un **langage** sur Σ est une partie de Σ^*.

Définition. Soit $A = (Q, \Sigma, q_0, F, \delta)$ un automate déterministe. On définit de manière récursive $\delta^* : Q \times \Sigma^* \to Q$ par :

$$\forall q \in Q, \quad \delta^*(q, \varepsilon) = q \quad \text{et} \quad \forall q \in Q,\ \forall a \in \Sigma,\ \forall w \in \Sigma^*, \quad \delta^*(q, aw) = \delta^*(\delta(q, a), w).$$

Cette application vérifie alors la propriété admise suivante :

$$\forall q \in Q,\ \forall v \in \Sigma^*,\ \forall w \in \Sigma^*, \quad \delta^*(q, vw) = \delta^*(\delta^*(q, v), w).$$

Définition. On rappelle que les langages rationnels sont définis de manière inductive par :
- l'ensemble $\varnothing$ est un langage rationnel ;
- les langages $\{a\}$ où a est une lettre, sont rationnels ;
- si L et L' sont des langages rationnels, $L.L'$, $L \cap L'$, $L \cup L'$ sont des langages rationnels ;
- si L est un langage rationnel, L^* est un langage rationnel.

Définition. Soit L un langage sur Σ. Le **carré du langage** L est l'ensemble $\{uu \mid u \in L\}$. Il est noté $L \odot L$.

Définition. Soit L un langage sur Σ. La **racine carrée du langage** L est l'ensemble $\{u \in \Sigma^* \mid uu \in L\}$. Elle est notée $\sqrt{L}$.

On pourra utiliser sans démonstration le théorème ci-dessous :

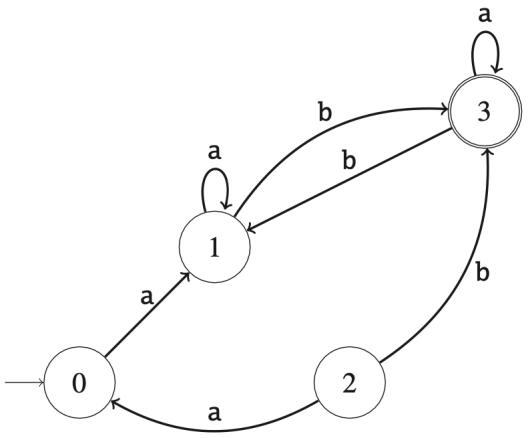
Théorème. Soit L un langage. Il est rationnel si et seulement s'il existe un automate déterministe et fini le reconnaissant.

Dans la suite, on décrit un langage rationnel par une expression rationnelle.

1. Décrire $\sqrt{L}$ lorsque $\Sigma = \{a, b\}$ et L est décrit par l'expression rationnelle a^*b^*.
2. Décrire $\sqrt{L}$ lorsque $\Sigma = \{a, b\}$ et L est décrit par l'expression rationnelle $b^*a^*b^*$.

Définition. Soit $A = (Q, \Sigma, q_0, F, \delta)$ un automate fini déterministe, q' un élément de Q et F' une partie de Q. L'automate $(Q, \Sigma, q', F', \delta)$ est noté $A_{q',F'}$. Si on note L le langage reconnu par A, $L_{q',F'}$ désigne le langage reconnu par $A_{q',F'}$.

Ici, L désigne le langage reconnu par l'automate A suivant :



3. Construire un automate reconnaissant $L_{3,\{1\}}$ en modifiant légèrement l'automate A.
4. On veut construire l'automate de Glushkov de L décrit par $a(a + ba^*b)^*ba^*$.
   (a) Décrire L', le linéarisé de L.
   (b) Déterminer les préfixes de L' de longueur 1, les suffixes de L' de longueur 1 et les facteurs de L' de longueur 2.
   (c) En déduire l'automate de Glushkov G de L.
5. Déterminiser l'automate G.

On fixe L un langage rationnel sur un alphabet Σ et $A = (Q, \Sigma, q_0, F, \delta)$ un automate fini reconnaissant celui-ci.

6. Soit u un mot de Σ^*. Montrer que u est un élément de $\sqrt{L}$ si et seulement s'il existe un $q \in Q$ tel que $u \in L_{q_0,\{q\}}$ et $u \in L_{q,F}$.
7. En déduire que $\sqrt{L}$ est un langage rationnel.
8. Montrer que l'on a $\left(\sqrt{L} \odot \sqrt{L}\right) \subset L$. Que peut-on dire de l'inclusion réciproque ?