A.1 Introduction au parsing

1. Soit la grammaire $S \rightarrow aB \mid bS \mid cC \mid c \mid bD$
   $B \rightarrow aS$
   $C \rightarrow aC \mid a$
   $D \rightarrow aB \mid bC \mid b$
   Détailler l'analyse descendante que l'on peut faire de *bbb*. Proposer une grammaire reconnaissant le même langage qui analyse le mot *bbb* sans retour-arrière (*backtrack*).
2. Ébaucher l'arbre d'exploration des solutions pour une analyse descendante pour la grammaire $S \rightarrow S + S \mid a \mid b$ et le mot reconnu $a + b$.
3. Est-ce qu'un grammaire régulière apporte un avantage par rapport à une grammaire algébrique quelconque du point de vue des algorithmes d'analyse vus en cours (ascendant et descendant) ?
4. Soit la définition suivante d'une formule logique (langage L_p) :
   (i) Si A est un nom de prédicat du vocabulaire de L_p, et chacun des $t_1 ... t_n$ une constante ou une variable du vocabulaire de L, alors $A(t_1, ..., t_n)$ est une formule.
   (ii) Si φ est une formule dans L, alors $\neg\varphi$ l'est aussi.
   (iii) Si φ et ψ sont des formules dans L, alors $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, et $(\varphi \leftrightarrow \psi)$ sont des formules de L.
   (iv) Si φ est une formule et x une variable, alors $\forall x\varphi$ et $\exists x\varphi$ sont des formules de L.
   (v) Rien d'autre n'est une formule

   (a) En fixant arbitrairement un vocabulaire **fini** pour L_p (noms de variables, noms de constantes et noms de prédicats), et en supposant pour simplifier que tous les prédicats sont unaires, proposer une grammaire hors-contexte (CFG) qui reconnaît ce langage.

   (b) Donner les arbres syntaxiques correspondants aux expressions
   $(P(x) \rightarrow Q(a))$ et $\forall x \exists y((F(x) \wedge A(y)) \rightarrow B(x))$

   (c) Supposons que l'on s'autorise à supprimer les parenthèses superflues dans les termes de la forme $((\alpha \wedge \beta) \wedge \gamma)$ (qui s'écrivent alors $(\alpha \wedge \beta \wedge \gamma)$). Proposer une grammaire hors-contexte *non ambiguë* qui reconnaît le même langage.

5. Une grammaire algébrique $G = \langle X, V, S, P \rangle$ est dite *simple* si G vérifie les deux conditions :
   – $P \subset V \times XV^*$
   – $\forall A \in V, \forall x \in X, \forall u, u' \in V^*, ((A \rightarrow xu) \wedge (A \rightarrow xu') \Rightarrow (u = u'))$
   Un langage algébrique est un *langage simple* s'il existe un grammaire simple qui l'engendre.

   (a) Trouver une grammaire simple pour le langage $\{a^n b^{n+1}, n \geq 0\}$

   (b) Trouver une grammaire simple pour le langage $\{a^n b^n, n > 0\}$

   (c) Soit L le langage engendré par : $S \longrightarrow aSS \mid b$. Construire une grammaire algébrique qui engendre le langage Lc^*d.

   (d) Montrer que la concaténation de deux langages simples est un langage simple. On demande une explication rigoureuse, pas nécessairement une démonstration mathématique.