1 Correspondance Automate à Pile / Grammaire hors-contexte

1. Soit la grammaire (déjà vue) sur l'alphabet $X = \{+, =, a\}$ engendrant $a^n + a^p = a^{n+p}$ avec $n > 0$ et $p > 0$.
   $S \rightarrow aSa$
   $S \rightarrow aBa$
   $B \rightarrow +X$
   $X \rightarrow aXa$
   $X \rightarrow a = a$
   Construire un automate à pile par l'algorithme de correspondance AP / grammaire hors-contexte.

2 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*, puis l'analyse ascendante.

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 $a + b$
   Détailler l'analyse ascendante.
3. Est-ce qu'une 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) ? Quelle forme particulière de grammaire algébrique pourrait avoir le même intérêt ?
4. On rappelle l'algorithme naïf d'analyse descendante en profondeur d'abord. Appliquer l'algorithme avec la grammaire de l'exo 1 et la reconnaissance du mot *bbb*. Quel problème constatez-vous ? Proposer un algorithme qui résoud ce problème, en utilisant la notion de préfixe terminal (on définit le préfixe terminal d'une chaîne comme le préfixe le plus long possible constitué uniquement de terminaux.). D'après les réponses à l'exercice précédent, modifier encore l'algorihme, modulo hypothèses sur la grammaire.

FIG. 1 – Algorithme descendant le plus naïf (et qui d'ailleurs ne marche pas)

```
// la fonction principale est tdlrp : top-down left-right parsing
// α est la protophrase courante, u l'entrée à reconnaître
tdlrp(u, v)
begin
    if (u == v) then return true
    u = u_1 u_2 ... u_k A γ // A est le premier non terminal de u
    foreach règle de la forme A → β do
        if tdlrp(u_1 u_2 ... u_k β γ, v)=true then return true
    return false
end.
```