TP3 - ALGORITHME DE CYK

1. Langages formels - TP 3

1.1. **Objectif du TP.** L'objectif de ce troisième TP est d'implémenter un programme avec deux arguments : un fichier `.txt` contenant la description d'une grammaire G sous forme normale de Chomsky et un mot w à générer. Votre programme doit :

[1] lire/charger la grammaire G à partir du fichier `.txt`,

[2] executer l'algorithme de CYK et afficher YES si le mot est généré par la grammaire, ou NO si le mot n'est pas généré par G.

Si vous avez fini d'implémenter l'algorithme CYK vous pouvez de plus utiliser le TP2 pour réaliser le programme qui fait la chose suivante :

[1] lire/charger la grammaire G à partir du fichier `.txt`,

[2] tester si G est sous forme normale de Chomsky et la mettre sous forme normale si ce n'est pas le cas,

[3] executer l'algorithme de CYK et afficher YES si le mot est généré par la grammaire, ou NO si le mot n'est pas généré par G.

Dépôt de fichiers : Vous pouvez travailler seul ou en binôme, dans ce cas pensez à bien préciser le nom des deux étudiants. Les rendus des 3 TP sont à déposer sur Amétice. Seul le rendu du TP3, qui contiendra toutes les fonctions implémentées au cours des 3 séances, sera noté.

2. Travail à effectuer

Tableau des facteurs. Vous devez implémenter l'algorithme CYK vu en cours. Pour cela il vous faut construire un tableau qui garde en mémoire pour chaque facteur u du mot d'entrée w la liste des variables pouvant générer u. Un façon d'implémenter un tableau en python est d'utiliser une liste de listes (dans ce cas donc une liste de listes de listes). Il vous faut initialiser votre tableau pour que chaque cellule du tableau contienne la liste vide.

Parcours des facteurs. Notez que si vous faites un parcours "naif" du type :

```
for i in range(n):
  for j in range(i,n+1):
    w[i:j]...
```

alors vous ne parcourez pas les facteurs par taille croissante. Pour faire un parcours de ce type il vaut mieux faire une première boucle qui parcours les longeurs des facteurs et une seconde qui parcours le point de départ du facteur :

```
for l in range(n):
  for i in range(n-l):
    w[i:i+l+1]...
```

Couper un facteur en deux et règles de la grammaire. Ensuite il vous faut parcourir toutes les façons de couper un facteur donné $w[i{:}i+l+1]$ en deux parties (non vides). Enfin, pour chaque façon de couper un facteur en deux, $w[i{:}i+k], w[i+k{:}i+l+1]$ il faut parcourir toutes les règles $X \to YZ$ de la grammaire et si Y et Z génèrent respectivement $w[i{:}i+k]$ et $w[i+k{:}i+l+1]$ alors on ajoute X à la cellule du tableau correspond au facteur $w[i{:}i+l+1]$.

Résultat. Une fois le tableau rempli, il suffit de vérifier si l'axiome de la grammaire peut générer le mot $w[0:n]$.

Fichiers tests. Vous devez non seulement implémenter la fonction mais aussi la tester. Pour cela, écrivez 3 grammaires sous forme normale de chomsky au format textuel pour étendre la base de tests au délà des fichiers fournis. Vous pouvez utiliser des grammaires vues en cours ou en TD. Ces grammaires doivent être placées dans le dossier test. Pour déboguer votre programme, n'oubliez pas que vous pouvez à tout moment appeler `print(G)` pour afficher la grammaire sur le terminal. En python, vous pouvez/devez aussi utiliser la bibliothèque pdb pour le débogage, et vous pouvez aussi typer votre programme et vérifier les types à l'aide de mypy.