**Introduction :**

**Cplex :**

**Représentation des agents et règles de transition :**

Après m’être approprié le sujet, la première difficulté a été pour moi de comprendre comment représenter les données dans cplex ; alors que nous avions des agents et les règles de transitions, il s’est avéré que cette mise en donnée était plus simple en utilisant deux matrices : une représentant les correspondances entre les agents, et l’autre représentant les correspondances entre les règles de transition de nos deux environnements. Ainsi, le lien entre les deux se fait dans le calcul de notre fonction d’évaluation, et nous trouvons en sortie toutes les correspondances entre les deux environnements.

**Contraintes non linéaires dans la fonction d’évaluation :**

Une des difficultés rencontrées a été la //multiplication de variables nécessaire au calcul de notre fonction d’évaluation. En effet cplex ne calcule pas les contraintes non linéaires. Pour pallier à ce problème, nous avons utilisé une astuce permettant indirectement d’avoir une contrainte contenant une multiplication de deux cases de tableaux dvar : sachant que nos valeurs sont des booléens, si Z==X\*Y n’est pas une contrainte calculable par cplex, Z<=X, Z<=Y, Z>=X+Y-1 nous renverra le même résultat pour Z et sera accepté par cplex. Ainsi, en développant notre calcul de base, nous obtenons un résultat identique et juste.

**Exemples travaillés sur cplex :**

Pour comprendre comment automatiser la comparaison de deux environnements, nous avons commencé par traduire deux exemples en code cplex, le premier, assez simple :

**Environnement 1 :**

Agents : A, B

Règles : A+ ==> B+ A+ ==> A-

**Environnement 2 :**

Agents : C, D

Règles : C- ==> D+ D- ==> D+

Lors de la comparaison de ces deux environnements sur cplex, on obtient une correspondance entre des couples d’agents (A+, C-), (A-, C+), (B+, D+) et (B-, D-), et des règles (A+ ==> B+, C- ==> D+) et (A+ ==> A-, D- ==> D+).

Cette correspondance est bien celle qui nous donne le meilleur résultat pour notre fonction d’évaluation S, on se demande alors comment étendre notre modèle à des cas aux règles plus complexes, nous essayons donc un deuxième exemple :

**Environnement 1 :**

Agents : A, B, C

Règles : A+, B- ==> B+ C- ==> A+

**Environnement 2 :**

Agents : D, E

Règles : D- ==> E+ E+, E- ==> E+

La taille des contraintes nous pose alors le problème de stockage de nos calculs de contraintes non linéaire. La soluation a alors été la création d’un tableau pour chaque case de notre matrice de contrainte, rendant ainsi notre exemple plus proche de ce que l’on voudra obtenir avec notre traducteur.

**Notre programme :**

**Récupérer nos fichiers environnements :**

Dans un premier temps, notre programme devra donc récupérer nos environnements et les stocker en mémoire

L’une des premières difficultés rencontrées lors de l’écriture de notre programme a été le stockage de nos agents et règles. En effet, il nous fallait un moyen pour faire le lien entre les différents agents d’une même règle de transition pour le calcul de notre fonction d’évaluation. Nous avons donc créé des structure adaptée d’environnement, d’agents et de règles. Les règles contiennent l’adresse des agents