# Introduction

Aie aie aie.

# Définitions

Dans cette section, nous nous intéressons donc à la définition d’un écosystème, ainsi que de ces composantes.

## Ecosystème

Un écosystème est donc un ensemble d’agents qui peuvent être présents ou non.

Nous partons du principe qu’aucun agent ne peut être présent et absent à la fois.

La présence d’un agent, appelée polarité, est dénoté par a+ si l’agent a est présent, et a- sinon, celle-ci dépend des règles de réécritures R.

Plus formellement, un écosystème Ɛ est un tuple (A, R) tel que :

* A est un ensemble d’agents avec polarité tel que :

Ap = {a+, a- | a є A}

* R est un ensemble de règles de réécriture sous la forme :

r : α+, α- >> ω+, ω-

Où r est le nom de la règle, α+ et ω+ sont les agents présents et α- et ω-sont les agents absents.

On appellera lhs(r) (respectivement rhs(r)) les agents présents à gauche (respectivement à droite) d’une règle de réécriture r.

L’état d’un écosystème est défini par la présence ou non de tous ses agents. On le décrit donc par la liste de tous les agents présents, et on considère les autres comme absents.

La dynamique d’un écosystème (A, R) dépend de son état initial s0 et contient tous les états atteignables en appliquant les règles de R de façon non déterministe.  
Une règle r est applicable à un état s si les agents présents dans le côté gauche de la règle sont présents dans notre état s. En appliquant cette règle, on obtient alors un nouvel état s’ tel que :

s’= (s \ ω-) ᴜ ω+

## Exemple

Voici donc un exemple d’écosystème composé d’agents et de règles de réécriture :

**Agents :**

B : Oiseaux

I : Insectes

Pe : Pesticides

P : Pluie

**Règles :**

B+ => I- Les oiseaux mangent les insectes

Pe-, R+ => I+ La pluie et l’absence de pesticides laissent les insectes se développer

## Similarités entre deux écosystèmes

On s’intéresse donc dans cette section aux points communs repérables deux écosystèmes. Intuitivement, on pourrait exprimer ces similarités en comptant le nombre d’agents ayant des rôles identiques dans les deux écosystèmes.

Cela signifie que si on avait une carte de correspondance entre les agents de deux écosystèmes, et leurs règles, le « score » de similarité qu’on pourrait avoir serait défini par le nombre de correspondance entre agents et le nombre d’agents que deux règles auraient en communs.

 Plus formellement, soit Ɛ1 = (A1, R1) et Ɛ2 = (A2, R2) deux écosystèmes, et µ et ρ les deux cartes de correspondances, respectivement entre agents et règles, la première est :

µ : A1 A2

Cela signifie que si l’agent a+ є A1 pourrait être associé ou à b+, ou à c- sachant que b+,c- є A2.

Nous devons donc faire le lien entre les agents des deux environnements, puis entre les règles. Nous utilisons pour ça deux matrices booléennes, la première ; X représentant les correspondances entre agents, et la seconde Y représentant les correspondances entre règles.

Par exemple, si l’on avait deux environnements :

**Environnement 1 :**

Agents : A, B

Règles : A+ => B+ A+ => A-

**Environnement 2 :**

Agents : C, D

Règles : C- => D+ D- => D+

La matrice X sera sous la forme :

A+ A- B+ B-

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

C+

C-

D+

D-

Ici, si la case [A+][C+] vaut 1, c’est que notre programme a trouvé que la comparaison entre nos environnement la plus optimale nous donne une correspondance entre les agents A+ et C+.

La matrice Y sera :

A+ => B+ A+ => A-

|  |  |
| --- | --- |
|  |  |
|  |  |

C- => D+

D- => D+

Si la première case vaut 1, c’est qu’il y a correspondance entre les règles A+ => B+ et C- => D+.

Les contraintes à mettre en place pour rendre les correspondances injectives sont alors assez simples ;

-Dans les deux cas on ne doit avoir qu’un « 1 » par ligne et par colonne, un agent ou une règle

# Mon travail

Le but a donc été pour moi dans un premier temps de créer des modèles cplex qui me permettraient d’obtenir en sortie les deux matrices de correspondances entre les agents et règles de deux écosystèmes, j’ai pour cela fait deux exemples, le premier assez simple, puis un second un peu plus complexe.

## Un premier exemple

Ce premier exemple est donc simplement composé de deux écosystèmes, ayant chacun deux agents et deux règles de réécriture.

Environnement 1 :

Agents : A, B

Règles : A+ ==> B+ A+ ==> A-

Environnement 2 :

Agents : C, D

Règles : C- ==> D+ D- ==> D+

Pour comparer ces deux écosystèmes dans cplex, les contraintes à mettre en place correspondaient à celles vu dans la partie définitions, et ont donc été assez simples à écrire, à l’exception de notre fonction d’évaluation.

En effet, ici par exemple,

S == Y[0][0] \* (2\*X[1][0] - 1 + 2\*X[2][2] - 1)

+ Y[0][1] \* (2\*X[1][0] - 1 + 2\*X[2][1] - 1)

+ Y[1][0] \* (2\*X[3][0] - 1 + 2\*X[2][2] - 1)

+ Y[1][1] \* (2\*X[3][0] - 1 + 2\*X[2][1] - 1);

## 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.

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 :

## Un exemple plus complexe

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 solution 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

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 structures adaptées d’environnement, d’agents et de règles. Les règles contiennent l’adresse des agents

# Analyse des résultats

Une fois notre programme terminé, nous avons donc fait de nombreuses expériences pour en tester les limites et trouver des cas d’applications.

## Notre exemple

Le premier test que nous avons fait sur notre programme a été de reprendre le 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+

En y exécutant notre programme, le modèle cplex obtenu a bien été identique à celui que nous avions fait à la main.

## Le pattern proie prédateur

Le pattern proie prédateur est

## Les limites

# Conclusion