Observation Table
Implémentation de l'algorithme de Dana Angluin

L'algorithme LStar de Dana Angluin :

```
Initialize S and E to {λ}.
Ask membership queries for λ and each a ∈ A.
Construct the initial observation table (S, E, T)
Repeat:
    While (S, E, T) is not closed or not consistent:
       If (S, E, T) is not consistent,
           then find s1 and s2 in S, a ∈ A, and e ∈ E such that
           row(s1) = row(s2) and T(s1· a · e) ≠ T(a2 · a · e),
           add a · e to E,
           and extend T to (S U S · A) · E using membership queries.
       If (S, E, T) is not closed.
           then find s1 ∈ S and a ∈ A such that
           row(s1 · a) is different from row(s) for all s ∈ S,
           add s1 · a to S,
           and extend T to (S ∪ S · A) · E using membership queries.

       Once (S, E, T) is closed and consistent. let M = M(S, E, T).
       Make the conjecture M.
       If the Teacher replies with a counter-example t, then
           add t and all its prefixes to S
           and extend T to (S ∪ S · A) · E using membership queries.
Until the Teacher replies yes to the conjecture M.
Halt and output M.
```

L'algorithme est basé sur les réponses d'un oracle, celui-ci répond à la présence ou non d'un mot dans l'ensemble des mots, et après répond oui ou non si l'automate correspond à l'alphabet, permettant l'arrêt de l'algorithme.

LStar construit une table d'observation :

T	**Λ**
Λ	1
0	0
1	0
00	1
01	0

Celle-ci est composé de trois parties, la première ligne *E* correspond aux préfixes du mot, la partie du haut *S*, correspond aux suffixes et enfin la dernière partie de la table d'observation, aux suffixes concaténés à l'alphabet, $S \cdot A$.

Enfin, il vérifie trois conditions pour créer un automate :

- La table d'observations doit être complète, il ne doit pas y avoir de vide.
- La table d'observation est consistante.
- La table d'observation est fermée.

Le programme

L'algorithme est fait avec le langage python et le principe de la programmation objet.

Il est composé de 5 classes et script qui permet lancer LStar :

- *ObservationTable* : Cette classe permet de stocker une table d'observation, de vérifier les conditions et de faire les modifications nécessaires. Il y a quatre parties importantes :
    - *AddCounterExemple* et *CompleteObservationTable* permettent d'interagir avec l'oracle. Ils permettent respectivement de rajouter un contre-exemple et de remplir la table d'observation.
    - *makeClosed* permet de vérifier si la table est fermé, puis, dans le cas contraire de résoudre la fermeture.
    - *makeConsistent* permet la consistance de la table, puis, dans le cas contraire de résoudre la consistance.
    - Enfin, *makeAutomata* permet de créer un automate de Melay à partir de la table d'observation.

ObservationTable repose sur plusieurs autres classes pour représenter la table d'observation :

- *Table* permet de représenter la seconde ou la troisième partie de la table.
- *Row* représente une ligne de la table d'observation et est utilisé dans *Table*.
- *ListKey* permet de contenir tous les mots de la table d'observation et facilite la recherche sur une clé.
- *Automata* permet de représenter un tableau pour contenant les états et les transitions entre états (Automate de Melay). Permet aussi, avec la fonction *drawAutomataGraphviz* de créer un fichier .gv permettant de dessiner un automate avec GraphViz.

Enfin, le fichier LStar contient l'algorithme de LStar et permet donc de créer un automate.

Il y a trois sorties au programme. La première sur la console permet d'avoir le déroulement de LStar et permet de répondre en tant qu'Oracle. Les deux autres sont des fichiers :

- saveInput.txt permet de garder une trace des entrées faites par l'Oracle et permet donc de refaire une exécution identique.
- Automata.gv est créé avec la fonction *drawAutomataGraphviz* de *Automata* et permet donc de dessiner un automate de Melay et d'avoir une représentation graphique.

GraphViz est un logiciel sous linux (graphviz.org) qui permet de dessiner des automates. Il faut un fichier .gv (automata.gv) et lancer cette commande : *dot -Tpng automata.gv -o nameFile.png*.

Déroulement de LStar

Je vais présenter les différentes parties qu'il y a dans le programme, pour l'exemple, on cherche l'automate qui reconnait les mots ayant un nombre pair de a et/ou b.

Pour indiquer l'alphabet du langage, il faut modifier la variable *A* dans LStar.

```
Symbol aa is present in this automate? (Yes:1 / No: 0) 1
Symbol ab is present in this automate? (Yes:1 / No: 0) 0
```

Tout d'abord une requête pour savoir si le mot est présent est reconnue par l'automate. Il suffit simplement d'indiquer *oui* ou *non*.

```
T -> ['']
-----
: [1]
a : [0]
-----
b : [0]
aa : [1]
ab : [0]

    ['a', 'b']
-----
0 : [1, 1]
1 : [0, 1]
-----

This is the automate that you want ? (Yes: 1 / No: 0) 0
What is the counter-example ? (A : ['a', 'b']) bb
Symbol bb is present in this automate? (Yes:1 / No: 0) 1
```

Le programme fait des requêtes jusqu'à ce que la table d'observation remplisse les trois conditions, complète, fermée et consistante. Si c'est le cas, il affiche la table d'observation, l'automate et demande si il correspond aux attentes. Si non, un contre-exemple contenant uniquement l'alphabet est demandé. Puis LStar recommence les requêtes jusqu'à remplir de nouveaux les conditions.

```
Final Observation table :
T -> ['', 'a', 'b']
-----
: [1, 0, 0]
a : [0, 1, 0]
b : [0, 0, 1]
bb : [1, 0, 0]
ab : [0, 0, 0]
abb : [0, 1, 0]
-----
aa : [1, 0, 0]
ba : [0, 0, 0]
bba : [0, 1, 0]
bbb : [0, 0, 1]
aba : [0, 0, 1]
abba : [1, 0, 0]
abbb : [0, 0, 0]

----- ----- ----- ----- ----- -----
Final Automata :
    ['a', 'b']
-----
0 : [1, 2]
1 : [0, 3]
2 : [3, 0]
3 : [2, 1]
-----
```

Si oui, Lstar se termine et affiche la table d'observation finale et l'automate finale. De plus un fichier automata.gv est créé.

Le programme est fourni avec deux exemples pour l'exécution de LStar. Ceux-ci peuvent être données comme redirection d'entrée au lancement du programme :

- Le premier *testpairab.txt* est l'exemple utiliser par la plupart des articles et cours sur internet, le langage accepte les mots ayant un nombre pair de a et/ou b. L'alphabet n'est composé que des lettres a et b, et du mot vides.
- Le deuxième *testfinabcd.txt* accepte tous les mots se terminant par *abcd*. L'alphabet est composé des lettres a, b, c et d, ainsi que du mot vide.

Continuation du projet

Il y a deux pistes pour continuer sur le projet.

La première consiste à modifier l'algorithme pour que, au lieu d'indiquer si une trace est correcte pou non, indiquer sa catégorie. Mais cela demande une modification de l'algorithme, tout d'abord redéfinir la consistance et la fermeture de la table d'observation, ensuite déterminer ce que l'automate et redéfinir la création de l'automate.

La seconde consiste à enlever l'Oracle, et l'automate se créé seulement à partir des traces. Il y a donc deux parties à définir : savoir si l'automate correspond au langage et dans le cas contraire fournir un contre-exemple et gérer les requêtes pour remplir la table d'observation.

Dana Angluin propose une méthode statistique sans avoir besoin de l'Oracle. Celle-ci est définit dans la partie 4 de son article : FINDING COUNTEREXAMPLE USING SAMPLING.

Pour remplacer les requêtes, le programme pourrait s'appuyer sur un fichier d'apprentissage contenant un ensemble de mots avec chacun une indication si ils sont présent ou non dans le langage.

Bibliographie

Angluin, D. (1987). Learning Regular Sets from Queries and Counterexamples. *INFORMATION AND COMPUTATION*(75), pp. 87-106.

D'Souza, D. (n.d.). *Automata Theory and Computability.* Retrieved from Deepak D'Souza: http://drona.csa.iisc.ernet.in/~deepakd/atc-2015/L_Star_Algo.pdf

De La Higuera, C. (2009). Learning with Queries. In *Grammatical Inference* (p. 215). Retrieved from Université de Nantes: http://pagesperso.lina.univ-nantes.fr/~cdlh//book/Learning_with_Queries.pdf

De La Higuera, C. (2010). Retrieved from Université de Nantes: http://pagesperso.lina.univ-nantes.fr/~cdlh/Downloads/Active_Zadar.pdf

JAFAROV, O., & GASIMOV, S. (2012-2013). *Rapport de projet "Machine Learning".* Retrieved from https://dept-info.univ-fcomte.fr/joomla/images/CR0700/Projets/2013/M2/RapportProjet_%20Jafarov_Gasimov.pdf