Projet

Un éditeur textuel ou graphique pour les automates d'états finis (AEF)

ING1 GMA & GIA

Notion d'automates d'états finis

Un **automate fini** ou **automate avec un nombre fini d'états** (en anglais *finite-state automaton* ou *finite state machine* ou *FSM*) est un modèle mathématique de calcul, utilisé, par exemple, pour la conception de programmes informatiques ou de protocoles de communication. Dans la Figure 0, une modélisation du fonctionnement d'un portillon est donnée.

Exemple 1 :

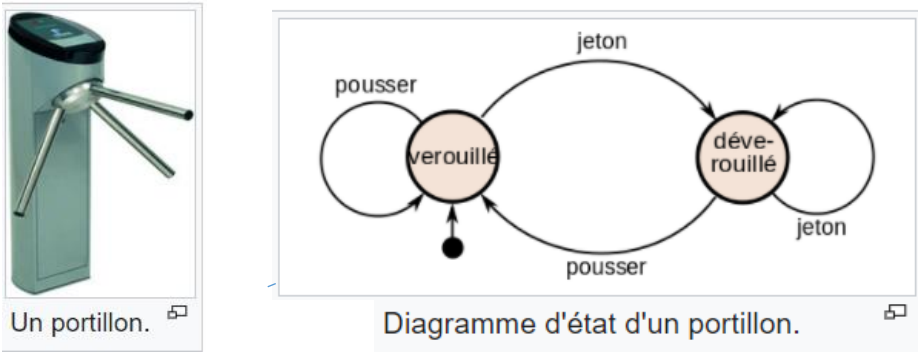

Figure 0 : Modélisation d'un portillon.

Travail demandé

Il vous est demandé de réaliser un ***éditeur textuel*** dédié aux automates d'états finis (AEF). Votre éditeur doit permettre de :

1. Manipuler un AEF :
   - Saisir un AEF.
   - Importer un AEF à partir d'un fichier.
   - Modifier un AEF.
   - Sauvegarder un AEF dans un fichier.
   - Supprimer un AEF.

2. Vérifier si un mot est reconnu par un AEF.

   Par exemple, l'AEF donné dans la Figure 1 reconnait n'importe quel mot commençant par une série de a, pouvant être vide, et se terminant obligatoirement par *b.*

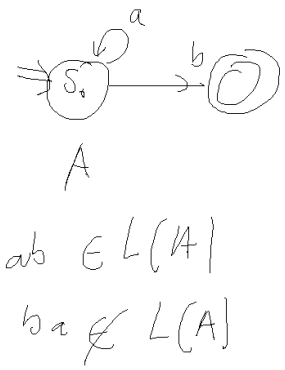

Figure 1 : Vérification de l'appartenance de mots au langage reconnu par un automate.

3. Vérifier si un automate est complet.

   Dans un AEF complet, chaque état possède une transition pour chaque symbole de son alphabet à partir de cet état. Dans la partie gauche de la Figure 2, nous avons rajouté une étoile verte à l'unique état complet de l'automate, c'est-à-dire s_0, et des étolies rouges aux états non complets, à savoir s_1 et s_2.

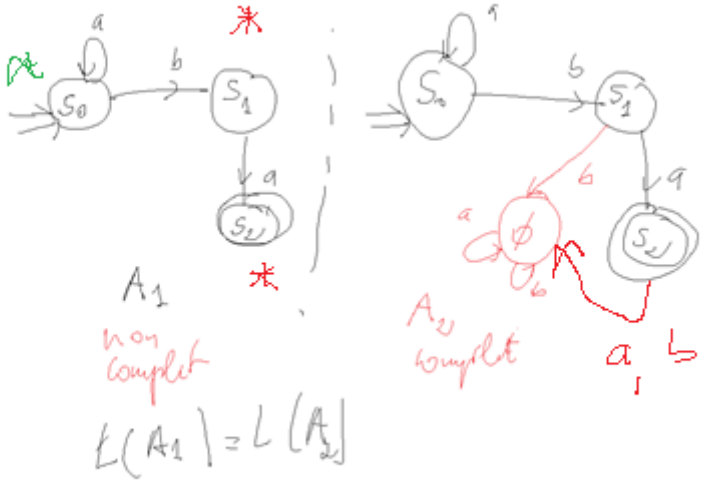

Figure 2 : Un automate non complet et un second complet.

4. Rendre un automate complet.

   Pour rendre un automate complet, dans le cas où il ne l'est pas, un état *phi* est rajouté. Celui-ci est non final et boucle autour de chaque symbole de l'alphabet de l'automate. Pour tout état *s* non complet de l'automate et chaque symbole *a* où aucune transition n'est possible, rajouter une transition de *s* vers ***phi*** par *a* (voir l'automate de droite de la Figure 2).

5. Vérifier si un automate est déterministe.

   La définition ainsi que la procédure de transformation d'un automate non déterministe en un autre déterministe sont données dans le cours des langages réguliers. Par exemple, les automates donnés dans la Figure 2 sont déterministes alors que celui donné dans la Figure 3 ne l'est pas.

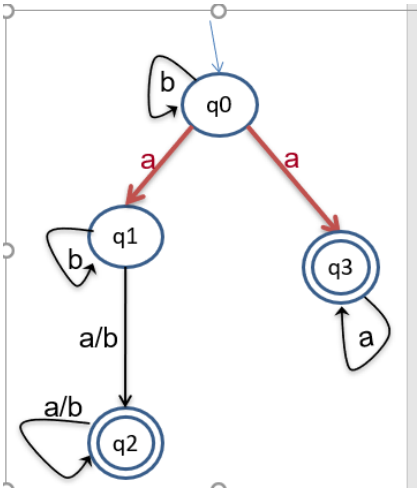

Figure 3 : Un automate non déterministe.

6. Rendre un AEF déterministe (voir le cours les langages réguliers).

7. Réaliser les opérations suivantes sur les AEFs :

   - *Complément d'un AEF* : Le complément d'un AEF est un nouvel AEF obtenu en inversant les états finaux et non finaux de l'AEF d'origine. Si un état était final dans l'AEF d'origine, il devient non final dans le complément, et vice versa.

   - *Miroir d'un AEF* : Le miroir d'un AEF est un nouvel AEF obtenu en inversant la direction des transitions de l'AEF d'origine. Autrement dit, si l'AEF d'origine avait une transition de l'état A vers l'état B avec un symbole donné, le miroir aura une transition de l'état B vers l'état A avec le même symbole. Les états finaux (resp. initiaux) sont initiaux (resp. finaux) dans le nouvel AEF.

- Produit de deux AEFs : Le produit de deux AEFs *A* et *B*, noté *A* * *B*, est un nouvel AEF dont les états sont toutes les combinaisons d'états possibles de *A* et *B*. Les états finaux du produit sont ceux pour lesquels les états correspondants de *A* et *B* sont tous deux finaux. Les transitions sont définies de manière similaire, où les transitions du produit relient les états en fonction des transitions des AEFs d'origine.

- Concaténation de deux AEFs : La concaténation de deux AEFs *A* et *B*, notée *A.B*, est un nouvel AEF où les états et transitions de *A* sont suivis immédiatement par les états et transitions de *B*. Les états finaux de *A.B* sont les états finaux de *B,* et l'état initial de *A.B* est l'état initial de *A*.Pour plus d'informations, vous pouvez consulter le site suivant : https://www.lirmm.fr/~lafourcade/ML-enseign/Cours%20Langage/Cours%20Langages%20ch2%20(d14).htm

8. Extraire une expression régulière à partir un automate donné. Par exemple, *a*b* est une expression régulière que l'on peut associer à chacun des deux automates de la Figure 2.

9. Trouver le langage reconnu par un automate donné. Par exemple, {*a*b*} est le langage reconnu par chacun des automates de la Figure 2.

10. Vérifier si deux automates sont équivalents, i.e. ils reconnaissent les mêmes langages.

11. Rendre un automate émondé (voir le cours).

12. Rendre un automate minimal. i.e. le nouvel automate reconnaît le même langage que l'automate original et a le nombre minimal d'états possible tout en respectant la première condition. Cela signifie qu'aucun état ne peut être supprimé de l'automate sans changer le langage qu'il reconnaît.

Travail demandé

- Programmation en C (prépa externe) et en python (prépa interne)
- Interface en mode console
- Toutes les fonctionnalités doivent être implémentées dans le projet en Python, et autant que possible dans le projet en C.
- Des tests unitaires automatiques.
- Une documentation en anglais de votre application, complètement ″English″.
- Votre projet entier doit être sauvegardé sur github.