LOG 2810
STRUCTURES DISCRETES

Rapport de laboratoire #2
TP2 : Automates

Éléments évalués	Points
Qualité du rapport : respect des exigences du rapport, qualité de la présentation des solutions	2
Qualité du programme : compilation, structures de données, gestion adéquate des variables et de toute ressource (création, utilisation, libération), passage d'arguments, gestion des erreurs, documentation du code, etc.	3
Composants implémentés : respect des requis, logique de développement, etc.	
C1	3
C2	3
C4	2
C5	3
C6	2
Total de points	20

Soumis par :

Bouchard, Billy - 1850477

Gnaga, Dogbeda Georges - 1870143

Obossou Ema-Wo, Sanyan - 1780896

Le 12 Avril 2018

Table des matières

Introduction

Dans l'optique de mettre en application les notions théoriques sur les automates apprises durant le cours de structures discrètes, nous sommes amenées à

créer une application qui grâce à des automates et banques de mots génère des mots de passe.

Un étudiant cambrioleur a réussi à mettre la main sur la liste des mots de passe des coffres-forts de ses banques cibles mais malheureusement ces mots de passe ont été corrompus lors du transfert du fichier, notre objectif est d'aider cet étudiant à retrouver les mots de passe afin qu'il puisse continuer sa mission.

Pour ce faire nous avons à partir de quelques fichiers contenant des règles linguistiques, de fichiers contenant des variantes correspondantes à ces règles et surtout grâces à des automates qui reconnaissent tous les mots dont ces règles définissent, trouver les mots de passes pour accéder à ces coffres-forts.

1. Présentation

Après tout échange avec l'étudiant, nous nous sommes procurés les fichiers nécessaires pour désamorcer son problème. Nous disposons de 10 fichiers, 5 fichiers contiennent les règles linguistiques utilisées pour créer les mots de passe et 5 autres fichiers contenant les variantes de chaque mot de passe potentiellement corrompu. Afin de d'aider l'étudiant nous avons développé une application. L'application permet

- De créer des automates qui suivent chacun les règles contenues dans les 5 premiers fichiers.
- Une fonction qui lit le fichier texte contenant les variantes de mots de passe à tester et le numéro de l'automate nécessaire.
- Une fonction qui passe les variantes de mots de passe à l'automate responsable et procède au test de validation.
- Et enfin une fonction qui affiche les mots de passe trouvés.

Afin de facilité la compréhension de l'application nous avons mis en place un menu qui indique clairement à l'étudiant la procédure à suivre pour retrouver ces mots de passe.

Pour coder notre application, nous avions le choix entre trois langages de programmation à savoir le C++, le Java et le Python. Nous avons choisi Python car, comparé aux deux autres, il est plus simple syntaxiquement parlant, et surtout parce que c'est un langage que nous ne maitrisions pas parfaitement et nous avons trouvé en cette application une très bonne occasion de se lancer dans son apprentissage.

Après l'étude de notre problématique, nous avons élaboré un diagramme de classe (voir figure 1) qui résume toute l'application, laquelle est séparée en trois fichiers.

Fichier automate.py

Ce fichier contient

- La classe Automate qui définit un automate qui suit une règle linguistique donnée. Elle est composée des méthodes suivantes :

  - __init__(self, name) : initialise l'attribut du nom de fichier et le tableau qui contiendra les différentes règles.

  - __lshift__(self, regle) : permet la transition entre les états de l'automate

  - __contains__(self, item) : vérifie la présence de la règle en paramètre dans le tableau des règles

  - __getitem__(self, key) : retourne une règle contenue dans le tableau des règles

  - solve(self, string) : définit le principes des états de l'automate

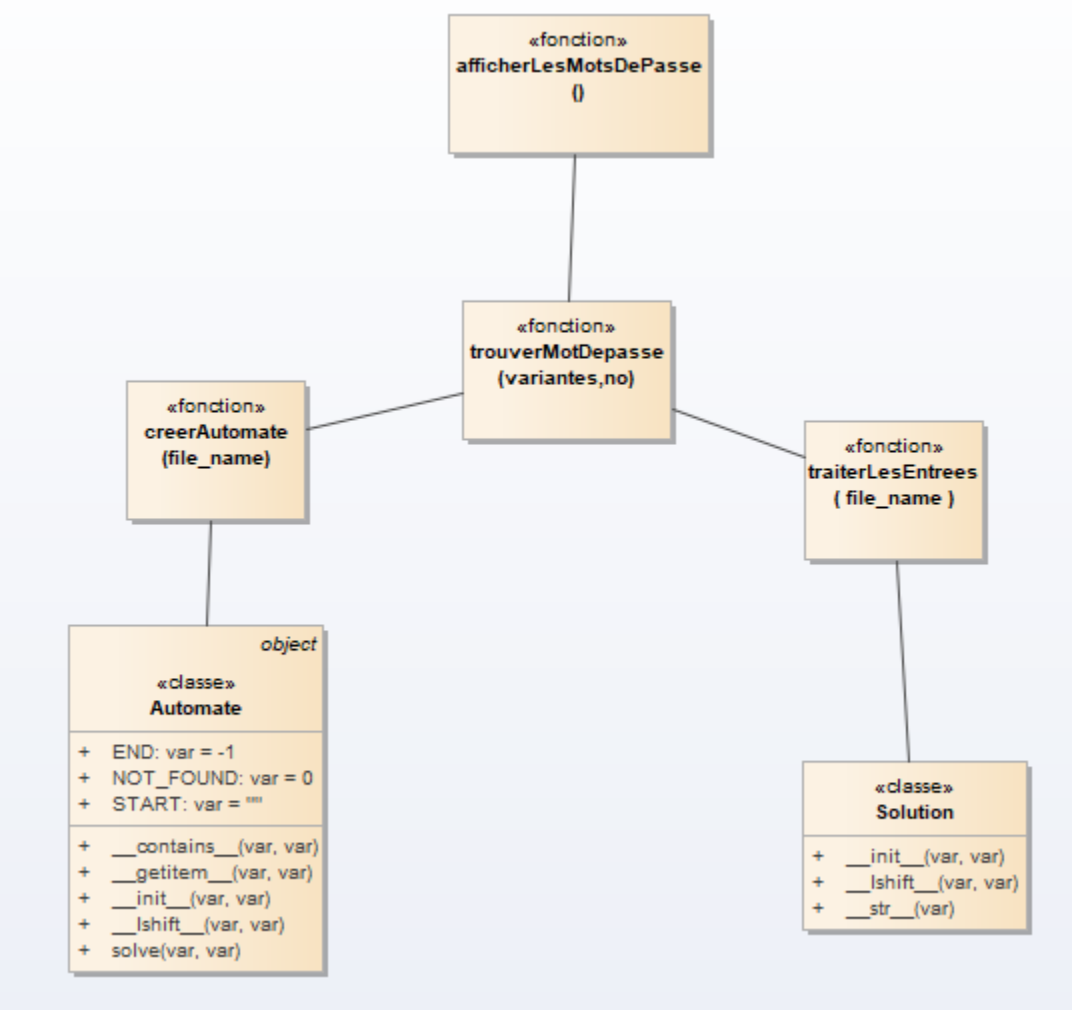

Figure 1 : Diagramme de classe complet de l'application.

1.1. Fichier solution.py

Ce fichier contient

- La classe Solution :

  Elle est composée des méthodes suivantes :

- __init__(self, no) : initialise un tableau qui contiendra les variantes de mots de passes et quelques attributs

- __lshift__(self, item) : ajoute des variantes de mots de passe au tableau de variantes

- __str__(self) : retourne le mot de passe trouver ou un message sinon

1.2. Fichier main.py

Ce fichier contient

- Les principales fonctions de l'application. Il est composée des fonctions suivantes :

- creerAutomate(file_name) : créé un automate qui sera responsable de valider les mots de passe à partir d'un fichier de règle.

- trouverMotDePasse(variantes, no) : permet de trouver et de valider les mots de passe parmi les variantes proposées.

- traiterLesEntrees(file_name) : lit un fichier texte contenant un numéro d'automate et des variantes de mots de passe à tester.

- afficherLesMotsDePasse() : indique les mots de passe validés et le numéro de l'automate associé

2. Difficultés rencontrées

Voici les difficultés auxquelles nous nous sommes confrontés :

- Recherche régulière des diverses syntaxes pour écrire nos algorithmes.
- Débogage du code un peu plus complexe vu la maîtrise partielle du langage.
- Difficulté pour écrire l'algorithme qui implémente les automates.

3. Solutions aux difficultés rencontrées

Nous nous sommes imposé une bonne discipline dans la réalisation de notre application, et nous avons donc adopté des méthodes telles que :

- Visionnement de plusieurs tutoriels sur le fonctionnement des automates.
- Révision des notions théoriques sur les automates vus en classe.
- L'entraide collective dans notre équipe nous a permis de facilement repérer nos erreurs et de les corrigés.
- Tests unitaires réguliers et continu des méthodes et fonctions durant les implantations.

Conclusion

En résumé ce travail pratique nous a permis de mettre en pratique les notions théoriques sur les automates, cela nous aussi d'appliquer les connaissances apprises dans d'autre cours comme conception logiciels (LOG2410), ingénierie logiciel (LOG1000), les enseignements de HPR qui ont été très utiles pour une bonne organisation interne qui nous a permis de réussir ce travail pratique.

Contrairement à l'ancien TP nous avons passé moins de temps sur celui-ci. Nous ne maitrisions pas totalement python mais grâce aux connaissance acquise durant le TP1 nous avons pu vite évoluer et venir à bout de ce TP dans la contrainte de temps fixée. Nous espérons pour les étudiants de la session prochaine un temps de travail similaire et bien sûr un projet aussi nourrissant et constructif comme l'a été le notre.