POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

LOG 2810
STRUCTURES DISCRETES

TP2 : AUTOMATES

Session	Hiver 2018
Pondération	10 % de la note finale
Taille des équipes	3 personnes
Date de remise du projet	17 avril 2018 (23h55 au plus tard)
Directives particulières	Soumission du livrable par moodle uniquement (https ://moodle.polymtl.ca).
	Toute soumission du livrable en retard est pénalisée à raison de 10% par jour de retard.
	Programmation C++, Python ou Java
Les questions sont les bienvenues et peuvent être envoyées à: Loïc Lerat (loic.lerat@polymtl.ca), Juliette Tibayrenc (juliette.tibayrenc@polymtl.ca).	

1 Connaissances requises

- Notions d'algorithmique et de programmation C++, Python ou Java.
- Notions sur les automates.
- Notions sur les structures de données.

2 Objectif

L'objectif de ce travail pratique est de vous permettre d'appliquer les notions théoriques sur les automates que nous avons vues en cours, sur des cas concrets mais hypothétiques. À cet effet, il est question dans ce travail de déterminer les mots de passe pour pouvoir ouvrir les coffres-forts des banques ciblées par une série de cambriolages.

3 Mise en situation

Le même étudiant que vous connaissez bien du TP précédent est toujours en train de planifier ses cambriolages. Il est cette fois-ci en train d'étudier la sécurité des coffres-forts des banques cibles. Il a réussi à obtenir une liste de mots de passe pour différentes banques, mais les mots de passe ont peut-être été corrompus pendant le transfert des fichiers. Il a aussi découvert que le générateur aléatoire de la bibliothèque utilisée par ces banques n'est pas réellement aléatoire : tous les mots de passe générés font partie de langages dont il a pu déterminer les règles de production. Il lui suffit donc d'essayer

différentes variantes de chaque mot de passe potentiellement corrompu et d'utiliser celle qui est validée par l'automate correspondant à l'un de ces langages le jour du casse. Il sait que les notions du cours de LOG2810 et tout spécialement le module de la théorie des langages seraient très pertinents à sa préparation, mais il est débordé de travail et a donc encore besoin de votre aide sans quoi il ne finira pas ses plans à temps. Afin de vous aider, il vous donne les séries de mots de passe à tester, les règles de production du langage utilisé par le générateur pseudo-aléatoire et un court rappel des notions du cours de LOG2810 sur les automates. Ainsi, vous pourrez l'aider à implémenter certains composants de son algorithme.

4 Description

Lors de la réunion secrète de présentation des plans, il vous explique les détails suivants.

- Les règles de production des différents langages se trouvent dans des fichiers qu'il vous fournit.
- Le format de ces fichiers est détaillé dans l'annexe.
- Chacune des cinq banques cibles utilise un langage différent. Il y a donc cinq fichiers de règles.
- Il faut construire un automate pour chaque langage. Cet automate reconnaîtra tous les mots du langage correspondant et pas d'autres mots.
- Les mots de passe et leurs variantes se trouvent dans des fichiers d'entrée. Il y a un mot de passe et ses variantes à tester dans chaque fichier, ainsi que le langage auquel la bonne variante appartient.
- Les fichiers d'entrée peuvent être utilisés dans n'importe quel ordre.

Voici un schéma possible d'un automate reconnaissant les mots 'A2A' et 'A2B' :

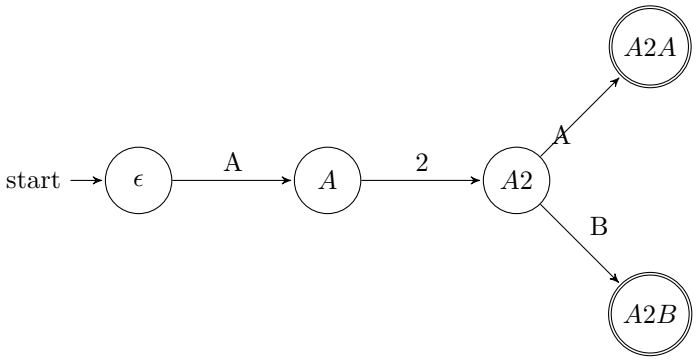

Fig. 1: Automate de validation

Un automate reconnaît un mot lorsqu'il existe une suite de transitions allant de l'état initial de l'automate à une sortie qui permet de former le mot (état final de l'automate). Ordre : la première partie du mot doit correspondre à la première transition, et ainsi de suite.

5 Composants à implémenter

C1. Écrire une fonction « creerAutomate() » qui lit le fichier texte contenant des règles et qui crée l'automate responsable de valider les mots de passe soumis par Joe. Cette fonction prend en paramètre le nom du fichier à lire.

C2. Écrire une fonction « trouverMotDePasse() » qui permet de trouver le mot de passe valide parmi les variantes proposées. Cette fonction prend en paramètre une liste (tableau, vecteur, etc.) de variantes à tester et un (numéro d')automate de validation.

C3. Écrire une fonction « traiterLesEntrees() » qui lit le fichier texte contenant un numéro d'automate et des variantes de mots de passe à tester. Cette fonction prend en paramètre le nom du fichier contenant les requêtes à traiter.

C4. Écrire une fonction « afficherLesMotsDePasse() » qui indique les mots de passe validés jusqu'à présent avec le numéro de l'automate associé.

C5. Faire une interface console qui affiche le menu suivant :

(a) Créer l'automate.

(b) Traiter des requêtes.

(c) Afficher les mots de passe valides obtenus.

(d) Quitter.

Notes

- Le programme doit toujours réafficher le menu, tant que l'option (d), ou « Quitter », n'a pas été choisie.
- L'utilisateur doit entrer un index valide, sinon le programme le signale et affiche le menu de nouveau.
- L'option (a) permet de lire un nouveau fichier de règles afin de créer l'automate correspondant. Pour lire un nouveau fichier de règles, le nom du fichier doit être demandé à l'utilisateur.
- L'option (b) permet de lire un fichier d'entrée, de déterminer l'automate à utiliser et de lui envoyer les mots de passe à tester. À la fin de l'exécution de cette option, le mot de passe valide doit être affiché.
- L'option (c) permet d'afficher un récapitulatif des mots de passe valides obtenus avec le numéro de l'automate associé.

Prenez note que selon votre convenance, le mot "fonction" peut être remplacé par "méthode" et vice-versa, et que plusieurs autres fonctions/méthodes peuvent être ajoutées pour vous faciliter la tâche.

6 Livrable

Le livrable attendu est constitué du code source et du rapport de laboratoire. Le livrable est une archive (ZIP ou RAR ou tar.gz) dont le nom est formé des numéros de matricule des membres de l'équipe, séparés par un trait de soulignement (_). L'archive contiendra les fichiers suivants :

- les fichiers .cpp ou .py ou .java ;
- les fichiers .h le cas échéant ;
- le rapport au format PDF ;
- les fichiers .txt passés en argument (option (a), option (b)), si leur format est différent de ceux fournis (vous pouvez en effet modifier le format des informations contenues dans les fichiers fournis sur Moodle, mais vous devez à ce moment-là les remettre avec vos travaux).

L'archive ne doit pas contenir de programme exécutable, de fichier de projet ou solution de Visual Studio, de répertoire Debug ou Release, etc. Les fichiers .cpp et .h, ou .py, ou. java suffiront pour l'évaluation du travail.

6.1 Rapport

Un rapport de laboratoire rédigé avec soin est requis à la soumission (format .pdf, maximum 8 pages). Sinon, votre travail ne sera pas corrigé. Le rapport doit obligatoirement inclure les éléments ou sections suivantes :

1. Page de présentation : elle doit contenir le libellé du cours, le numéro et l'identification du TP, la date de remise, les matricules et noms des membres de l'équipe.
2. Introduction avec vos propres mots pour mettre en évidence le contexte et les objectifs du TP.
3. Présentation de vos travaux : une explication de votre solution. Ajoutez le diagramme de classes complet, contenant tous les attributs et toutes les méthodes ajoutées.
4. Difficultés rencontrées lors de l'élaboration du TP et les éventuelles solutions apportées.
5. Conclusion : expliquez en quoi ce laboratoire vous a été utile, ce que vous avez appris, etc. Vous pouvez également indiquer le temps passé sur ce TP à des fins d'ajustement pour la prochaine session.

Notez que vous ne devez pas mettre de code source dans le rapport.

6.2 Soumission du livrable

La soumission doit se faire uniquement par Moodle.

7 évaluation

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
C3	2
C4	3
C5	2
C6	2
Total de points	18

8 Documentation

- `https://www.draw.io/` (Diagrammes UML)
- `https://www.lucidchart.com/pages/examples/uml_diagram_tool` (UML)
- `learnxinyminutes.com` (Apprendre rapidement les bases d'un nouveau langage)

Annexe

1 Format des fichiers de règles

Chaque fichier `reglesN.txt` comporte plusieurs lignes (le nombre de lignes varie).
Sur la première ligne se trouve le numéro de l'automate défini dans ce fichier (pas forcément le même que le numéro dans le nom du fichier). Sur les lignes suivantes se trouvent les règles de production, le "S" donnant le symbole de départ. Un signe "=" remplace la flèche habituelle.

2 Format des fichiers d'entrée

Chaque fichier `variantesN.txt` comporte plusieurs lignes (le nombre de lignes varie). Sur la première ligne, il y a le numéro de l'automate. Sur chacune des lignes suivantes, il y a une variante possible de mot de passe à tester. Une seule de ces variantes est valide selon les règles de l'automate désigné.