École Polytechnique de Montréal

Département de Génie Informatique et Génie Logiciel

LOG2810

Structures Discrètes

Automne 2018

TP2 : Automate d'arbre

Remis par :

Matricule	Prénom & Nom
1920352	Allan Beddouk
1839262	David Dratwa
1833489	Robin Canton Lamousse

Remise: 04 Décembre 2018

Introduction

Le cours de 2810 est un cours très théorique sur des notions mathématiques, durant ce TP nous allons essayer d'appliquer au réel. C'est pourquoi nous allons essayer d'utiliser les automates vus dans le cours pour modéliser une fonctionnalité d'un correcteur d'orthographe qui donne tous les mots contenus dans le lexique que nous lui donnons commençant par la chaîne de caractères que nous avons rentré. Nous devons mettre en place une interface qui prend une chaîne de caractères entrée par l'utilisateur afin d'afficher tous les mots du lexique commençant par cette chaîne. Nous devons aussi savoir à tout moment si le mot sélectionné a été utilisé dans les 5 derniers mots utilisée et incrémenter un compteur qui compte le nombre d'utilisations de chaque mot. Nous avons choisi de coder en C++ car c'est le langage dans lequel nous sommes tous le plus à l'aise et nous pouvons appliquer ce que nous avons appris en java dans le cours d' INF2010 (Structures de données et algorithmes) car cela ressemble beaucoup.

Nous allons donc essayer de créer et de coder notre premier automate réel utile, qui pourrait être utilisé dans une application de correcteur d'orthographe pour du traitement de texte par exemple.

Dans un premier temps nous allons expliquer en quoi consiste notre automate et comment nous avons mis au point un algorithme pour implémenter cet automate, par la suite nous allons décrire la création de l'interface que nous avons codé en QT sur QTcreator, puis nous allons décrire les problèmes rencontrés dans l'élaboration de ce TP et spécifier comment nous les avons surmontés si c'est le cas. Finalement nous conclurons sur les connaissances que nous avons acquises durant ce travail et nos impressions quant à celui-ci.

1/ La mise au point de l'automate

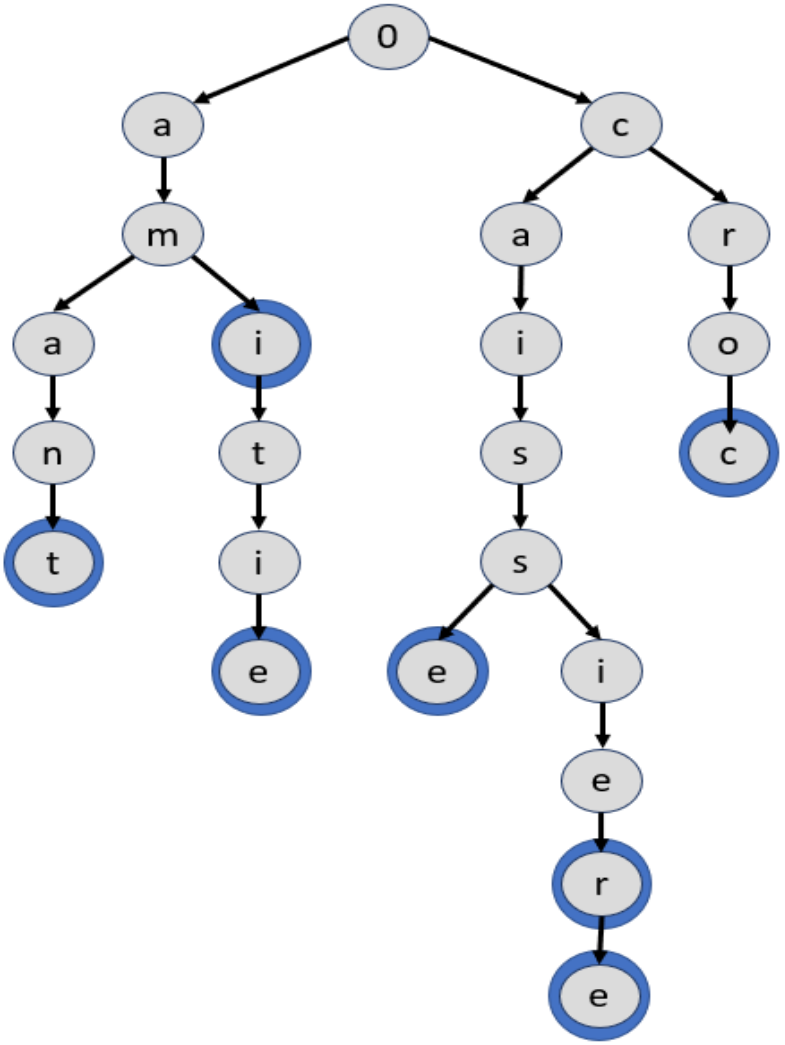
La mise au point de l'automate était le plus gros défi concernant la résolution du problème posé. Nous avons tout d'abord décider d'élaborer notre automate sur feuille de papier. Après plusieurs essais, nous avons finalement fait un parallèle entre l'automate que nous tentions de réaliser et un arbre vu plus tôt dans le cours INF2010 (Structure de données et algorithmes) et LOG2810 (Structures discrètes). Ainsi nous avons alors déterré les TP du cours de structures de données dans lesquels nous implémentons des arbres afin de s'en inspirer pour notre implémentation.

Une fois les classes s'apparentants aux noeuds et à l'arbre imaginées, le défi principal devient la création de l'arbre. Ainsi nous avons mis au point la méthode d'ajout de mot dans un arbre, développée en utilisant un algorithme récursif, elle ajoute chaque caractère à sa place (si le caractère n'est pas encore contenu dans l'arbre). La méthode développée viendra alors compléter le constructeur de l'arbre, car celui-ci ira chercher chaque mot dans le fichier 'lexique-n.txt' et utilisera l'algorithme afin de l'ajouter dans l'arbre. Une fois l'arbre créé, un nouveau problème se pose : développer un algorithme qui permet de retrouver les mots dans l'arbre.

Dans le but de développer cet algorithme, nous avons ajouté un attribut qui détermine si le noeud est un caractère final d'un mot. Ainsi il suffit de demander à l'utilisateur le début de mot qu'il recherche, grâce à la chaîne de caractères entrée, nous parcourons l'arbre afin de trouver tous les mots qui conviennent.

Ainsi l'élaboration de ces deux algorithmes permettent de résoudre le problème posé. Une fois tout cela développé, il restait à faire l'interface afin de présenter le résultat de notre solution à l'utilisateur. Nous avons développé une représentation d'un exemple d'arbre (disponible à la page suivante), afin d'illustrer notre solution.

Figure 1: exemple d'arbre créé avec notre automate à partir du lexique 1 ci-dessous



Lexique 1:

-amant -ami -amitié -caisse

-caissier -caissiere -croc

On note en bleu les états terminaux (finis) qui correspondent à la fin d'un mot du lexique. L'état 0 est un noeud vide (et qui le restera), représentant la racine de notre arbre, par la suite un mot est disponible à chaque noeud finaux. Dans notre code, le premier noeud est un caractère vide.

2/ L’interface

La conception de l’interface fut un vrai défi pour nous. Premièrement il y a eu un défi créatif à relever, il a fallu imaginer une interface facile d'utilisation qui répondait au mandat, cela a été fait rapidement. Deuxièmement le défi technique s’est posé. Il faut se rappeler que les seuls cours abordants des interfaces jusqu’ici sont : programmation orientée objet (le cours INF1010 en C++) et conception interfaces utilisateurs (le cours LOG2420). Étant donné que le cours LOG2420 s’oriente vers des interfaces web, nous avons dû nous appuyer sur nos connaissances développées dans le cours de programmation orientée objet. Ainsi durant ce cours un des chapitre porte sur les interfaces graphiques développées grâce au framework QT.

Ce framework permet le développement d’interfaces graphiques simples en utilisant des widgets, de plus il simplifie le modèle MVC en ayant besoin d’une seule classe pour le modèle et une seule classe pour la vue et le contrôleur. Alors une fois avoir imaginer le modèle de l’interface, il restait à l'implémenter et faire le lien avec les classes et méthodes qui leur sont associées.

L’implémentation de cette interface fût assez simple car le mandat concernant l’interface n’est pas très demandant (ce n’est d’ailleurs pas la difficulté du travail), il nous a simplement fallu implémenter les Slots et signaux nécessaire à notre programmations événementielle en utilisant les méthodes mise au point lors de la création et implémentation de nos méthodes.

3/ Les difficultés rencontrées

Durant ce travail, nous avons rencontré trois difficultés notables. la première difficulté était de réussir à implémenter l'arbre qu'on avait imaginé pour stocker les caractères du lexique, en effet un arbre imaginé sur feuille de papier peut s'avérer compliqué à développer virtuellement. Son implémentation impliquant énormément de récursivité, il nous a fallu beaucoup de temps afin d'obtenir le résultat final.

La seconde difficulté majeure fut la création de l'algorithme d'ajout d'un mot qui est un algorithme récursif, c'est ce qui nous a pris le plus de temps dans la réalisation de ce TP. Nous devions pouvoir rajouter n'importe quel mot dans notre arbre et pouvoir aller le rechercher ensuite en suivant la structure de la figure 1.

La troisième et dernière difficulté que nous avons rencontré se situait dans l'implémentation de l'interface graphique, en effet il fallait gérer les évènements à chaque clic et entrée de l'utilisateur, si cette fonctionnalité peut paraître simple à implémenter, cela ne l'a pas été pour nous. Effectivement, cette interface est la première que nous développons entièrement de notre cursus.

4/ Conclusion

Durant ce travail, nous avons pu aussi bien réviser des anciennes notions qu'apprendre de la matière. En effet, nous avons revu l'utilisation de l'API QT que nous avions apprise à manier en fin de session dans le cours d' INF1010 de programmation orientée objet, nous avons pu recréer une interface en nous aidant des anciens TPs que nous avions réalisé lors la dernière session. Nous avons aussi appris à appliquer les notions du cours d' INF2010 structures de données et algorithmes et de LOG2810 sur les structures discrètes pour créer un automate qui donne un résultat fiable pour chaque lexique rentré. Nous avons donc appris à nous adapter au réel, à adapter ce que nous apprenons à une utilité réelle pour le monde qui nous entoure étant donné que nous utilisons des éditeurs de texte et des applications de messagerie avec suggestion de mots tous les jours.