École Polytechnique de Montréal

Département de Génie Informatique et Génie Logiciel

LOG2810

STRUCTURES DISCRÈTES

Hiver 2017

TP2 : Automates et Langages

Remis par :

Matricule	Prénom & Nom
1801383	Philippe Courtemanche
1718526	Anthony Dentinger
1769769	Marc-Gaël Hounto

À :

<David Johannès>

Le 5 avril 2017

Introduction

Le contexte de ce travail pratique est celui où nous devons encoder une lexique en se servant d'un automate. En effet, il a plusieurs lexiques fournis sous format .txt et, à l'aide d'un automate, nous devons encoder le lexique choisi par l'utilisateur afin de rendre l'utilisation du lexique plus rapide et moins demandant en termes de ressources pour l'ordinateur. Ces fonctionnalités doivent d'ailleurs être programmées en C++. Ceci est fait dans le contexte où nous devons aider au développement d'une application contenant un éditeur de texte. L'utilisation de ce lexique fait donc partie de la réalisation de certaines des fonctionnalités de cet éditeur de texte.

L'objectif de ce travail pratique est donc d'utiliser ce lexique pour faire la complétion et la correction de mots écrits par l'utilisateur. Pour ce qui est de la complétion, au fur et à mesure que l'utilisateur inscrit les lettres de son mot, le programme doit lui suggérer une liste de mots qui commencent avec les lettres qu'il vient d'inscrire. D'autre part, pour la correction, dès que l'utilisateur actionne une touche du clavier qui est autre qu'une lettre, le programme doit vérifier l'orthographe du mot courant. Si l'orthographe du mot courant correspond à un mot du lexique, mais avec une lettre de différence, le programme doit suggérer une correction. S'il correspond à un mot du lexique ou s'il diffère de plus qu'une lettre, le programme ne réagit pas.

Présentation du travail

La Fig. 1 illustre le diagramme de classe de la solution apportée au problème précité.

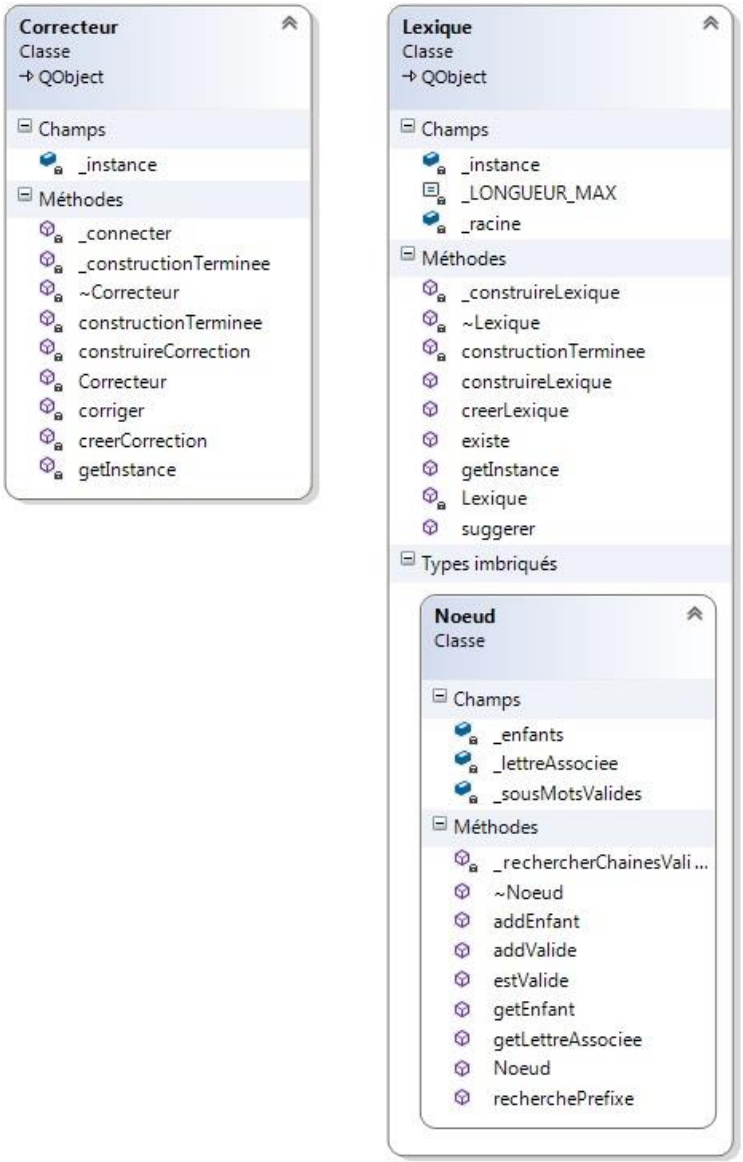

Fig. 1 – Diagramme de classes du modèle de la solution apportée

Notre lexique est représenté par une machine à états. L'idée de base est similaire à celle d'un dicitionnaire physique : la première lettre donne la section de la machine à états vers laquelle se rendre, la deuxième lettre en donne la sous-section, etc. Notre machine à états est toutefois légèrement modifiée par rapport à une machine à états habituelle, et ce pour les trois raisons suivantes :

1) La machine à états a la structure d'un arbre et n'est donc pas simplifiée.
2) La transition de la machine à états est inscrite dans l'état même. Ceci est possible dans la mesure la transition ne peut se faire que pour une lettre. Par exemple, le nœud «ba» ne peut être accédé qu'en lisant la lettre « a » depuis le nœud « b ».
3) La **hauteur de l'arbre est limitée** à une valeur paramétrisable.

Ce dernier point constitue la différence la plus grande avec une machine à états traditionnelle. Nous pourrions estimer la longeur des mots français à environ 7 lettres par mot. Subséquemment, **rajouter un mot tel qu' « anticonstitutionnellement » reviendrait à créer beaucoup d'états pour ne rajouter qu'un seul mot**, et donc à devoir parcourir plus de nœuds avant de se rendre au nœud désiré.

Le lexique a donc été programmé de telle sorte que la hauteur de l'arbre soit limitée. Ceci permet d'arrêter la création d'états lorsqu'un mot dépasse une certaine longueur. **Chaque état contient donc la liste des sous-chaînes qui sont valides** dans le lexique. Par exemple, supposons que nous soyons dans le nœud feuille qui s'atteint en lisant la chaîne « cordiale » (la longueur maximale du lexique a donc été fixée à 8). Ce nœud contient donc, entre autres, les sous-chaînes « », « s » et « ment », indiquant ainsi que les mots « cordiale », « cordiales » et « cordialement » sont valides. Les nœuds qui ne sont pas des feuilles et qui correspondent à un mot valide contiennent la liste composée uniquement de la chaîne vide. Ce serait, dans cet exemple, le cas de « cor ».

Notons qu'il est tout-à-fait possible avec cette méthode de simplifier l'arbre, ce qui nous donnerait un graphe. Toutefois, la solution proposée était suffisante pour les performances demandées.

Pour la **suggestion**, nous effectuons d'abord une descente dans l'arbre jusqu'à la section correspondant à la séquence de lettres inscrites par l'utilisateur, puis nous parconrons l'arbre en ordre infixe, ce qui correspond à l'ordre alphabétique. Nous arrêtons la recherche dès que nous avons atteint un certain nombre de mots trouvés.

Pour la **correction**, si le mot inscrit par l'utilisateur n'est pas valide, nous générons tous les mots qui ne diffèrent que d'une lettre du mot inscrit, et testons si ces mots appartiennent au lexique.

Difficultés rencontrées

Le lexique et le correcteur n'ont pas présenté de difficulté particulière ; la solution proposée au départ a fonctionné sans grande difficulté de conception ni de performances. Il faut dire que nous avions décidé de la solution et créé en équipe un diagramme de classe simple avant de commencer à créer la solution.

En revanche, la vue a présenté plusieurs difficultés, et a d'ailleurs représenté environ les deux tiers du travail du TP. En particulier, il fallait s'assurer de **ne pas essayer de lire en dehors de la boîte de texte**, ce qui causait des segfaults. De plus, pour les remplacements, il fallait sélectionner tout le mot saisi et de le remplacer par la suggestion ou la correction (par exemple, **à partir de « bonjiur », il faut obtenir « bonjour », pas « bonjour r »**). Il est également arrivé que l'on ne pouvait plus supprimer les doubles espaces précédés par un mot, puisque le correcteur réinsérait l'espace supprimé en réaction à sa suppression. Nous avons donc finalement décidé de **réagir au changement de la position du curseur plutôt qu'au changement de texte**, ce qui a simplifié et solutionné ces problèmes.

Conclusion

Ce laboratoire nous a été utile de plusieurs manières. Premièrement, nous avons pu consolider nos connaissances en programmation orientée objet en langage C++, ainsi qu'en Qt Creator. De plus, nous avons pu revoir et utiliser les notions d'automates et langages. Bien sûr, nos compétences de travail en équipe en temps de surcharge de travail ont été testées et consolidées.

Pour ce qui est de nos apprentissages, nous avons premièrement appris à implémenter un automate en C++ et des algorithmes pour effectuer la correction et la complétion. De plus, nous avons appris à nous servir de quelques nouvelles fonctionnalités de Qt Creator, telle que la barre de progression par exemple.

Pour nos projets futurs, d'abord, nous voudrions implémenter une fonctionnalité qui permettrait de souligner en rouge les mots mal orthographiés. Ensuite, nous voudrions tenter d'analyser la syntaxe de la phrase écrite par l'usager afin de corriger davantage d'erreurs.