LADIER Kévin

COURSIER Pierre

Licence 3 Informatique

Année 2014-2015

Responsable de projet : KOUCHNARENKO Olga

Projet de 3ème année de licence

Logiciel Autoroute

Animation d'algorithmes sur les automate d'états finis

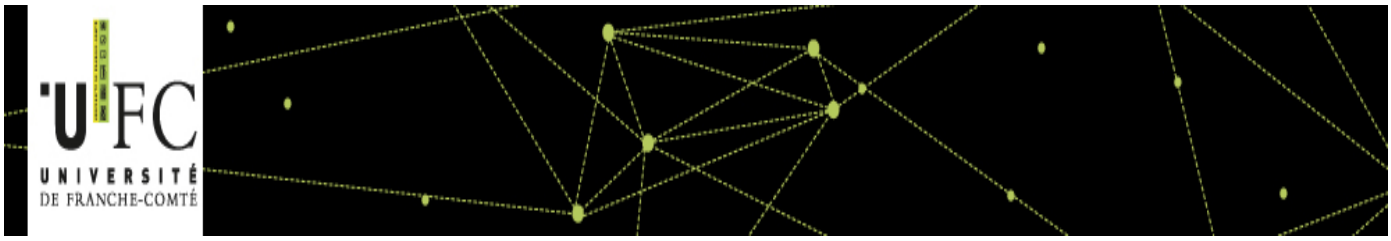

Table des matières

Introduction

Dans le cadre de notre troisième année de licence informatique nous avons réalisé un projet tuteuré en binôme. Le projet s'intitule animation d'algorithme sur automates d'états finis, il était demandé de réaliser un logiciel, à but pédagogique, permettant d'expliquer de manière la plus claire possible les étapes des algorithmes appliqués aux automates.

Ce logiciel serait utilisé par des professeurs pour des étudiants dans le cadre d'un enseignement à distance. Un petit dessin étant souvent plus explicite qu'un grand discours, ce logiciel pourrait faciliter la compréhension des algorithmes par les étudiants.

1°) Présentation du sujet

Nous devions donc permettre au logiciel de fonctionner sur les plateformes Linux et Windows. Le sujet nous imposait l'implémentation de fonctionnalités pour faciliter la compréhension des algorithmes pour les utilisateurs du logiciel. Il était question de permettre un déroulement séquentiel des algorithmes et la possibilité de traverser l'algorithme dans les deux sens, et ainsi revenir sur une étape qui pouvait poser problème.

Au départ nous disposions également de deux versions existantes du projet, réalisées dans le même cadre ces dernières années. Il nous fallait donc les étudier afin de décider quelle version utiliser. Reprendre une version du projet nous imposait également les outils à utiliser.

Enfin nous nous étions fixé comme objectif de créer une documentation pour notre logiciel pour faciliter sa prise en main par les utilisateurs comme par les futur développeurs.

Après l'analyse de notre sujet nous devions donc :

- Analyser les projets existants.

- Proposer des nouveaux algorithmes (minimisation standardisation…).

- Réaliser une documentation du logiciel.

- Garder les éléments déjà présents dans la version utilisée (un exécutable, support de Linux et Windows).

2°) Analyse de l'existant

Au commencement de notre projet notre encadrante avait mis à notre disposition deux versions du logiciel. Nous devions donc dans un premier temps les étudier afin de savoir laquelle nous allions poursuivre.

2.2 °) Version de 2010

La première des deux versions datait de 2010, elle proposait une version du logiciel fonctionnelle sur Linux et sur Windows. Elle offrait à l'utilisateur aux travers d'une interface la possibilité de réaliser les actions de bases sur les automates : créer un automate, ouvrir un automate existant, elle permettait également de réaliser le produit de deux automates et de déterminiser un automate. Elle proposait également la fonctionnalité de parcours des algorithmes dans les deux sens. Le logiciel répondait donc parfaitement au côté pédagogique demandé par le sujet. D'autre part les créateurs du logiciel avaient réalisé de nombreux test (en fabriquant des automates types) de chacune de leur fonction. Ces tests démontraient la fiabilité de leur produit. Cependant elle ne fournissait aucune documentation logicielle, et le code étant dépourvu de commentaire il nous était difficile de le prendre en main. Cette version étant néanmoins déjà très complète par rapport au sujet, elle semblait convenir à nos besoins.

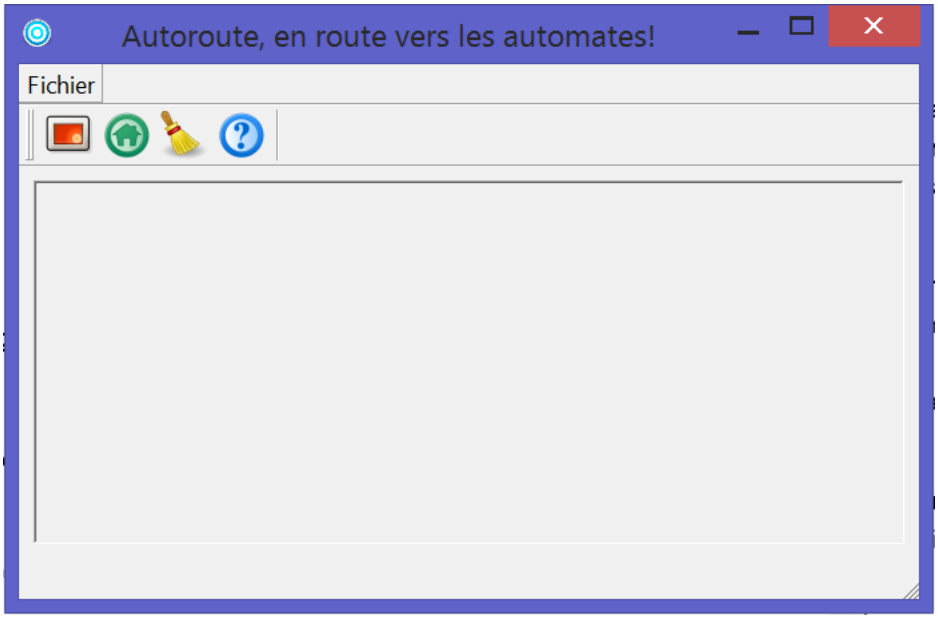

*Figure **Erreur ! Utilisez l'onglet Accueil pour appliquer 0 au texte que vous souhaitez faire apparaître ici.**.**Erreur ! Utilisez l'onglet Accueil pour appliquer 0 au texte que vous souhaitez faire apparaître ici.**.1 Interface de la version 2010 du logiciel*

2.2°) Version de 2013

La deuxième version en date du logiciel avait été réalisée en 2013. Elle reprenait la première version du projet. Cependant après les développeurs n'avait pas réussi à lier leur travail à l'interface graphique, ainsi après l'implémentation de leur algorithme dans le logiciel, aucune visualisation du résultat n'était possible. Aucun test n'a pu donc être réaliser. Il nous paraissait donc extrêmement difficile de prendre en main cette version.

De plus cette version n'était utilisable que sur Windows, la compatibilité avec Linux a été abandonné.

2.3°) Bilan de l'étude

Cette donc naturellement qu'après notre étude nous avons décidé de poursuivre le projet en prenant la version de 2010 comme base.

A la suite de cette analyse avions donc notre liste finale d'objectifs à réaliser :

- Mise en place d'un l'algorithme de minimisation d'automates.
- Mise en place d'un algorithme de standardisation d'automates.
- Mise en place d'une documentation utilisateur/développeur pour une prise en main plus intuitive.

Nous avions également la liste des outils avec lesquels nous allions devoir travailler. Le code de la version du logiciel choisie était implémenté en C++, l'interface graphique avait été réalisée à l'aide du logiciel QTCreator, et Graphviz qui permet la réalisation de fichiers dot (format des fichiers automates) ainsi qu'un affichage correct des automates à partir de ces .dot.

3°) Une meilleure accessibilité

3.1°) Mise en place d'un serveur GIT

Travailler à deux sur un même projet a été notre premier obstacle. En effet il nous était très peu pratique de modifier notre code les deux en même temps. Nous avions eu à travailler à deux sur un même fichier et avions à cette époque fonctionner en travaillant chacun notre nous. Il nous fallait ensuite s'envoyer par mail le fichier après chaque modification pour être tous les deux au même niveau. Cette méthode étant très fastidieuse et très peu productive nous avons donc réfléchis à la mise en place d'un système de travail commun. Après quelques recherches deux solutions s'offraient à nous : SVN ou GIT.

Nous avions déjà eu l'occasion d'étudier SVN au cours de notre cursus à dans le cadre de notre module MOP avec SVN Tortoise. Ce logiciel répondait à nos attentes en termes de possibilité. Pour GIT nous l'avions déjà utilisé dans un cadre personnel. Ayant une meilleure connaissance de GIT nous avons préféré utiliser cet outil.

Afin de perfectionner l'idée d'accessibilité au logiciel pour tous, nous avons donc décidé d'utiliser GiTHub qui permet la mise en place de serveur public. Les sources de notre code sont donc disponibles pour tout le monde à l'adresse de notre dépôt.

Le logiciel SmartGit nous a apporté un complément et facilite la prise en main de cet outil extrêmement puissant.

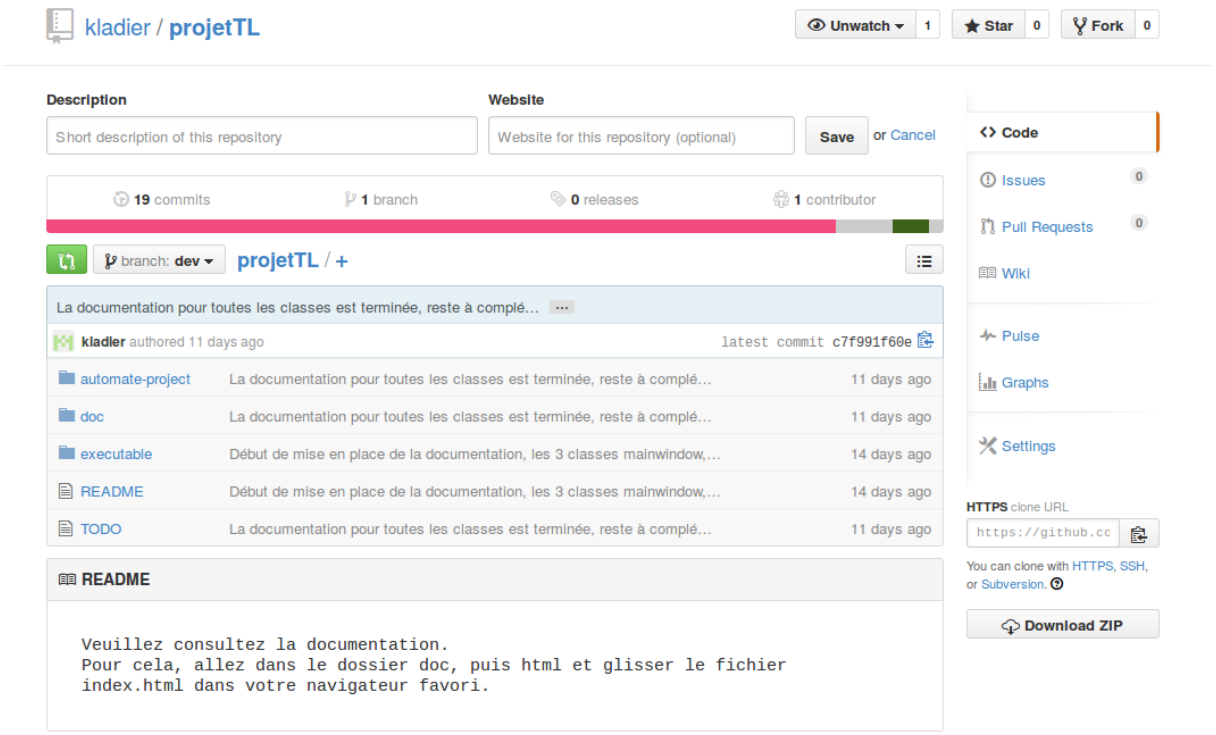

Figure 2 Dépot GIT

3.2°) Insertion d'une documentation.

Lors de l'étude des projets existant le manque d'information quant au logiciel et à son fonctionnement ont été un gros frein à la prise en main du code. Cela a donc renforcé notre détermination pour la création d'une documentation la plus complète et intuitive possible.

3.2.1°) Doxygen

Doxygen est un outil qui permet à partir de code C++ et au moyen de balises dans le code de générer une documentation très complète sous différents formats comme LateX ou HTML. La première étape consiste à installer Doxygen et à le configurer. Il faut pour cela créer un fichier qui contiendra toutes les informations nécessaire à la création de la doc, qui sera appelé lors du lancement de la fonction de création (encodage, nom du projet, …)

```
C:\Users\Stank\Desktop\projetTutore.git\doc>doxygen Doxyfile
```

Figure.3 Commande initialisation doxygen

Afin de pouvoir générer la doc souhaitée, il faut implémenter dans les fichiers des commentaires en utilisant des balises qui seront ensuite traduites par le logiciel et généreront selon le format de sortie une documentation adaptée. Par exemple pour référencer toutes les fonctions dans la documentation il faut se placer dans les fichiers Headers (.h) du projet.

```
/*!
     *  \brief Constructeur
     *
     *  Construit l'automate passé en paramètre
     *
     *  \param a : automate à construire
     */
Automate(const Automate&a);
```

*Figure**Erreur ! Utilisez l'onglet Accueil pour appliquer 0 au texte que vous souhaitez faire apparaître ici.** *4 Exemple de doc pour une fonction*

Doxygen crée ensuite, à l'aide du fichier de configuration crée au préalable une documentation complète en s'appuyant donc sur tous les commentaires qu'il trouve au format adéquat dans le code. Cette documentation peut être générée dans plusieurs formats. Nous avons pour notre part choisi de la générer en HTML et en format LateX qui sont les deux formats les plus standards et les plus faciles à utiliser.

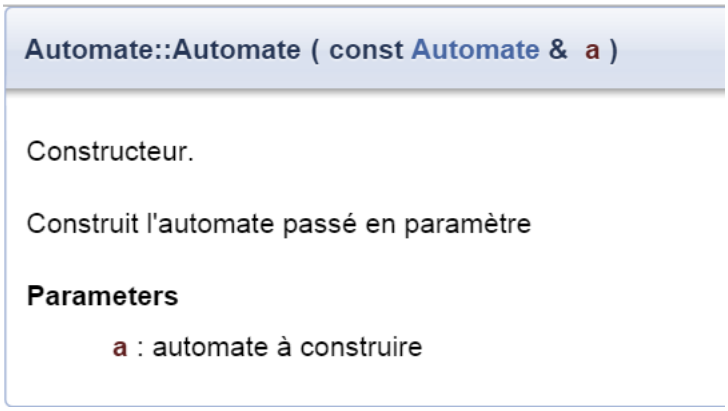

Figure Exemple de documentation HTML

Pour confirmer le côté intuitif de notre documentation nous l'avons fait découvrir à des personnes extérieures au projet, qui nous ont affirmé de sa simplicité.

3.2.2°) Des algorithmes clairement expliqués.

Il nous paraissait indispensable de faire comprendre au mieux les algorithmes aux étudiants. C'est dans cette idée que l'implémentation du code des algorithmes a donné lieu à des explications claire et précise du déroulement des opérations dans une fenêtre de l'interface (déjà présente et utilisée dans la version précédente)

3.3°) Une installation très simple

La première utilisation du logicielle par les utilisateurs est l'installation qui peut s'avérer parfois très fastidieuse si elle est compliqué et mal expliquée. C'est pour éviter cette situation qu'un setup a été mis en place pour les utilisateurs de windows. Les utilisateurs de linux pour leur part doivent suivre le tutoriel d'installation fournit par la documentation du logiciel (mais ils sont en théorie habitué à l'absence de .exe dans leurs programmes).

4°) Implémentation des nouvelles fonctionnalités

Après une première prise en main du code du logiciel le premier problème survenu était lié à sa date de création. En effet depuis 2010, le framework QT est passé de la version 4 à la version 5, ce qui entrainait des erreurs d'exécutions, nous avons donc du mettre à jour les bibliothèques utilisées et faire quelques modifications car les paramètres de certaines méthodes avaient changés.

Afin de rendre le logiciel plus complet par rapport à sa première version, il nous a été demandé d'implémenter 2 nouveaux algorithmes. A savoir la minimisation et la standardisation. Chacun de ses deux algorithmes donne lieu à un nouveau bouton sur l'interface graphique. et j'ai mis les images des boutons

Figure 5 Bouton des nouveaux algorithmes

4.2°) Algorithme de standardisation

Nous avons commencé par la standardisation, car c'était le plus simple à mettre en place pour commencer et une bonne occasion de prendre en main les sources et comprendre le fonctionnement du logiciel.

C'était le problème rencontré le plus important car c'est celui qui nous a pris le plus de temps à régler : comprendre le fonctionnement global et en détails du logiciel.

Ensuite, nous avons du choisir les structures de données à utiliser pour ces fonctionnalités. Nous nous sommes inspirés de ce qui avait déjà été utilisé pour la déterminisation, en remplissant un vecteur contenant une paire : le 1er élément étant un automate et le second la chaine contenant le commentaire à mettre à chaque étape. Chaque élément du vecteur représente donc en fait une étape.

Les boutons suivant et précédent servent donc à incrémenter ou décrementer l'indice dans ce vecteur et afficher les éléments correspondants.

Nous avons utilisé cette structure aussi pour la minimisation ce qui permet aussi de commencer à rendre le code plus clair comme la même structure est utilisée pour toutes les fonctionnalités sauf le produit.

Pas de problème particulier lié à la conception, la standardisation est plutot simple à mettre en place une fois qu'on a compris le fonctionnement du logiciel.

4.3°) Algorithme de minimisation

A l'inverse la minimisation était beaucoup plus compliquée pour la conception. La minimisation d'un automate se réalise en 2 étapes : faire le tableau de minimisation puis construire un nouvel automate à partir du tableau. Construire le tableau de minimisation était une étape assez technique, longue à mettre en place. Pour m'aider j'ai pris un exemple un peu complexe d'automate à minimiser, et j'ai fais des tests au fur et à mesure du développement. J'ai aussi du développer plusieurs fonctions utilisées seulement par la minimisation pour rendre plus clair la fonction de minimisation et pour éviter de répéter les mêmes blocs de code.

Il y a des explications pour construire le tableau de minimisation ainsi que pour construire le nouvel automate à partir de ce tableau.

Pour les 2 algorithmes, une fois que la fonctionnalité était mise en place, nous avons fait une batterie de tests qui nous a permis de rectifier plusieurs erreurs restantes.

5°) Problèmes rencontrés

Une fois encore, bien qu'il soit intéressant de prendre en main les sources d'un logiciel, il manquait vraiment un vrai travail de documentation et de mini-tutoriels. Il nous a donc été assez long de comprendre le fonctionnement du logiciel. Conscient de cette problématique, nous nous sommes donc particulièrement attaché au fait qu'il soit bien plus facile et rapide pour les futurs développeurs de comprendre le fonctionnement du logiciel, de connaître les outils qui leur seront utiles, de prendre en main les sources. Nous en avons aussi profité pour produire les textes destinés aux utilisateurs : installation du logiciel et utilisation basique. Il nous a fallu aussi faire des recherches pour trouver un outil équivalent à Javadoc que nous avions déjà utilisé : nous sommes assez rapidement tombés sur Doxygen qui correspondait très bien à nos besoins.

La documentation a été longue et fastidieuse à réaliser. Nous l'avons faite juste après avoir implémenté la standardisation ce qui nous avait permis de prendre en main une partie des sources. Elle nous a tout de même obligé à faire le tour de tous les attributs et méthodes de toutes les classes et de comprendre le fonctionnement en détails de ce logiciel.

Nous avons déjà parlé des problèmes liés à la conception, surtout pour la minimisation qui demandait un certain niveau d'abstraction et une bonne connaissance des structures pratiques en C++ ainsi qu'une connaissance détaillée de la minimisation.

Conclusion

Ce projet nous a été très bénéfique car il nous a permis d'approfondir nos connaissances dans divers domaines. D'une part sur le travail en équipe qui nous a fait utiliser Git un peu plus sérieusement que lors de nos expériences passées. Il nous aussi été très bénéfique d'utiliser un code déjà existant car cela nous a permis de nous rendre compte de l'importance de la documentation dans un code. Il est également très intéressant de reprendre le travail de quelqu'un d'autre et d'étudier son style de développement. Travailler sur ce logiciel a grandement enrichi nos compétences en C++ et avec le framework QT. Il nous a également permis de découvrir un outil puissant (Doxygen) et d'apprendre à le maîtriser.

En reprenant la liste des objectifs de départ il semblerait que le contrat ait été remplit car toutes les consignes ont étés respectées. Il reste cependant toujours des améliorations à apporter (rien n'est parfait) qui pourront être implémentés par les prochains étudiants en charge de ce projet. Il existe également de nombreux algorithmes qui mériteraient d'être développés afin de faciliter l'apprentissage des automates à des étudiants (expressions régulières, compléter un automate etc…)

Lien github : https://github.com/kladier/projetTL.git