Documentation et rapport de projet : Automates Cellulaires

2019 - 2020

Ladislas WALCAK - Simon DRIEUX

Introduction

Ce projet a pour but le développement d'un programme C permettant la modélisation d'un automate cellulaire simple, ainsi que sa sauvegarde éventuelle dans un fichier .pgm.

Un automate cellulaire consiste en une matrice de cellules, chaque cellule contenant un état choisi et qui évolue au cours du temps en fonction de l'état de ses voisins. Les changements d'états des cellules sont appliqués selon une règle définie par l'utilisateur. Selon la dimension de l'automate, le nombre d'états possibles ou encore la règle, l'affichage final sera différent. Dans notre projet, nous traiterons des automates de dimensions 1.

Structure du code

Le projet a été structuré en plusieurs répertoires, ainsi qu'un fichier Makefile adapté contant plusieurs commandes utiles. La structure du projet est :

- Un dossier *include* contenant tous les fichiers en-têtes (.h)
- Un dossier *src* contenant tous les fichiers sources (.c)
- Un dossier *obj* créé lors de la compilation contenant tous les fichier objets (.o)
- Un dossier *doc* créé lors de la génération de la documentation avec l'outil Doxygen
- Un fichier *Makefile* contenant les commandes liées au projet
- Un fichier *config.txt* pouvant être utilisé pour la configuration de l'automate

De plus, ce projet contient un nombre de fichiers supplémentaires lié au projet :

- Un fichier *README.md* contenant des instructions
- Un fichier *.gitignore* lié au répertoire Git du projet
- Un fichier *Sujet.pdf* contenant le sujet détaillé du projet
- Un fichier de configuration doxygen *Doxyfile* nécessaire pour la génération de la documentation

Afin de respecter les consignes données, seul les dossiers *src* et *include*, ainsi que les fichiers *Makefile* et *config.txt* sont stockés dans l'archive. Ces fichiers sont disponibles sur le répertoire Github du projet (https://github.com/SimonDr18/C-ellular-project).

Lors du développement du projet, nous avons décidé d'utiliser le format .txt pour le fichier de configuration de l'automate. Ce choix a été fait pour deux raisons. Premièrement, le format .txt est un format utilisable par tous les ordinateurs, permettant une modification simple et rapide et une grande portabilité. De plus, ce format de fichier ne requiert aucune en-tête, permettant également une simplicité d'utilisation au sein du programme lui-même.

Le projet est constitué de 5 couples de fichiers en-têtes / sources, ainsi qu'un fichier source supplémentaire faisant la liaison de tous ces fichiers.

- Les fichiers *constants* contiennent l'ensembles des constantes, importations de bibliothèques nécessaires, et les deux structures ("automaton" et "pgm") partagés par tous les fichiers, regroupés au sein d'un même fichier dans un but de simplification.
- Les fichiers *menu* contiennent les déclarations et définitions des fonctions d'affichage et gestion du menu utilisé par l'utilisateur afin d'utiliser le programme. Ce menu consiste en une suite de question afin de paramétrer l'automate.
- Les fichiers *pgm* contiennent les déclarations et définitions des fonctions utiles à l'utilisation de la structure "pgm", utilisé lors de l'enregistrement de l'automate au sein de d'un fichier .pgm (nommé *output.pgm)*.
- Les fichiers *printers* contiennent les déclarations et définitions des fonctions par défaut d'affichage et de sauvegarde de l'automate.
- Les fichiers *structure* contiennent les déclarations et définitions des fonctions utiles à l'utilisation de l'automate.
- Le fichier *automaton.c* est le fichier faisant la liaison entre tous ces fichiers, lançant le programme.

Lors du développement de ce projet, des fonctions ont dû être modifiées ou bien sont devenues inutiles, ainsi que certains fichiers. Tous ces éléments ont été totalement supprimés du code afin d'obtenir un version propre du projet, et ne sont donc présents qu'au sein de l'historique des version du répertoire Git.

Afin de permettre un compréhension facile du programme, le code a été fortement commenté lors de la phase de développement. De plus, toujours dans un soucis de compréhension, la possibilité de générer une documentation grâce à l'outil doxygen a été ajoutée.

Lancement du programme

Un nombre de commandes ont été ajoutées au fichier *Makefile* afin de compiler et lancer le programme :

- **make clean** - permet la suppression de tous les fichiers obtenu après la compilation
- **make** - permet de compiler le projet afin d'obtenir l'exécutable *exec*
- **make run** - permet de lancer le programme compilé avec l'outil Valgrind
- **make doc** - permet de générer la documentation (nécessite un fichier *Doxyfile*)
- **make archive** - permet de placer les dossiers *src* et *include*, ainsi que les fichiers *Makefile* et *config.txt* au sein d'une archive *WALCAK_DRIEUX.tar.gz*

Fonctionnement

Le programme est divisé de manière à s'exécuter "fichier par fichier".

Dans un premier temps, les fonctions de *menu* sont appelées afin d'obtenir la configuration de l'automate. Ces fonctions, grâce à une série de questions, permettent de déterminer la configuration.
Lors de l'utilisation d'un fichier de configuration, le fichier doit respecter de manière strict ce format (chaque ligne doit finir par un retour à la ligne) :

```
States=_
Size=__
Iteration=__
Start=__________
Rule=__________
```

Avec $2 \leq States \leq 10$, $5 \leq Size$, $2 \leq Iteration$, Start une chaîne de caractère de longueur *Size* et contenant des caractères $\in [0, States[$, et enfin Rule une chaine de caractère de longueur $((States - 1) * 3) + 1$ et contenant des caractères $\in [0, States[$.

Ensuite, ce sont les fonctions de *structure* qui sont appelées avec la configuration donnée par l'utilisateur, afin de créer et calculer les différentes itérations de l'automate.

Pour finir, les fonctions de *printers* sont appelées pour l'affichage et, dans le cas de la sauvegarde en fichier .pgm, ces fonctions recourent au fonctions de *pgm* pour la gestion de ce format de fichier.

Améliorations

Dans un premier temps, l'amélioration du fichier de configuration pourrait permettre une plus grande flexibilité d'utilisation pour l'utilisateur. Actuellement, le fichier doit suivre de manière très strict le format du fichier, et peut donc mener à des erreurs. Permettre plus de flexibilité sur les espaces, les retours à la ligne ou l'ordre des paramètres serait de bonnes améliorations.

Ensuite, l'amélioration de la lecture des paramètres, par l'utilisateur ou dans un fichier, peut être améliorée. Actuellement, lors de la lecture de la règle et de la première itération de l'automate, les données sont stockées dans des tableaux. Cependant, il est actuellement impossible de prédire et/ou limiter la taille des données envoyées par l'utilisateur, pouvant donc mener à la réception de données plus grand que la taille des tableaux définis. Afin de limiter cela, des indications sont affichés à l'écran afin de guider l'utilisateur, ainsi que des tableaux plus grand que nécessaire sont créés.