LO21 - Rapport intermédiaire 3

Romain de Laage, Victor Blanchet, Maxime Goret, Luning Yang, Boris Cazic, Léon Do Castelo

6 juin 2021

1 Introduction

Suite à la répartition précédente, nous avons fait une nouvelle réunion afin de faire le point sur notre avancée. Nous avons mis en commun de nouvelles idées pour établir et répartir précisément ce qu'il nous reste à faire. Nous estimons avoir avancé le projet à 65%.

2 Avancée depuis la dernière fois

— Conceptualisation et implémentation du voisinage : gestion du voisinage arbitraire par Boris, Maxime et Luning (3h). Luning a amélioré la structure entre les classes et ses fonctions du voisinage (2h). Maxime a finalisé l'implémentation des voisinages de Moore et Neumann (3h).

— Conceptualisation et implémentation d'un automate : la conceptualisation (que doit faire la classe, avec quelles autres classes doit elle être liée, etc.) a été pensée par Romain (3h) et Leon (2h) avant une mise en commun des idées et une répartition de l'implémentation.
Romain s'est occupé du singleton, du constructeur, destructeur, opérateur d'affectation, constructeur par recopie, et du début d'implémentation d'un Timer (5h).
Leon s'est occupé du lien entre l'Automate et les grilles : gestion du buffer, de la grille courante, de la possibilité de naviguer dans le buffer, de calculer la grille suivante (4h).

— Conceptualisation et début d'implémentation de l'interface : création d'une classe Autocell par Victor permettant de générer une interface graphique sous la forme d'une fenêtre sur laquelle apparait un ensemble de Qt Widgets.
Elle permet de choisir un modèle enregistré, ou, d'en paramétrer un nouveau (une nouvelle fenêtre s'ouvre alors. Celle ci est à implémenter). Elle permet également de configurer une grille de départ après le choix d'un modèle (remplissage, dimensions, sélection parmi des grilles préenregistrées associées au modèle). L'utilisateur peut également contrôler une simulation (définir un pas de temps, démarrer / stopper l'exécution, sauvegarder la grille courante). Elle permet d'afficher la grille par l'intermédiaire d'un QTableWidget. Lors d'un clic sur une cellule du réseau, son état est incrémenté et sa couleur change en conséquence. Enfin, elle peut remettre à zéro la configuration de l'application. Tout cela a pris 18h.
Boris s'est chargé du paramétrage d'un modèle en essayant de trouver un compromis entre ergonomie et fonctionnalité (5h, risque d'évoluer). L'implémentation a pris environ 7h, elle prend la forme d'un widget intégré à l'interface, menant à une nouvelle fenêtre dans laquelle l'utilisateur va pouvoir paramétrer son automate manuellement.

— Doxygen : géré par Romain (1h30), la configuration est terminée et quelques commentaires ont été mis mais la majorité de la documentation reste à faire.

— Définir un format de sauvegarde des données (bibliothèque d'automates, règles, voisinages, structures, grilles, ...) et une manière de les importer dans l'application : géré par Romain (6h) et Leon (6h30) en SQLite.
Romain s'est chargé de la conceptualisation des tables Automate, ReglesTransition, ReglesVoisinage et CoordVoisinage qui permettent de stocker les attributs d'objets des classes Voisinage et Fonction, qui sont liées à une instance Automate. Il a fait les premières recherches pour implémenter SQLite dans le programme et a rédigé les premières fonctions. Ses implémentations sont presques terminées, il est possible de récupérer dans des fonctions getAutomates(), getFonction(), et getRegleVoisinage() des objets correspondant aux classes du programme à partir des tuples stockés dans la base de donnée.
Leon s'est chargé de la conceptualisation des tables Reseaux, EnsembleEtats, Etats, et Cellules. Beaucoup d'itérations ont été nécessaires avant d'arriver à une version définitive, les fonctions permettant de récupérer les objets correspondants sont encore à finaliser. Il a également mis en forme l'UML de la base de données en l'état.

3 Répartition des tâches restantes

— Conceptualisation et implémentation d'un automate :
Gérer l'ensemble des états d'un automate (i.e. les états que les cellules des grilles peuvent prendre) ; timer qui exécute automatiquement l'automate à intervalle régulier quand il est actif ; quelques modifications de voisinage pour être cohérent –> Romain (environ 4h).

— Conceptualisation et implémentation de l'interface :
Changer le label du bouton "run/stop" pour lancer la simulation lors du clic ; rédiger une notice pour l'utilisateur ; associer les widgets de contrôle de l'exécution aux méthodes de l'automate pour contrôler la simulation et afficher graphiquement le réseau ; générer le contenu des listes déroulantes pour sélectionner un modèles / sélectionner une grille de départ ; associer le bouton "sauvegarde de la grille courante" à une méthode de sauvegarde en mémoire de celle-ci –> Victor (environ 10h).
Mise en place d'une grille où l'utilisateur sélectionne les cellules faisant partie de son voisinage ; établissement d'un lien entre cette interface et le code permettant l'ajout de nouvelles règles/fonctions de transition –> Boris (environ 15h).
Adaptation de certaines classes pour faciliter leurs interactions avec l'interface ; récupération des entrées de l'utilisateur ; paramétrage de l'application –> Maxime (environ 15h).
Gestion des voisinages dans l'interface –> Luning (environ 10h).

— Définir un format de sauvegarde des données (bibliothèque d'automates, règles, voisinages, structures, grilles, ...) et une manière de les importer dans l'application :
Stockage d'anciens voisinages par rapport à un voisinage courant –> Luning (environ 2h).
Fonctions d'enregistrement et de lecture de grilles et de l'ensemble d'états –> Leon (environ 6h), en discussion avec les personnes chargées de l'implémentation.
Enregistrement des règles de transition, des règles et coordonnées de voisinage d'un automate –> Romain (environ 6h).
Stockage des modèles demandés dans le sujet –> Romain et Leon (environ 10h).

— Doxygen :
Rédaction de la documentation –> Romain (environ 4h).

— Vidéo de présentation : Boris, Maxime et Luning (environ 3h).

— Rapport final : Victor et Leon (environ 10h).

— Nous ne pensons pas avoir le temps pour réaliser les tâches optionnelles. Cela se décidera en fonction de l'avancée de chacun lors de la semaine.

4 Conclusion

L'implémentation a bien avancé, il nous faut maintenant l'intégrer à l'interface. Cela nécessitera plus de communication qu'auparavant.

Il va falloir avancer au plus vite dans la semaine afin de se laisser le temps de produire convenablement les livrables attendus.