|  |
| --- |
| Ecole Centrale de Nantes |
| Projet d’application |
| Bioinfo gPH Interface |



Etudiant :

Jean Baptiste Clavel

Thomas De Pujo

Rémi Thang

Accompagnateur :

Maxime Folschette

Julie Gras

**Objet du document**

Ce rapport a pour but de présenter le travail effectué par trois étudiants de l’Ecole Centrale de Nantes sur le projet gPH Interface dans le cadre d’un projet de groupe, de Janvier à Mars 2013

**Remerciement**

L’équipe projet souhaite remercier leur encadrant Maxime Folschette et Julien Gras pour leur disponibilité et le suivi qu’ils ont fait du projet.

Nous tenons aussi tout particulièrement à remercier Gaëtan Girin, Marlène Jaulin, Quentin Servais Laval et surtout Jérémy Amar, membres du précédent groupe ayant travaillé sur ce projet, pour leurs réponses à nos questions et leur travail de documentation ayant grandement facilité le début de notre propre travail.

Ce projet a permis aux membres de l’équipe de mettre en application les connaissances acquises au cours de leur cursus au sein de l’Ecole Centrale de Nantes mais aussi d’acquérir des compétences nouvelles sur de nouveaux outils informatiques.

Nous avons mené ce projet avec plaisir et enthousiasme et espérons que notre travail sera profitable à l’équipe MeForBio.

# Introduction

## Contexte et cadre de travail

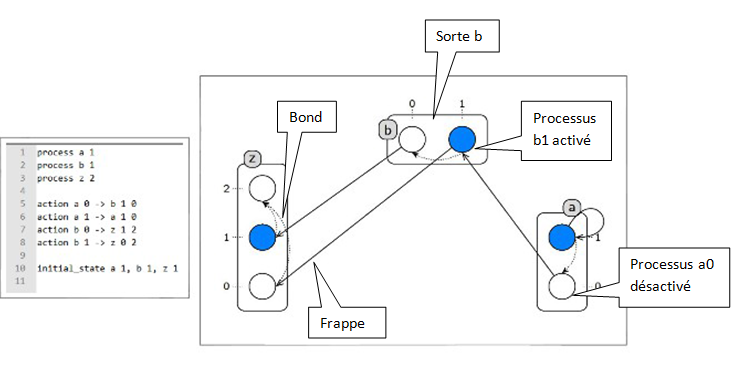
Le logiciel gPH Interface s’inscrit dans un projet mené par l’équipe **MeForBio** (**Mé**thodes **For**melles pour la **Bio**-Informatique) qui a pour but de permettre une utilisation graphiquement du logiciel Pint, lui aussi développé par l’équipe MeForBio. Cette dernière a développé une nouvelle manière de modéliser la représentation des réseaux biologiques de régulation de grande ampleur : le formalisme *Process Hitting*, ou frappes de processus. Le logiciel Pint a été créé pour pouvoir manipuler des modèles de Process Hitting via différents programmes (simulation stochastique, recherche d’états stables, …). Cependant, l’utilisation de Pint, via une console en ligne de commande, peut s’avérer ardue pour toute personne n’étant pas familière avec ce type d’interface. C’est pourquoi depuis 2ans, l’équipe MeForBio fait appel à des étudiants de l’option informatique pour développer une interface graphique permettant tout d’abord une représentation des modèles mais aussi une utilisation plus facile de Pint.

Les modèles de frappes de processus permettent de modéliser les mécanismes par lesquels des protéines sont générées depuis différents gènes, et l’influence qu’à l’émission d’une protéine particulière sur un gène donné. La simplicité du formalisme proposé par cette modélisation permet entre autres d’effectuer des analyses statistiques efficaces et rapides de systèmes biologiques complexes.

## Le modèle de frappes de processus

Le modèle de frappes de processus permet la modélisation des réseaux biologiques de régulation (BRN).

L’objectif étant de décrire mais aussi d’établir une interprétation abstraite des scénarios de frappes de processus pour en approximer les états de stabilité, le formalisme employé et les représentations utilisées sont les suivants :

Représentation d’un modèle de frappe de processus

* les sortes regroupent un ensemble fini de processus (exemple : la sorte b regroupe 2 processus, b\_0 et b\_1), un sorte est la modélisation d’un couple constitué d’un gène et de la protéine associée. Chaque processus représenter un état de la sorte, si on a une sorte avec 3 processus : à 0, la sorte est désactivée, à 1 elle est activée, la protéine sera produite, à 2, la sorte est activée et la protéine sera produite en abondance.
* à chaque instant, un et un seul processus de chaque sorte est activé,
* les actions sont de la forme "processus de a\_k frappe le processus b\_i pour le faire passer à b\_j" où a et b sont des sortes (une action correspond à une frappe suivie d’un bond (ou saut)).

## L’application gPH

### Travail déjà réalisé

Le premier groupe ayant travaillé sur le projet a posé les bases de l’interface graphique du logiciel avec le choix de travailler avec le framework Qt ainsi que la bibliothèque Graphviz qui permet le calcul des graphes. Ils ont aussi mené un travail important sur le *parsing* d’un fichier .ph, qui est le fichier texte représentant un modèle de frappes de processus, qui permettait ainsi de pouvoir créer un graphe mathématique sur lequel pouvait travailler Graphviz mais aussi de créer les objets graphiques représentables sous Qt. A l’issu de leur travail, le logiciel permettait l’ouverture d’un fichier .ph et son affichage dans une fenêtre de manière statique. Il y a eu aussi l’implémentation de certains outils Pint, qui pouvaient être ainsi appelés depuis la fenêtre.



Visualisation d’un graphe lors de la première version du logiciel

Par la suite, le travail du second groupe a porté principalement sur l’implémentation de fonctionnalités permettant plus de dynamisme et d’interaction avec le graphe affiché : on pouvait ainsi déplacer des sortes par un mécanisme de *drag and drop.* Il était aussi possible d’effectuer des zooms +/- sur le graphe. Il y a eu aussi implémentation de nombreuses fonctionnalités permettant une personnalisation de la part de l’utilisateur : création de groupes de sortes, choix de la couleur du fond, des sortes et des groupes, réduction et déploiement des panneaux affichant les groupes et le fichier .ph.



Visualisation d’un graphe lors de la seconde version du logiciel

### Le travail demandé

Dans la dynamique du groupe précédent, le travail qui nous a été demandé était d’améliorer cette interface afin de toujours faciliter au mieux l’utilisation du logiciel auprès de biologistes n’ayant pas forcément de connaissances poussées en informatique. Pour cela il nous a été d’une part demandé de résoudre certains bugs du logiciel qui empêchait une bonne utilisation et d’autre part d’implémenter de nouvelles fonctionnalités. L’ensemble des tâches demandées sont listées et détaillées dans le cahier des charges joint avec ce rapport. Voici une liste en faisant une description succincte :

* Résolution du bug « Save as »
* Résolution du bug de non-coïncidence entre les flèches de frappes et de bonds
* Problème d’optimisation d’affichage des graphes
* Mis à jour du graphe via les modifications sur le fichier texte
* Sauvegarde des paramètres utilisateurs (positions des sortes, couleurs des sortes et du fond, groupes créés et leur couleur)
* Module de branchement d’outils Pint
* Résolution du bug de fonctionnement de gPH sous Ubuntu français
* Mettre un message indiquant « chargement en cours » lors chargement d’un fichier .ph
* Coloration du fichier texte suivant une règle syntaxique
* Permettre la sélection d’une suite de processus
* Résolution du bug de fonctionnement de l’outil Pint « Compute Reachability »

# Outils, logiciels et dépendances

## Frameworks et bibliothèques

En lançant ce projet, les groupes précédents ont fait le choix d’utiliser certaines technologies, notamment des bibliothèques et des frameworks particuliers qui vont comporter des classes déjà définies avec des méthodes déjà utilisables. Nous allons par la suite décrire le rôle de chacune de ces technologies.

### Axe

Axe est un générateur de *parser* écrit en C++11. Il est utile pour parcourir et analyser les fichiers .ph. Dans l’application gPH, il sert à créer une abstraction objet d’un modèle de frappes de processus à partir d’un fichier .ph.

### Boost

Boost est un ensemble de bibliothèques libres écrites en C++ qui étendent les fonctionnalités du langage. Le projet utilise les bibliothèques suivantes :

* *Smart Pointer*: permet l’utilisation de pointeurs avec compteur de référence pour optimiser la gestion de la mémoire,
* *Filesystem*: permet l’accès au système de fichiers,
* *Exception*: étend le fonctionnement de la levée d’exceptions.

### Graphviz

Graphviz est une bibliothèque écrite en C qui permet de calculer des représentations graphiques de graphes (au sens mathématique du terme) selon différents algorithmes de positionnement. L’API de cette bibliothèque n’est disponible qu’en C.

### Qt

Qt est un *framework* open source écrit en C++ qui permet de développer des interfaces graphiques indépendamment de la plate-forme ciblée. L’application développée requiert la version 4.8 du logiciel Qt.

Qt est un outil extrêmement puissant avec lequel nous avons pris plaisir à travailler notamment grâce la documentation complète et détaillée qu’on peut trouver sur internet. Pour appréhender au mieux cette technologie, nous conseillons aux futurs groupes allant travailler sur ce projet de suivre le tutoriel Qt sur le site du zéro : <http://www.siteduzero.com/informatique/tutoriels/programmez-avec-le-langage-c/introduction-a-qt> .

## Programme Pint

Pint est un *framework* qui fournit des outils en ligne de commande utiles au traitement des graphes de frappes de processus. Pour effectuer ces traitements, l’application fait appel aux exécutables du projet Pint, notamment phc, ph-exec, ph-stat et ph-stable.

## IDE : Qt Creator

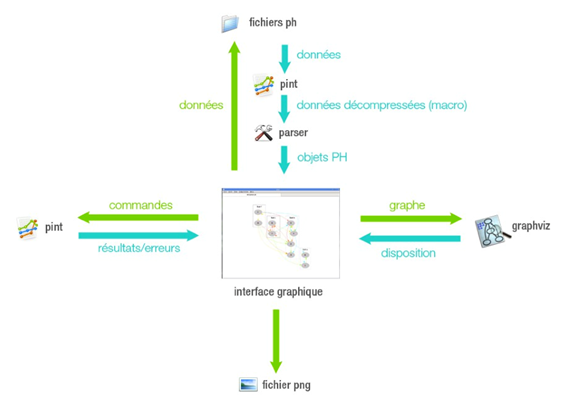
L’utilisation du *framework* Qt entraîne mécaniquement le choix de l’environnement de développement intégré QtCreator, qui lui est dédié. Nous avons utilisé la version 2.4.1, non commerciale.

## Gestion de versions

Afin de gérer les différentes versions de la base de code et de permettre un développement collaboratif, nous avons utilisé le programme Gitavec un dépôt web public sur le site Github.

## Architecture globale

Le schéma ci-dessous récapitule les différents échanges entre modules et bibliothèques qui se produisent lors de l'utilisation de l'application. Le code source de l'application est réparti en différents paquets qui correspondent à ces fonctionnalités.



Vue globale du système

Lorsqu’on veut représenter un fichier .ph via gPH Interface, ce dernier fichier .ph va être *parsé*, le logiciel va dès lors identifier les différents éléments du modèle : les sortes, les processus et les frappes. gPH interface va alors créer un graphe mathématique contenant ces informations et sur lequel pourra travailler Graphviz. Pour Graphviz, un processus est un *node,* ces nodes forment des groupes qu’on appelle des *clusters* qui vont représenter les sortes. Enfin les frappes sont des *edges.*

Avec ce graphe mathématique, Graphviz va alors via son algorithme, calculer la position optimale des clusters, nodes et des edges pour que le graphe soit le plus lisible. Dès lors, gPH Interface va récupérer les informations intéressantes sur les objets clusters, nodes et edges : leur position et leur taille notamment, pour ensuite créer les objets graphiques correspondant sous Qt : les GSort, GProcess et GAction. Ce sont ces derniers qui vont être affichés dans la fenêtre graphique.

Pour résumer, on aura 3 types d’objet pour représenter les éléments d’un modèle de processus de frappes :

* Les classes sort, process et action qui sont les représentations des sortes, processus et frappes au niveau du fichier .ph
* Les classes GCluster, GNode et GEdge qui sont les représentations des sortes, processus et frappes pour Graphviz
* Les Classes GSort, GProcess et GAction qui sont les représentations graphiques des sortes, processus et frappes sous Qt

# Livrables